

Multiresolution Motif Discovery in Time Series

Nuno Castro*

Paulo Azevedo†

CCTC – Department of Informatics
University of Minho, Portugal
{castro, pja}@di.uminho.pt

Abstract

Time series motif discovery is an important problem with applications in a variety of areas that range from telecommunications to medicine. Several algorithms have been proposed to solve the problem. However, these algorithms heavily use expensive random disk accesses or assume the data can fit into main memory. They only consider motifs at a single resolution and are not suited to interactivity. In this work, we tackle the motif discovery problem as an approximate Top-K frequent subsequence discovery problem. We fully exploit state of the art iSAX representation multiresolution capability to obtain motifs at different resolutions. This property yields interactivity, allowing the user to navigate along the Top-K motifs structure. This permits a deeper understanding of the time series database. Further, we apply the Top-K space saving algorithm to our frequent subsequences approach. A scalable algorithm is obtained that is suitable for data stream like applications where small memory devices such as sensors are used. Our approach is scalable and disk-efficient since it only needs one single pass over the time series database. We provide empirical evidence of the validity of the algorithm in datasets from different areas that aim to represent practical applications.

Keywords

Time Series, Motif Discovery, Frequent Patterns, Multiresolution

1 Introduction

The extraction of frequent patterns from a time series database is an important data mining task. These patterns, also known as motifs, provide useful insight to the domain expert about the problem at hand [13] and help summarize/represent the time series database. For that reason, motif discovery has been used in

areas as different as telecommunications, medicine, web, motion-capture and sensor-networks. For example, in EEG time series (figure 1) a motif may be a pattern that usually precedes a seizure; in DNA it may be a sequence of symbols that has been preserved through evolution [13]; in motion capture a particular gesture (e.g. throw ball); in telecommunications, a typical burst in traffic when major social events are located near an antenna. Figure 1 shows one example of such type of pattern in the context of EEG time series from [17]. This specific motif is detected in two different time series of the database. Figure 2 shows the three occurrences of the motif in figure 2 along the same axis after normalization. Normalization of the time series is required to remove offset and scaling effects [5]. It has been shown that comparing time series that are not normalized is meaningless [3]. After normalization the series are very similar but not exactly equal. The aim of this work is to find frequent patterns but still be able to handle noise and other distortions in the time series.

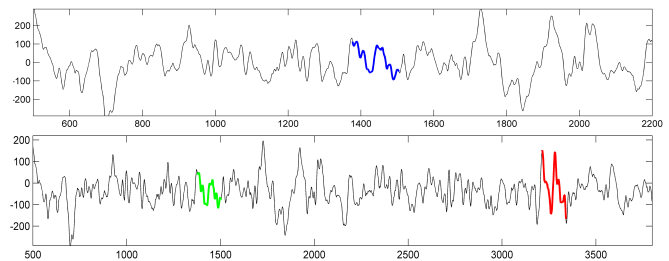


Figure 1: Example of a motif of length 128 in EEG time series in its original context.

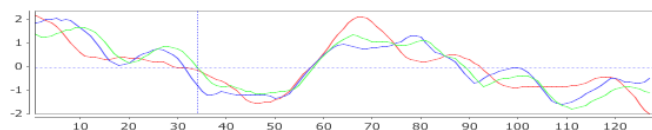


Figure 2: Example of the 3 instances of the motif in the same referential.

*Nuno Castro is supported by *Fundação para a Ciência e a Tecnologia* grant SFRH/BD/33303/2008.

†Paulo Azevedo is supported by *Fundação para a Ciência e Tecnologia*, Project PFAM, Project ProtUnf, FEDER and Programa de Financiamento Plurianual de Unidades de I&D.

Most of the motif mining proposals heavily rely on random disk accesses to the disk database [4, 13, 21] which is inefficient. It is known in the database community that accessing 10% of a disk database randomly takes essentially the same time as traversing the entire database sequentially [18]. Even for moderate sized datasets this becomes an issue. Other techniques [6, 10, 16, 17], tackle this problem by putting the entire dataset in main memory. The assumption that the data can fit in main memory is incorrect most of the time. While this may work in small synthetic datasets, it becomes an unfeasible task when using real world databases that typically present Gigabytes data. Researchers have countered this problem by using approximate time series representations [5]. The aim is to convert the time series database to a representation that requires less space but still retains most of the information in the series. Then, the converted dataset is loaded into main memory using one sequential disk scan. The problem is solved in main memory and only few accesses to the original disk data are required in order to confirm the results [5]. It is straightforward to observe that loading the entire representation of the time series database in main memory is unfeasible for massive datasets. Unless the dimensionality reduction factor yield by the new representation is large, causing the loss of the most important time series features, the assumption that the representation of the time series fits in main memory does not hold. There are very specific applications where storing the representation is also not possible, such as sensor networks or mobile applications. These applications use devices where the amount of memory is limited, requiring space efficient algorithms. Furthermore, in some of the applications the entire dataset is not available and the algorithms are often not prepared to handle this important type of scenario e.g. data streams. A similar setting is that of massive databases. In these cases the amount of data on disk is very large and each element can only be visited once.

There are many methods designed for univariate time series. However, many time series are multivariate, for example multi-sensor systems [16]. Also, most existing approximate algorithms only find motifs at a single resolution i.e. using a fixed number of symbols in the time series representation. However, it is desirable to handle different levels of the time series representation to achieve further insights about the data.

Data mining algorithms typically provide as output a set of patterns: the most recurrent patterns, the nearest neighbor subsequences in the database, etc. We believe that user interactivity is an important part of data mining. For that reason data mining and motif

discovery algorithms should provide means to facilitate the development of user interactivity.

In this work, we tackle time series motif discovery as an approximate Top-K frequent pattern problem. We provide a scalable algorithm to retrieve the most frequent subsequences of the database. We achieve time efficiency by using a single sequential disk scan to read the time series database, a clever time series representation and a hashtable based counting technique. Memory efficiency is achieved by combining our method with the space-saving algorithm [9], now applied to time series data mining. Our approach is based on the state of art time series representation – iSAX [20], exploiting its multiresolution property to derive motifs at different resolutions. This property enables the development of powerful visualization applications, allowing the navigation in the Top-K motifs hierarchy structure. This yields better understanding and intuition about the database at hand to the user. The single sequential disk pass makes the method a strong candidate for data streaming applications.

