Mining Approximate Motifs in Time Series

Pedro G. Ferreira,[1] Paulo J. Azevedo[1], Cândida G. Silva[2], Rui M.M. Brito[2] ⋆

[1] Department of Informatics, University of Minho 4710-057 Braga, Portugal
{pedrogabriel,pja}@di.uminho.pt
[2] Chemistry Department, Faculty of Sciences and Technology, and Centre of Neurosciences of Coimbra, University of Coimbra, 3004-517 Coimbra, Portugal
csilva@student.uc.pt, brito@ci.uc.pt

Abstract. The problem of discovering previously unknown frequent patterns in time series, also called motifs, has been recently introduced. A motif is a subseries pattern that appears a significant number of times. Results demonstrate that motifs may provide valuable insights about the data and have a wide range of applications in data mining tasks. The main motivation for this study was the need to mine time series data from protein folding/unfolding simulations. We propose an algorithm that extracts approximate motifs, i.e. motifs that capture portions of time series with a similar and eventually symmetric behavior. Preliminary results on the analysis of protein unfolding data support this proposal as a valuable tool. Additional experiments demonstrate that the application of utility of our algorithm is not limited to this particular problem. Rather it can be an interesting tool to be applied in many real world problems.

1 Introduction

The mining of time series data has gathered a great deal of attention from the research community in the last 15 years. These studies have impact in many fields, ranging from biology, physics, astronomy, medical, financial and stock market analysis, among others. The research in mining time series has been mainly focused in four problems [9]: *indexing* or *query by content*, *clustering*, *classification* and *segmentation*. Lately, the problem of mining unusual and surprising patterns [10] has also been enthusiastically studied. Other challenging and recently proposed problem in the context of time series is the mining of previously unknown patterns. These patterns, here referred as *episodes*, consist of subsequences that appear, in a unique and longer sequence [15, 7, 5] or are subsequences that occur simultaneously in more than one sequence from a set of related sequences, in this case called *motifs* or *sequence patterns*. These motifs have a wide range of applications. They can be used in the clustering and classification of time series. They can also be applied in the generation of sequence rules and in the detection

⋆ The authors acknowledge the support of the "Fundação para a Ciência e Tecnologia" and the program FEDER, Portugal, POCTI/BME/49583/2002 (to RMMB) and the Fellowships SFRH/BD/13462/2003 (to PGF) and SFRH/BD/16888/2004 (to CGS).

of interesting behaviors, which can give the user/domain expert valuable insights about the problem that is being studied.

In this work we are interested in the extraction of time series motifs. We present an algorithm that given as input the symbolic representation of a set of comparable time sequences, it finds all the patterns that occur a number of times equal or greater than a threshold value. A sequence pattern or motif consists in a set of subsequences that share among them a similarity greater than an user defined value. The definitions adopted in this work and the development of the algorithm were mainly motivated by the specificities of the mining of time series data from protein folding/unfolding simulations, as we will discuss in section 4. However, as we will see in the same section, these ideas can also be applied to many different domains and application contexts.

2 Definitions and Notations

In this section we will present some definitions that will be used throughout this paper.

Definition 1. (Time Sequence) *A time sequence T is an ordered set of values $(t_1, t_2, \ldots, t_n)$, where $t_i \in \mathbb{R}$; $(t_p, t_{p+1}, \ldots, t_q)$ is a subsequence of T starting at position p and ending at position q, where $1 \leqslant p$ and $q \leqslant n$. The length of sequence, $|T|$ is equal to n.*

In this work we are interested in finding patterns over a set of temporal sequences that for a certain period of time reflect a similar and/or a symmetric tendency. This trend or tendency reflects a measure of interest of the motif and is called here *approximate similarity.*

According to our notion of approximate similarity, several measures and coefficients appear as candidates for similarity functions. The most popular measures of distance appearing in literature are the Euclidean distance and the Dynamic Time Warping (DTW) measure [6, 11]. For this particular problem, the drawback of these two measures is that they are not sensitive to the association linearity between the elements of the subsequences. We are interested in finding patterns based on an approximate similarity, thus a more suitable measure is required.