The remainder of the paper is organized as follows: section 2 describes the state of the art in time series motif discovery with a particular subsection on the iSAX representation; background and notation used throughout the paper are described in section 3; section 4 introduces our approach; the experimental analysis is shown in section 5; finally, we derive conclusions and discuss future work.

2 Related work

Since the definition of motif discovery in time series in [2], several algorithms have been proposed to tackle this problem [4, 6, 10, 16, 17, 21]. The first proposal ([2]) defines the problem of motif discovery in time series regarding a user-defined range parameter R and a motif length m . That is to say, two subsequences of length m match and form a motif if their *Euclidean Distance* is smaller than R . The concept of motif is generalized to K -Motifs, where the top K motifs are returned. The 1-Motif, or the most significant motif in the time series, is the subsequence that has most non-trivial subsequence matches. The exact motif discovery solution for a time series presents quadratic complexity. For that reason, optimizations based on the symmetry and triangular inequality properties of the Euclidean Distance are proposed. This leads to complexity reduction by a large constant factor. However the approach is still intractable for large datasets [6]. It is also space inefficient.

Due to the quadratic complexity of exact algorithms, researchers have changed focus to approximate solutions. These solutions present in general $O(m)$ or

$O(m \log m)$ complexity with very high constant factors [21]. The first work to consider this approach is [6]. It is based on research for pattern discovery from the bioinformatics community [1]. The authors develop an algorithm to find motifs in linear time – Random Projection. The algorithm is robust against noise and uses a probabilistic and iterative approach. It uses as base structure a collision matrix whose rows and columns are the SAX representation of each time series subsequence. The subsequences are obtained using a sliding window approach. At each iteration, it selects certain positions of each word as wildcards and traverse the word list. For each match, the collision matrix entry is incremented. In the end, the largest entries in the collision matrix are selected as motifs candidates. Finally each candidate is checked for validity in the original data. This algorithm has been widely used in time series motif discovery since its introduction. It is robust to noise and can find all the motifs with high probability, after a certain number of iterations. It presents linear complexity in terms of the number of subsequences, word length, number of iterations, and number of collisions [16]. If the distribution of the projections are not sufficiently wide, i.e. if a large number of subsequences have the same projection, the algorithm becomes quadratic in time and space [16]. The space efficiency of the algorithm is based on the sparse implementation of the collision matrix. Also, the algorithm assumes that the collision matrix can fit into main memory, which in some scenarios is not the case. It presents several parameters that need to be tuned: range R , motif length m , number of columns masked in the collision matrix c , and the SAX alphabet size a and word length l . Some of these parameters are unintuitive for the user e.g. how to optimize c and R . This can be attained only by experimentation, which is most of the time unfeasible for large datasets. Failing to achieve optimal parameter values can lead to misleading results: no motifs found, a massive number of motifs or meaningless motifs being found. The SAX parameters have been experimentally shown to have little impact on the quality of results [5] and can be set by default. The motif length m is a relatively more intuitive parameter to tune even for the ordinary user.

Oates introduced an algorithm called PERUSE to find recurring patterns in multivariate time series [4]. This algorithm can handle data sampled at different rates and motifs having arbitrary lengths. It uses the raw time series instead of discrete approximations. However it presents a large computational complexity and some stability problems on the estimated models.

In [10], the authors introduce an algorithm to find motifs in multivariate time series. They use Principal Component Analysis to transform the multivariate time

series into one signal. The univariate result data is handled using Chiu’s Random Projection [6].

Another approach to find motifs in time series of Proteins is described in [13]. The algorithm receives as inputs a database of time series, a minimum support δ , the motif length m , a minimum similarity R_{min} , and a window frame length $deltaW$. It first converts the time series subsequences into the SAX representation. Then, it attempts to find clusters of subsequences (matches). Finally, each retrieved motif length is extended until the similarity drops below the user-defined R_{min} threshold. This algorithm is able to handle multivariate time series. It presents a more intuitive R parameter (e.g. $R = 1$ means maximal similarity) and also allows to retrieve motifs of different sizes and inverse motifs (symmetric shapes). The motif length extension capability seems desirable in motif search. Its limitations are the quadratic complexity and the need to load the entire dataset to main memory, which is untenable even for databases of moderate size.

Minnen et al. introduce an algorithm to find motifs in multivariate time series in linear time [16]. In this work, motifs may span only across some of the time series, or dimensions – subdimensional motif discovery. The algorithm is also based in the proposal presented in [6]. The difference is due to the multivariate case, where a combined projection is necessary. Whereas in previous work by the same authors the collision matrix is incremented only if all dimensions matched a particular subsequence, now it occurs once there is a match in one of the dimensions. The Dynamic Time Warping (DTW) is also tested as the distance measure. However the complexity rises to quadratic. The authors use the 10% Sakoe-Chiba band constraint in the warping path. It has been shown that when using the LB_Keogh lower bounding technique DTW becomes essentially linear [12]. However, Keogh has shown recently in [19] that for large time series databases, using the DTW is not significantly better than the Euclidean Distance, being the latter much faster. The strengths of this algorithm are its ability to handle the multivariate case and creating the conditions to avoid Random Projection from growing quadratically. That is, it makes the distribution of the collisions sufficiently wide by dynamically adjusting the SAX’s alphabet size and c parameters.

In [17], the authors present an algorithm to handle stretching in the time axis of time series. This phenomena can happen if the sampling rate of two time series is different. The approach extends Random Projection to allow the detection of motifs under uniform scaling. The definition of motif is also modified and it is no longer centered on an unintuitive range parameter. There, mo-

tifs are defined in *nearest-neighbor* terms: the motif of a time series is constituted by the two subsequences that are the nearest to each other, i.e. present the lowest distance from each other. The only parameter that needs to be defined by the user is the motif length (besides SAX's parameters). This algorithm inherits Random Projection's problems and increases its overhead due to the need to search for the best scaling factors.