The Pearson Correlation Coefficient [20], r, measures the magnitude and the direction of the association between the values of the subsequences. The correlation coefficient ranges from -1 to +1 and reflects the linear relation between the values of the subsequences. The Pearson correlation is a metric measure that satisfies the three following properties: *positivity*$(r(x,y) \geq 0$ and $r(x,y) = 0$, if $x = y)$, *symmetry* $(r(x,y) = r(y,x))$ and *triangle inequality* $(r(x,z) \leq r(x,y) + r(y,z)))$. This last property as we will see, will be particularly useful to cluster pairs of similar subsequences to form the motifs. In fact, Pearson correlation tends to be robust to small variations.

Definition 2. (Match) *Given the similarity function between two subsequences X and Y, $sim(X,Y)$, we say that subsequence X matches Y if $|sim(X,Y)| \geqslant R$, where R is a user supplied positive real number.*

The absolute similarity value used in definition 2 handles the notion of approximate similarity, i.e. two subsequences may have an inverted behavior and still be considered a match.

Definition 3. (Instance) *A subsequence X is an instance of a subsequence Y, where $|X| = |Y|$, if $|sim(X,Y)| \geqslant R$.*

Definition 4. (Overall Similarity) *The overall similarity for a set of subsequences corresponds to the average value of similarity between all the pairs of subsequences in the set.*

Therefore, a motif can informally be defined as a set of interesting subsequences. The interestingness is defined by its overall similarity and by the frequency of appearance, i.e. how much recurrent are the subsequences in the set of the input time sequences. The frequency is provided by the cardinality of the set and is usually called as *support*. Hence, formally a motif can be defined as follows:

Definition 5. (Approximate Motif) *Given a database D of time sequences, a minimum support σ and a minimum value of similarity/correlation R_{min}. We consider that k subsequences consist of an approximate motif, if $k \geq \sigma$ and all subsequences pairwisely match for a value of R_{min}.*

Definition 6. (K-Cluster) *We denote a group of k (related) instances as k-cluster.*

Definition 7. (Cluster Containment) *A cluster C_1 is contained in a cluster C_2, if all instances of C_1 are in C_2.*

Definition 8. (Overlap Degree) *The degree of overlap between two sequences X and Y is defined as $Od = \frac{|X \cap Y|}{w} \times 100\%$, where $\cap$ is the intersection operation and w the length of the sequences. $|X \cap Y|$ gives the overlap region between X and Y.*

3 Algorithm

In this section we start by formulating the described problem. Next, we give an overview of the algorithm that we propose to tackle this problem.

Problem Formulation: Given a database D of time sequences, a minimum support σ, a minimum similarity R_{min}, a window length w and a window frame length deltaW [3] *find all the approximate motifs.*

Typically, before the algorithm is applied, a pre-processing step is performed. We adopted a two step approach called SAX [14]. In the first step, a complexity/dimensionality reduction on the data may be necessary and desirable. This

[3] Concept introduced in subsection 3.1.

reduction can be achieved through means of a *scaling* operation, also called PAA (Piecewise Aggregate Approximation) [16]. This operation basically consists in a reduction on the number of intervals in the time axis. A group of successive points is replaced by their average value. The second part of pre-processing step is the *symbolization*. The amplitude axis is scaled and divided into a finite number of intervals of equal size. Each interval is represented by a symbol. Sequence values are them mapped into the respective symbol of belonging interval. In our particular case all the sequence values are mapped into integer values (called *alphabet*, and denote it as Σ), therefore a symbolic representation of the time sequences is used. This approach brings two benefits to the data analysis, which are the robustness to small variations and to noise [14]. Both scaling and symbolization are performed in linear time. We continue by outlining the proposed algorithm.