Under the new *nearest-neighbor* motif definition, Abdullah Mueen et al. [21] proposed the first tractable exact motif discovery algorithm – *MK*. This algorithm is up to three orders of magnitude faster than brute-force. It is based on the notion of early abandoning the Euclidean Distance calculation when the current cumulative sum is greater than the *best-so-far*. The motif search is guided by heuristic information calculated by the linear ordering of the distance of an object with respect to a few random reference points. *MK* is a sound contribution in the exact motif discovery search. However, the use of the Euclidean Distance directly in the raw data can give rise to robustness problems when dealing with noisy data. Euclidean Distance can be highly affected by noise [6]. Also, when data does not fit in main memory it will perform a large number of random disk accesses, which may prevent the algorithm from scaling.

Despite the recent research in exact algorithms, we believe approximate algorithms will continue to be the best option in many application domains due to its time/space efficiency. For instance, sensor networks and telecommunication networks monitoring need most of the time real time results. In these domains, the trade-off between execution time and accuracy of the solution clearly bends towards the former.

The process of counting subsequences occurrences in time series is not trivial. The criteria for which one could consider one subsequence as the repetition of another are diverse. It is clear that counting only equal subsequences as repetitions is of no great use. One should use "similar" subsequences instead. Once this assumption is considered, the next question to address is what similarity definition one should use. To let the data mining practitioner set this as a parameter such as the range (R) is not an interesting approach, since this parameter is largely domain dependent and unintuitive to adjust. One could choose to select the pair of subsequences in the database that are the nearest to each other (*nearest-neighbor* motif definition). However, this solution does not take into account the frequency of the subsequences. We may have a pair of subsequences that are the nearest neighbors but are rare in the database (e.g. one occurrence). An interesting approach is to formulate the motif discovery problem as a top-K fre-

quent pattern problem [9]. In this framework, we need a container where we can put similar time series with an adjustable margin of similarity. Approximate time series representations appeared to be the best solution. Among these, the Symbolic Aggregate Approximation (SAX) has been shown to outperform other approaches [19]. SAX has been widely used in the time series data mining community. The most important features of this approximation is that it reduces the dimensionality and lower bounds the true distance of the original time series. Despite losing some of the information in the reduction process, it conserves the overall shape of the time series. The average calculation of the Piecewise Aggregate Approximation (PAA) is good against noise, except if sudden variations are the important aspect of an application domain. As shown in [15] in the context of clustering, SAX sometimes outperforms Euclidean Distance on noisy data. This results from the smoothing caused by dimensionality reduction. SAX has been further enhanced to iSAX [20]. The built-in multiresolution property made the original SAX even more attractive, since adjusting the margin of similarity to use i.e. increase or decrease the iSAX resolution, becomes a build-in available feature.

iSAX Representation

As a symbolic approximation, SAX converts the original real time series T of length n into a sequence of w symbols – *word* – belonging to an alphabet of size a . The alphabet size of a given SAX word is called *resolution*. This operation is represented by the following notation: $SAX(T, w, a)$. SAX operates as follows:

- First, the dimensionality of the time series is reduced by dividing it into w segments (word length) with the same length ($\frac{n}{w}$) using the Piecewise Aggregate Approximation (PAA) algorithm. This algorithm assigns to each segment its average value.
- Then, the amplitude of the time series is divided into a intervals, so that a symbol can be assigned to each interval. The best way to generate equiprobable intervals is to use $a - 1$ breakpoints that produce the same area under the Normal curve, as shown in figure 3. These breakpoints can be obtained by using a statistical table.
- Finally, symbols are obtained from the intervals. The segments below the smallest breakpoint are assigned the 0 symbol. The segments between the first and second breakpoints the symbol 1, and so forth. In order to assist calculations, binary numbers are used instead of actual symbols.

For clarity, these are typically displayed in their decimal format. Figure 3 a) and b) depicts this idea for resolutions of 4 and 16, respectively.

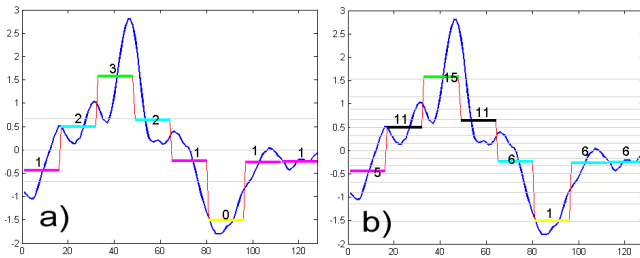


Figure 3: Example of the SAX conversion process for a time series with length 128, $w = 8$ and resolutions: a) 4, b) 16. Image generated by MATLAB and code provided by SAX authors [5]

The iSAX representation extends classic SAX by allowing different resolutions for the same word. To avoid ambiguity, the resolution of each symbol needs to be made clear in the iSAX word. It is this enhancement that enables the creation of a time series index. However, for the scope of this work, we are not interested in the indexing capabilities of iSAX. Rather, we are interested in interleaving between different resolutions within the same iSAX word.

Converting from a higher to a lower resolution is simple: one just needs to ignore one trailing bit as we reduce the resolution by half. However the opposite is not true, since one can have several possibilities for the higher resolution. We need to convert the original time series to the new resolution if we want the correct result. Later, details on how this step is performed in an efficient way will be described. An important issue to consider is that as the resolution increases, the more similar two time series need to be in order to originate the same word. Each interval narrows considerably each time we duplicate the resolution of the iSAX word. This intuition can be observed in figure 3.

3 Background and Notation

In this section we introduce some notations and useful definitions. First we define our object of study.

DEFINITION 3.1. A *time series* T of length n is an ordered succession of a variable's observations (t_1, \dots, t_n) over time, with $t_i \in \mathbb{R}$.

We are typically interested in mining a collection of time series with arbitrary lengths.

DEFINITION 3.2. A *time series database* D is a set of $|D|$ unordered time series ([21]).

Time series data mining algorithms often use subsections, or subsequences, of the original time series in their calculations.

DEFINITION 3.3. Given a time series T of length n , a *time series subsequence* $S = s_i, \dots, s_{i+m-1}$ is a sampling of $m \leq n$ contiguous positions of T , such that $1 \leq i \leq n - m + 1$ (definition from [6]).

A special type of time series can present several variables varying over time. One example of a multivariate time series are EEG recordings in several electrodes placed in the scalp over the period of one minute.

DEFINITION 3.4. A *Multivariate time series* is a set of several time series (variables) in the same time range.

A time series can be generated through a device or process that is continuously deriving data. In this specific case it is called streaming time series, or a data stream. We now formalize this concept.

DEFINITION 3.5. A *Streaming time series* X is a time series with $n = \infty$, whose data points arrive continuously at an arbitrary rate.

This type of time series is present in many application domains e.g. data captured by sensor networks. In this case we may be in the presence of multivariate streaming time series. Typically, researchers transform streaming time series into static time series by defining an end point to the time series (length n to a specific value). Then, a traditional offline data mining algorithm i.e. an algorithm where the time series does not change during its execution, is applied to the truncated time series. Online (or real-time) algorithms go beyond this approach and can be applied to streaming data. As soon as a new data point is available the internal state of the algorithm is updated according to this new point. Our algorithm, as we shall see, exhibits this property.

DEFINITION 3.6. A subsequence S with length m is an *instance* of an iSAX word W if $iSAX(S, w, a) = W$, for a given word length w and alphabet size a .

Both w and a parameters do not need to be set by the user. Rather, they are part of the algorithm process as we shall see in section 4.2.

DEFINITION 3.7. The *cluster* of the iSAX word W is the set of all instances of W in D .

We now clarify the goal of our data mining task. We aim to find the top-K motifs.

DEFINITION 3.8. The *Top-K Motif* of a time series database D is the cluster ranked at the k^{th} position regarding number of instances.

Notice that our motif definition does not consider a distance measure. This is due to the multiresolution property of our algorithm which we inherit from iSAX. As we increase the resolution, finding a cluster for a given iSAX word becomes increasingly difficult. Our intuition is that at the largest resolution, we will be working very close to the level of raw data. For that reason, finding a cluster at a large resolution implies that the Euclidean Distance among the instances is very small. This assumption will prevent us from performing expensive distance calculations.

4 Our algorithm

In this section we describe the proposed algorithm. We start by briefly describing the space-saving algorithm for frequent items mining.

4.1 Space Saving algorithm

The Space-Saving [9] (*SS*) algorithm was proposed to efficiently compute frequent elements in data streams. To the best of our knowledge, this algorithm has never been applied to time series motif discovery as it can not be applied to raw time series. The iSAX representation outputs a discrete sequence of symbols, which is suitable to apply the SS algorithm.

Space-Saving is a relatively simple algorithm. Suppose we want to compute the top m elements of a data stream. The algorithm maintains only m counters. These counters are updated such that the number of occurrences of the significant elements are accurately estimated. If the observed element e is in the monitored group then its frequency is incremented. Otherwise, the element e_m with the least estimated hits min in the monitored group is replaced by the observed element and the counter of that element is incremented. The algorithm's main goal is to never miss a frequent element. However, e could actually have between 1 and $min + 1$ hits.

Algorithm 1 Space-Saving(m counters, stream S)

```

for each element,  $e$ , in  $S$  do
  if  $e$  is monitored then
    let  $count_i$  bet the counter of  $e$ 
    Increment-Counter( $count_i$ )
  else
    let  $e_m$  be the element with least hits,  $min$ 
    Replace  $e_m$  with  $e$ 
    Increment-Counter( $count_m$ )
    Assign  $\varepsilon_m$  the value  $min$ 
  end if
end for

```

Space-Saving is one of the most efficient techniques

for estimating top frequencies in terms of space. Nevertheless, it is experimentally shown that monitoring only a moderate number of counters guarantees very small errors. Also, for each monitored element e_i , the maximum-overestimation ε_i for that element is saved, which is the value that the counter presented when the element was first inserted in the list. This gives an upper-bound in the over-estimation errors.

4.2 Multiresolution Motif Discovery

In this section we describe the Multiresolution Motif Discovery in Time Series algorithm – *MrMotif*. The algorithm is based on the iSAX representation. The main idea of the algorithm is to start from a low iSAX resolution and then expand to higher resolutions. As this expansion is performed, the number of instances of a given cluster reduces as each cluster is split into several of the next resolution. At the highest resolution, a cluster is formed only if the subsequences in that cluster are very similar, as each iSAX symbol covers only a narrow interval in the amplitude of the time series. This idea can be observed in figure 3.

For simplicity we assume the time series database D is available on disk. Regardless, each raw time series is not consulted more than once. Hence, the algorithm can be directly applied to streaming data. The minimum possible resolution g_{min} in iSAX is 2. The maximum resolution g_{max} is assigned to 64. Resolutions bigger than 64 are most of the time at the level of the raw series, where it is not possible to find clusters, or only trivial clusters (equal time series). Note that we are not interested in analyzing all resolutions in between g_{min} and g_{max} . Rather, we only aim to study the g_{min} powers of 2 until we reach g_{max} . That is to say, we use the 2, 4, 8, 16, 32 and 64 resolutions. We maintain a set of hashtables $count$ in main memory, one hashtable $count_g$ per resolution. Thus, pairs of ($cluster, count$) are stored.

Our algorithm aims to find the solution for the following problem:

Problem definition: Given a time series database D , a motif length m and a K parameter, for each resolution in $(g_{min}, g_{min} \times 2, \dots, g_{max})$ find the top- K motifs.