3.1 Phase One: Motif Detection

This phase of the algorithm is divided into two steps. In step 1, all the subsequences of length w in D are scanned and compared against each other, in order to find similarities among them. If two subsequences match, they form a 2-cluster. In the second step of this phase, the pairs of subsequences are successively clustered with the goal of finding longer clusters of subsequences. If the overall similarity is above the minimum similarity threshold R_{min}, i.e. all instances pairwisely match, and its cardinality is greater than the minimum support σ, the cluster is then considered a motif. Our technique is a bottom-up or agglomerative method and it resembles the hierarchical agglomerative clustering technique. We start with all the clusters of size two and we keep merging the most similar ones until we obtain one cluster that can no longer be extended. If the cluster satisfies the motif definition 5, then it is considered a motif.

```
    input  : $w, R_{min}$
    output: $ClusterInfo$ : $List\ with\ the\ 2clusters$
 1  $cnt = 0$;
 2  foreach $S$ in $D$ do
 3      for $i = 0$ to $|S| - w + 1$ do
 4          $ss = S(i, i + w)$;
 5          $lstFSS = findFollowSS(D, ss)$;
 6          foreach $fss$ in $lstFSS$ do
 7              if $|sim(fss, ss)| \geq R_{min}$ then
 8                  $clusterInfo[cnt] = \{ss, fss\}$;
 9                  $cnt = cnt + 1$;
10              end
11          end
12      end
13  end
```

Algorithm 1: *2clustersEnumeration* function.

Step 1: 2-Cluster Enumeration The pseudo-code in algorithm 1 shows the application of the sliding window methodology. Each subsequence is defined by the tuple $< seq, start >$, that represents the sequence identifier and the start where the subsequence occurs in D. It scans all the sequences in the database (line 2) and for each sequence it also scans all its subsequences of size w (line 3 and 4). For each subsequence (line 5) the respective following subsequences are obtained. If any of these subsequences match ss, the information for this pair of subsequences is saved (line 8) in the clusterInfo list. Function findFollowSS retrieves all the subsequent subsequences of ss. It basically consists in two for loops that scan all the windows that occur after ss in D. At this point two scenarios are possible. The algorithm may look for all the occurrences in D of the motif (one sequence may contribute with more than one instance for the counting) or only for the occurrences in different sequences. In the first case, the function starts to scan D in the sequence $ss.seq$ at position $ss.start + 1$. In the second case, it starts in the sequence $ss.seq + 1$ at position 0. This last case eliminates the existence of trivial matches, i.e. matches between subsequences that are apart from each other only few points. In this work we are only interested in the second case.

In the motif discovery process, we have decided for an agglomerative approach to cluster all the instances of a motif. Another possibility would be to use a subsequence oriented approach, where for each scanned subsequence (reference) and respective matches, if the motif definition is verified it would be immediately reported as a motif. Although simpler, this approach has two problems. First too many repeated motifs would be reported. For example, for a motif with instances {A,B,C} the motifs {A,C,B}, {C,B,A} and so on would be reported. The second problem is that when some instances match with the reference subsequences but not with the other matches of this subsequence all the possible combinations have to be tried. Consider the example in figure 1(a). Instance A (reference) matches B,C,D but instance B does not match C and D. Thus, we have at least two clusters {A,B} and {A,C,D}. When the number of non-matching instances is greater, the number of possible clusters combinations increases, which become more difficult to manage. The agglomerative approach prevents these two problems by avoiding most of the motif redundancy and making a clear separation of the clusters.

Concerning the enumeration of 2-clusters, its dispersion along the time line arises as an interesting question. Suppose a motif that has an instance that occurs in the initial part of a time sequence and other in the end of other sequence. A significant distance between the instances may invalidate the meaning of the motif. Therefore, we incorporate the option to enumerate motif instances that occur only within a certain window frame.