We describe the pseudo-code in Algorithm 2. The actual implementation can be accessed at [23]. For simplicity, we describe the algorithm without considering details about memory usage or cluster hierarchy. We will detail later when exactly the Space-Saving setting is activated and how the information that will allow visual tools to navigate through the motif structure is saved. For the time being, assume Space Saving is not active (line 9), and ignore line 3. The algorithm is relatively straightforward. A sliding window of size m is

Algorithm 2 MrMotif(D, m, K)

```
1: for each subsequence  $S$  of length  $m$  in  $D$  do
2:    $W \leftarrow iSAX(S, g_{min} \dots g_{max}, w)$ 
3:    $motifTree.Update(W)$ 
4:   for each  $w_g$  in  $W$  do
5:     if  $w_g$  is in  $count_g$  then
6:        $c_g \leftarrow count_g.get(w_g)$ 
7:        $count_g.Update(w_g, c_g + 1)$ 
8:     else
9:       if Space-Saving is Active then
10:         $(e_m, min) \leftarrow count_g.getMin()$ 
11:         $count_g.Update(w_g, min + 1)$ 
12:         $\varepsilon_m = min$ 
13:         $count_g.updateMinimum()$ 
14:      else
15:         $count_g.Update(w_g, 1)$ 
16:      end if
17:    end if
18:  end for
19: end for
20: return count
```

used to scan the subsequences of all time series T_i (with possibly different sizes) in database D . Also a bounded buffer of size m is used to keep this disk traversal sequential. We are aware that contiguous subsequences are likely to be almost identical and for that reason a step greater than one is used in the sliding window approach. This prevents spurious motifs from being found, also known in the literature as "trivial matches" [6]. Each read m -length subsequence is converted to an iSAX word for each resolution of interest (line 2). Note that this conversion is executed in one single step, as for the same time series most of the conversion process is similar at all resolutions (only the symbol lookup is independent). Then, if the cluster exists in the corresponding hashtable $count_g$ structure, its count is incremented and the location of the subsequence saved (lines 4 – 7). Otherwise it is set to one (line 15). Finally, the top- K Motifs for each resolution are outputted (line 20).

Space-Saving

In section 4.1 we have described the Space-Saving algorithm without referring when it is activated and which elements to monitor. The intuition is to directly apply it to the Top- K motif problem. However, this would not yield satisfactory results because this K set is typically very small (for instances, Top-10). Thus, it could potentiate the number of over-estimation errors. Instead, we let the user decide the maximum amount of memory the algorithm's implementation has available. The amount of memory the algorithm is using is monitored.

If the algorithm reaches the user defined threshold the Space-Saving mode is activated. In that case, the algorithm will execute lines 9–13. For example, in a mobile device this threshold can be set to 1 MB. It is also possible that the user chooses not to set this threshold. In that case the algorithm is executed until no more memory is available. As this situation can make a system stop operation, we actually "hard-code" this threshold to 99% of the available memory. One could argue that by using a clever representation technique as we do, this will hardly occur. However, that is not the case. We will show this situation using a relatively small time series where the number of different clusters in the counters hashtable increases at a fast rate. On the other hand, having initialized the system in "full memory" mode provides us a large enough number of counters to ensure very small errors. The use of ε_m provides guarantees about the quality of a given execution of the algorithm.

Interactive Visual Tool

In section 1 we have discussed that data mining algorithms should provide rich outputs. This would facilitate the development of applications that receive as input that same output, such as Visualization applications [8]. Hereby we show an example of such application, as we believe these provide the data mining practitioner with further understanding and intuition about the data at hand.

Our example application is a motif navigator, which allows to perform several exploration and "drill-down" operations along the motif hierarchy. During the algorithm's execution, an iSAX word is generated for each subsequence within each resolution. Larger resolution words are contained in smaller. In this sense we say that the discovered motifs form an hierarchy. For example, the length 2 word $(4^8, 3^8)$ contains the words $(9^{16}, 7^{16})$ and $(8^{16}, 7^{16})$. For that reason, we say that the lower resolution is the parent motif and the higher is the child. The words generated for the same subsequence form a word family. It is straightforward to keep and maintain this information in a tree structure. Line 3 of the algorithm performs this maintenance operation. The motif tree structure can then be given to a graphical user interface in the form of a graphical tree (similar to a file system tree). The user can explore and visualize the motifs at different resolutions, in order to realize which of the frequent motifs are significant for his particular domain/problem. Figure 4 displays a screenshot of this motif navigator where the motif hierarchy structure can be observed.

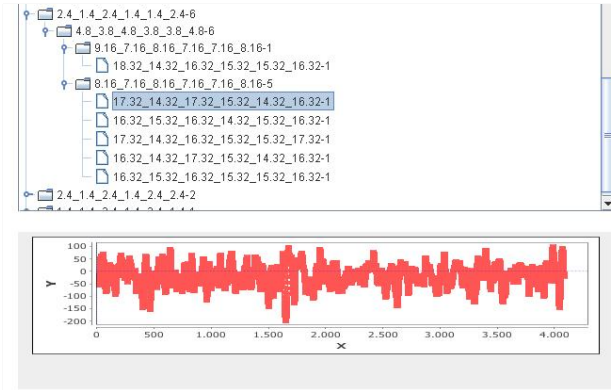


Figure 4: Snapshot of a motif navigator

5 Experimental Analysis

In this section we perform experiments to validate the impact of the proposed algorithm. First the space and time scalability is analyzed in a large synthetic database. Then, the effect of noise in the algorithm is studied. Finally, the impact of MrMotif is shown in three different real applications. The experiments were performed on a machine with a Quad-Core AMD Opteron™ Processor 2352 with 16 GB of RAM. The MrMotif algorithm is implemented in Java and the compiler used was the JDK 6.

5.1 Scalability Experiments

The experimental analysis begins by considering the scalability of the proposed algorithm. We compare our approach to Random Projection (RP) [6] in terms of execution time. This algorithm is selected for comparison due to its popularity. It is the most cited time series motif discovery proposal up to date (more than 140 citations) and is the basis of many current approaches that tackle this problem [10, 16, 17]. Furthermore, the execution time of this algorithm can be used as a lower bound on the execution time of all approaches that are based on it. We also perform comparisons with the "full memory" (FM) version of our algorithm in order to understand the impact of Space-Saving (SS). The dataset used in this section is constituted by random walk time series available in the MK algorithm [21] website. We select these data for two reasons: they have been used before and results on similar datasets are encouraged in order to walk towards data mining benchmarks; also, the size (in the Gigabytes order of magnitude) makes them attractive to test any algorithm. The dataset is composed by ten different sets of random walk series, with 10000 to 100000 time series of length 1024. These sets occupy a large amount of disk space ranging from 160 MB to 1.5 GB, for a total of about 8 GB in the

database. We reproduce these datasets by following the instructions in [21] website, using the same random seed as the MK's authors. In the MK proposal, the algorithm is executed 10 times for each of the ten increasingly large datasets and the average of the execution time for each dataset is recorded. We follow the same approach with RP and both implementations of MrMotif – SS and FM.