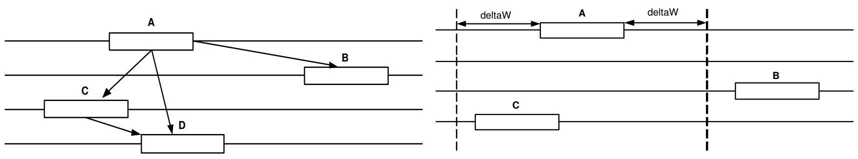

Fig. 1. (a) Example where a subsequence has several matches in more than one cluster; (b) Example of the application of a window frame to the enumeration of motif instances.

Figure 1 (b) shows an example of the application of the window frame of size $deltaW$. Although, instances A, B, C may have an overall similarity greater than R_{min}, instance B is not within the frame $[A.start - deltaW; A.end + deltaW]$. Thus, B should not be considered as a motif instance. Note, that we could ignore this window frame option and still consider the sequences in all their extension. This feature can be easily handled by introducing in line 7 of algorithm 1, the condition $|ss.start - fss.start| < deltaW$ and in line 8 of algorithm 3 the condition $\forall i, j \; |ss_i.start - ss_j.start| < deltaW$.

One problem that typically arises in heuristic methods that find probabilistic motifs in biological data like in [18, 2], is the "phase" problem. This means that when looking for a pattern, the algorithm may get locked in a local optimum and not in the global optimum of the pattern. For example, consider that the best solution of a pattern starts at position 7, 19, 13, ... within the given sequences. If after some iterations the algorithm chooses positions 9, 21 in the first two sequences it probably will consider the position 15 in the third sequence and so on. It means that the algorithm got trapped in a local optimum and the obtained solution is shifted by a few positions from the correct solution. The sliding-window based methods may also suffer from this problem, as is the case of our algorithm. In order to solve it we analyze the 2-cluster list, where each cluster is compared to all the other 2-clusters that are within a certain neighborhood range. Only the 2-cluster which maximize the similarity value is kept. We consider that two sequences can be considered different when they differ in more than 1/3 of its length. Thus, two 2-cluster are in the same neighborhood if the overlap degree between the respective subsequences is greater than 2/3 of their length.

Step 2: Cluster Extension At the end of step 1 we obtain a list of 2-clusters that will form the basis of the potential motifs. In an agglomerative way the similar pairs of subsequences (2-clusters) can be merged, until one big cluster is found. The criteria used to merge one 2-cluster into a N-cluster ($N \geq 2$) is based on the triangle inequality property of our similarity measure. The idea is that if instance x is similar to y and, y is similar to z, then it is expected that x is also similar to z. Thus, the cluster $\{y, z\}$ is merged with the cluster $\{x, y\}$ when all the pairs of instances from $\{x, y, z\}$ match for a value of R_{min}. In fact, the triangle inequality property allows us to define a linkage method to merge the clusters. The extension of a cluster C stops when no more 2-clusters exist to merge or when a in the result of a cluster extension a pair of instances does not match. Function *seedExtension* summarizes the described ideas.

```
   input  : $ClusterInfo, R_{min}, \sigma$
   output: $motifList$ : $List\ with\ motifs$
1  for $i = 0$ to $clusterInfo.size - 1$ do
2      for $j = i + 1$ to $clusterInfo.size$ do
3          $status = extendCluster(clusterInfo[i], clusterInfo[j], j, R_{min}, \sigma)$;
4          if $(\sigma == 2) AND (status == NotExtended)$ then
5              motifList.add(clusterInfo[i]);
6          end
7      end
8  end
```