The motif discovery algorithms are executed with $K = 10$ and $m = 1024$. We follow the recommendation of the SAX authors [5] and set $w = 8$ (iSAX word size) for all experiments. The maximum amount of memory used by SS is set to 128 MB. This value is chosen because it is close to the average RAM available in current mobile phones. We implement the RP algorithm and set the parameters $w = 8$ and $a = 8$. The c parameter is randomly chosen between 2 and 7, to assure that the distribution of the projections is wide enough to prevent the algorithm from becoming quadratic. For fairness, we remove the disk verification of candidate motifs (a module part of RP), since MrMotif does not perform this expensive step. Both implementations of MrMotif and RP are available on MrMotif website [23]. Figure 5 displays the results of the execution.

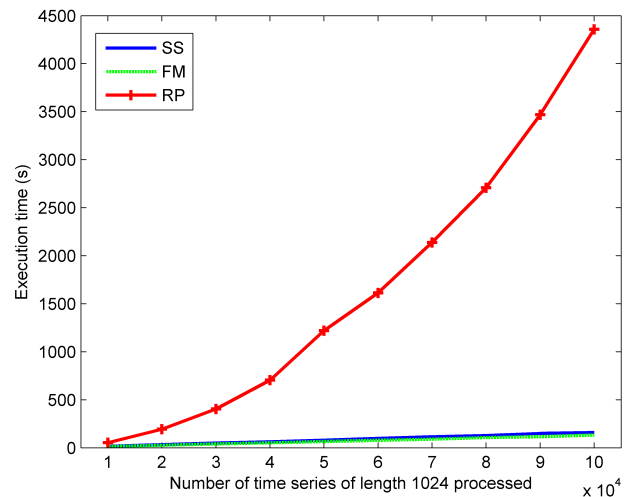


Figure 5: Variation of the execution times of the three algorithms as the number of processed time series increase.

It can be observed that MrMotif is about one order of magnitude faster than RP for each of the ten increasingly sized sets that constitute our dataset. Also, MrMotif execution time increases linearly as the dataset size increases, as expected. However, RP seems to grow quadratically. The reason for this is that RP is quadratic with respect to the SAX word list size. Note that we present results for just one iteration of RP. However, as an iterative algorithm, several iterations

are necessary in order to converge. The reason we show results for one iteration is to make clear that MrMotif full execution outperforms one iteration of RP. A full execution of MrMotif returns the top-10 patterns for the 2, 4, 8, 16, 32, and 64 resolutions deterministically. It can also be observed that the FM version of MrMotif executes faster the SS version. This is due to a small overhead that Space-Saving adds to the algorithm.

The next experiment reports the memory usage of the MrMotif SS and FM versions during the execution in the dataset containing 100000 time series of length 1024. The MrMotif SS and FM versions are compared in this experiment in order to show the impact of Space-Saving. Figure 6 depicts the memory used by the Java Virtual Machine of the FM and SS algorithms versions.

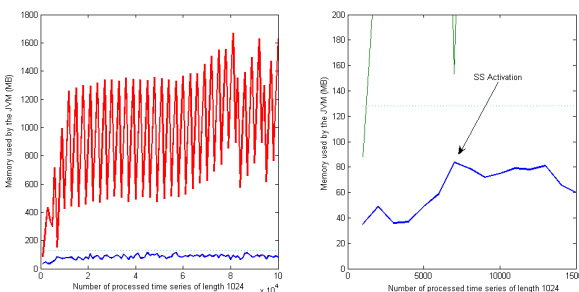


Figure 6: Variation of the memory used by the JVM as the number of processed time series increase. Red: FM, Blue: SS. The right figure zooms in the left bottom quadrant of the chart.

It can be observed (as expected) that when the SS algorithm is activated (time series 6000), the memory used by the algorithm remains below the imposed limit. The wave-form of the memory usage variation can be explained by the effect of the Java garbage collection (GC). The FM version of the algorithm uses a large amount of memory. This is due to the fact that 100000 random time series produce a large number of iSAX words, which quickly fill the hashtables. This also happens with RP or any other algorithm that saves the iSAX representation of all time series in main memory. However, the FM version is only used for experimentation purposes. In real scenarios, we use the SS approach.

In this section we have demonstrated that MrMotif SS version is time and space efficient for relatively large datasets (8 GB). The results show that the algorithm is linear regarding the number of time series in the database. This is due to the use of a single sequential disk traversal and constant time structures (hashtables). It is also observed that the proposed algorithm executes about one order of magnitude faster than RP. This is an encouraging result because RP is the basis of many

existing motif discovery algorithms.

5.2 Experiments with noise

In this section an analysis of the impact of noise in MrMotif results is performed. We start by applying MrMotif to the set of 10000 time series of length 1024 from the previous experiment. We record the top-10 patterns of resolution 4 and use these results as the ground truth for our study. Then, we produce ten noisy variations of our dataset using the technique (and code) in [7]: Gaussian noise and small time warping are added to the original series. Further details of the technique can be accessed in the original paper ([7]) or in our website [23]. For each variation we increase the range of noise introduced, from 10% up to 100% of the original series standard deviation. We apply our algorithm to each of the ten noisy versions of our dataset, recording the information retrieval metrics *precision* and *recall* of each execution with respect to the original (noise free) version. These measures are calculated by using the number of true positives (TP), false positives (FP), and false negatives (FN) for each execution: the TP are the number of *clusters* present in the top-10 of the noisy and original version; the FP are the number of clusters that are incorrectly in the noisy version; and the FN are the clusters that are not in the noisy dataset but are in the original execution's. We have $Precision = TP / (TP + FP)$ and $Recall = TP / (TP + FN)$. The results are shown in figure 7.

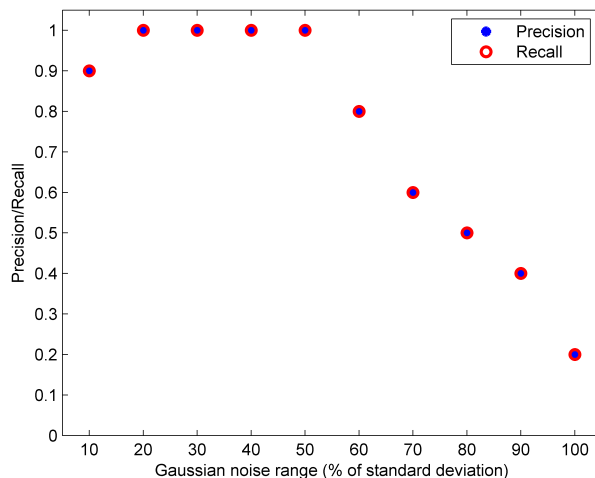


Figure 7: Variation of the Precision and Recall of each increasingly noisier variation of the original 10000 size dataset.

We can observe that precision and recall present values above 90% until the noise range is greater than

half standard deviation. From this point onwards, the noise level causes the series to significantly differ from the original ones. We can conclude that MrMotif is robust to relatively high levels of noise and small time warping. For these levels, few of the top-10 patterns were missed and a small number of false patterns in the top were discovered. This capability is derived from the smoothing effect of the iSAX dimensionality reduction process. In our experiment setting a fixed value for the top patterns (10) is used. For that reason, when our algorithm misses a top pattern (FN) it obviously introduces a spurious one in the top-10 (FP) and vice-versa. Therefore, precision and recall present the same value for this experiment scenario.

5.3 Real Applications

In this section the MrMotif algorithm is applied to three different application areas. Our goal is to validate that MrMotif is a strong candidate for a wide range of applications where time and space efficiency are necessary. Our algorithm is applied to real datasets from the areas of protein unfolding, sensor networks monitoring and telecommunication networks.

Protein Unfolding

Protein folding involves the formation of the 3D structure of a protein from a sequence of aminoacids. Folding or unfolding disorders of a protein cause diseases such as the neurodegenerative Alzheimer's. The Transthyretin (TTR) is one example of such proteins whose unfolding disorders cause severe diseases. Unfolding mechanisms of this protein have been studied by computationally analyzing variations on certain molecular properties over time. The Solvent Accessible Surface Area (SASA) is one example of such properties that are important to study in order to understand the cause of the disorder and consequent manifestation. This analysis is performed by means of simulation from Molecular Dynamics (MD) unfolding of TTR [11]. The dataset is constituted by 127 time series of 10000 points each, corresponding to the variation of the SASA in each of the 127 aminoacids of the protein during a period of 10 nanoseconds (ns): one point per picosecond (ps) of simulation. The actual time necessary to run this $10ns$ simulation surpasses one month. We apply our proposed algorithm to find the Top-10 patterns of size 64 ($64ps$), as this may unveil important repetitions in unfolding behavior among the different aminoacids. The top-1 motif retrieved by our algorithm is discussed. The larger resolution cluster with at least 2 repetitions was discovered at the 16 resolution. The top clusters at resolution 8, 4, and 2 presented 5, 35, and 1029 instances, respectively. In figure 8 one example of a motif found

at the resolution 4 is displayed.

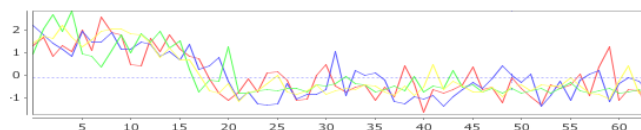


Figure 8: The four instances of a motif discovered at resolution 4.

The discovered motif is repeated 4 times in the database. This motif highlights the robustness to noise characteristic of our algorithm. The time series instances are not exactly equal and present a relatively large Euclidean Distance. Nevertheless, they were successfully counted as repetitions. This capability of MrMotif provides the biologists with further insight on the domain.

Sensor Networks Monitoring

To apply data mining techniques to emerging architectures such as sensor networks is of particular importance. These devices will be widely used in the future in fields as diverse as health, forest fire detection, and general surveillance. Sensors communicate with the sensor base through wireless channels. These operate at a frequency close to the Wifi networks and for that reason are subject to interference and failures. It is then vital to monitor parameters of the sensor networks communication protocols, such as the delay or number of retransmitted packets caused by packet loss. The reason is that a packet loss will cause a retransmission, which is the most energy-expensive operation for these battery-run devices. Our dataset is composed by averaged delay data of a specific sensor in a wireless network of biomedical sensors. There are 9 time series, each covers a monitor period ranging from 7 hours up to 18 consecutive hours. Each data point contains the average delay of packet transmission by the iLPRT MAC protocol [22]. MrMotif is applied to the dataset to find the Top-5 motifs, using a motif length of 16 covering the last 16 minutes, as suggested by the domain expert. The memory limit is set to 1 MB for this particular scenario. This highlights the amount of memory these devices typically have available. Figure 9 shows one example of a motif detected at resolution 8.

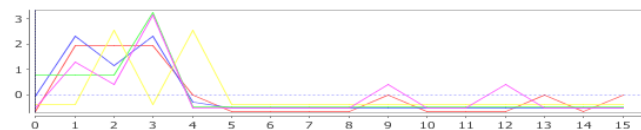


Figure 9: Example of a motif with 5 instances. The variation was due to interference by a laptop's antenna wireless.

This motif presents 5 time series repetitions. The motif has been acknowledged by the domain expert as worthy of further investigation. The displayed motif occurred in situations where the user intentionally approached a laptop to the sensors range in order to cause interference. This provides promising results in further applications of MrMotif and other time series data mining algorithms. The goal is to help improve communication protocols for wireless sensor networks.

Telecommunication Networks

Telecommunications networks interconnect people of different cities, countries and continents. For that reason they play a central role in nowadays society. These networks are characterized by very complex and large structures that are monitored by network operators, using reports of performance counters such as traffic data. For network troubleshooting problems it is interesting to detect frequent patterns. This helps in preventing future failures, obtaining further knowledge about the domain, and achieve better next generation networks. In this experiment MrMotif is applied to a traffic dataset from a Portuguese telecommunications network operator. The data regards several network elements (nodes), whose traffic was recorded in the period of a week at a granularity of 15 minutes. The algorithm parameters K and m were set to 10 and 360, respectively. The goal of the network operator is to attempt to find regularities in the network traffic, possibly at different nodes, over the period of a few days. In figure 10 one example of a motif with two instances returned by our algorithm at resolution 8 can be observed.

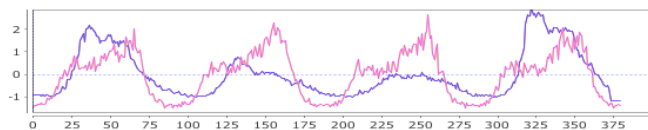


Figure 10: Motif with two instances found at two different nodes.

The network operator has found this result and approach interesting. These motif represents the same characteristics in telecommunications traffic at a given week and the causes remain to be investigated by the telecom operator.

6 Conclusion and Future Work

We have proposed the Multiresolution Motif Discovery in Time Series algorithm – MrMotif. This proposal tackles limitations of existing algorithms such as disk access and memory inefficiency. It brings to time series motif discovery the Space-Saving algorithm in order to efficiently handle strict memory requirements present

in emerging architectures as sensor networks. The multiresolution property inherited by the solid iSAX representation allows to find motifs at different resolutions. This provides useful insight to the practitioner about the database at hand. MrMotif is scalable and can have a strong impact on different application areas due to the good performance and robustness to noise. Future work includes investigating time series motifs evaluation measures [14] and studying automatic methods to derive the parameters K and m .

Acknowledgments

We would like to thank the anonymous referees who helped to significantly improve this paper with their invaluable feedback.

References

- [1] Buhler, J. and Tompa, M., *Finding Motifs Using Random Projections*, in Proceedings of the Fifth Annual international Conference on Computational Biology (2001), pp. 69–76.
- [2] Lin, J., Keogh, E., Lonardi, S., Patel, P., *Finding Motifs in Time Series*, Proceedings of the 2nd Workshop on Temporal Data Mining (2002), pp. 53–68.
- [3] Keogh, E, Kasetty, S, *On the need for time series data mining benchmarks: a survey and empirical demonstration*, in Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002), pp. 102-111.
- [4] Oates, T., *PERUSE: An Unsupervised Algorithm for Finding Recurring Patterns in Time Series*, Second IEEE International Conference on Data Mining (2002), pp. 330.
- [5] Lin, J., Keogh, E., Lonardi, S., and 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 (2003), pp. 2–11.
- [6] Chiu, B., Keogh, E., and Lonardi, S., *Probabilistic discovery of time series motifs*, in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2003), pp. 493–498.
- [7] Vlachos, M., Lin, J. , Keogh, E., Gunopulos, D., *A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time-Series*, in Proceedings Workshop on Clustering High-Dimensionality Data and its Applications, SIAM International Conference on Data Mining (2003), pp. 23–30.
- [8] Lin, J., Keogh, E., Lonardi, S., Lankford, J., and Nystrom, D., *VizTree: a tool for visually mining and monitoring massive time series databases*, in Proceedings of the Thirtieth international Conference on Very Large Data Bases – Volume 30 (2004), 1269-1272.

- [9] Metwally, A., Agrawal, D., and Abbadi, A., *Efficient Computation of Frequent and Top-k Elements in Data Streams*, in Proceedings of the 10th International Conference on Database Theory (2005), pp. 398–412.
- [10] Tanaka, Y., Iwamoto, K., and Uehara, K. 2005. *Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle*, in Machine Learning 58, (2005), pp. 269–300.
- [11] Azevedo, P., Silva, C., Rodrigues, R., Ferreira, N., Brito, R., *Detection of Hydrophobic Clusters in Molecular Dynamics Protein Unfolding Simulations using Association Rules*, in Proceedings of the 6th International Symposium on Biological and Medical Data Analysis (2005), pp. 329–337.
- [12] Ratanamahatana, C., Keogh, E., *Three Myths about Dynamic Time Warping Data Mining*, in the Proceedings of SIAM International Conference on Data Mining (2005), pp. 506–510.
- [13] P. Ferreira, P. Azevedo, C. Silva, and R. Brito, *Mining approximate motifs in time series*, in Proceedings of the 9th International Conference on Discovery Science (2006), pp. 89–101.
- [14] Ferreira, P., Azevedo, P., *Evaluating Protein Motif Significance Measures: A case study on Prosite Patterns*, in Proceedings of IEEE Symposium on Computational Intelligence and Data Mining (2007), pp. 171–178.
- [15] Lin, J., Keogh, E., Li, W., Lonardi, S. *Experiencing SAX: A Novel Symbolic Representation of Time Series*. in Data Mining and Knowledge Discovery Journal (2007). pp. 107–144.
- [16] D. Minnen, C. Isbell, I. Essa, and T. Starner, *Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery*, Seventh IEEE International Conference on Data Mining (2007), pp 601–606.
- [17] Yankov, D, Keogh, E., Medina, J., Chiu, B., and Zordan, V., *Detecting Motifs Under Uniform Scaling*, in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007), pp. 844–853.
- [18] Yankov, D., Keogh, E., Rebbapragada, U., *Disk aware discord discovery: finding unusual time series in terabyte sized datasets*, in Proceedings of the 7th IEEE International Conference on Data Mining (2007), pp. 381–390.
- [19] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. ,*Querying and mining of time series data: experimental comparison of representations and distance measures*, in Proceedings of the VLDB Endowment (2008), pp. 1542–1552.
- [20] Shieh, J. and Keogh, E. 2008 *iSAX: indexing and mining terabyte sized time series*, in Proceedings of the 14th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (2008), pp. 623–631.
- [21] Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B., *Exact Discovery of Time Series Motifs*, in the Proceedings of SIAM International Conference on Data Mining (2009), pp. 473–484.
- [22] Gama, O., Carvalho, P., Afonso, J., Mendes, P., *An improved MAC protocol with a reconfiguration scheme for wireless e-health systems requiring quality of service*, in First International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (2009), pp. 582–586.
- [23] Castro, N., *Multiresolution Motif Discovery in Time Series website* , <http://www.di.uminho.pt/~castro/mrmotif>.