Algorithm 2: *seedExtension* function

In algorithm 2 the list of 2-clusters (clusterInfo) is traversed in order to try the extension of one 2-cluster with another 2-cluster. If the cluster can not be extended and the minimum support is 2, then it can be considered as a motif (line 5). In order to avoid redundant motifs (motifs contained in other motifs) we only consider a cluster to be a motif when it can no longer be extended.

```
    input  : $ClusterInfo, clusterExt, clusterPair, index, R_{min}, \sigma$
    output: $motifList$ : $List\ with\ motifs$
 1  if ($index == clusterInfo.size$) then
 2      return NotExtended;
 3  end
    /* Intersect the subsequences of the clusters                    */
 4  $IS = intersect(clusterExt, clusterPair)$;
 5  if ($index == \emptyset$) then  return NotExtended;
 6  else  $clusterExt = join(clusterExt, clusterPair)$;
    /* Find the new similarity value                                  */
 7  $newSimil = avgSimil(clusterExt)$;
 8  if $|newSimil| \geq R_{min}$ then
 9      for $j = index + 1$ to $clusterInfo.size$ do
10          $status = extendCluster(clusterExt, clusterInfo[j], i, R_{min}, \sigma)$;
11          if ($clusterExt.size \geq \sigma$)$AND$($status == NotExtended$) then  motifList.add(clusterInfo[i]);
12          return NotExtended;
13          else  return Extended;
14      end
15  end
```

Algorithm 3: *clusterExtension* function.

In procedure *clusterExtension*, the input consists of a cluster to be extended, clusterExt, and a cluster which is used to try an extension, clusterPair. If one subsequence in clusterPair is present in clusterExt (verified with function intersect in line 4), clusters are joined (line 6) and an extension is tried (lines 7 to 15). If all pairs of instances match, the new similarity value of the cluster is above R_{min}, the cluster can be extended. If its cardinality is equal or greater than σ and it is the last possible extension then it is considered a motif (lines 11 to 12). Otherwise, no extension is performed and the status *NotExtended* is returned. Note, that since this procedure is applied recursively, line 1 is used to test if no more 2-clusters exists when trying a new extension. At the end of this step the algorithm already retrieved the list of all possible motifs in D. Eventually, due to the extension process, some of these motifs are non-maximal, i.e. they are contained in other motifs. In order to eliminate this redundancy we apply a procedure that removes non-maximal motifs from this list. This is a time-consuming operation since each motif in the list has to verified whether it contains or is contained in another motif.

3.2 Phase2: Length Extension

In phase 2, a length extension of all the motifs extracted in phase 1 is tried. Since the subsequences of a motif are still a motif, we need to find its longest length. Thus, for a given motif all its instances are extended until the overall similarity drops below R_{min}. When the length extension is performed, a previous test has to be done since it is not know in advance the extension direction. For each pair of instances in the motif, all the four possible direction combinations are tried. The information, namely the extension direction and gainOfExtension = newValue - oldValue of each instance, is saved when the tested extension maximizes the gain. When all the pairs of subsequences are verified, a list with the directions of the extension of each subsequence is obtained. Next, based on these directions, an extension is performed one event at a time, until the overall similarity of the motif drops below R_{min}.

3.3 Motif Features and Statistical Significance

Two types of patterns can be distinguished. Motifs that contain only positive correlations and mixed motifs that contain both positive and negative correlations. If a sequence Y regresses on X, the equation $Y \simeq \beta_0 + \beta_1 X + u$, where u is the model error, may model such regression. In order to provide the scale in which all the instances of the motifs are related we calculate the average value of β_1 (refer to [20] for this calculation).

To assess the statistical significance of a motif, two measures are provided for each motif: Information Gain [19] and Log-Odds significance [12]. The information content I of a pattern, measures how likely a pattern is to occur, or equivalently, what is the amount of "surprise" when the pattern occurs. The odds score of a sequence measures the degree of surprise of the motif by comparing its probability of occurrence with the expected probability of occurrence according to the background distribution.

4 Experimental Evaluation

In our experiments we used a prototype developed in the C++ language. All the tests were done in 3.0GHz Pentium4 machine with 1GB of main memory, running windows XP Professional. Our algorithm works in "in-memory" way since it first loads to main memory the entire dataset and then starts the mining. Nevertheless, in all the experiments the maximum memory usage was 20MB. Each motif graphic contain the correspondent Overall Similarity (S), average β_1(B) and the type correlation of the pattern (T): only positive (1) and both (0).

4.1 Protein Unfolding

Protein folding is the process of acquisition of the native three-dimensional structure of a protein. The 3D structure, ultimately determined by its linear sequence of amino-acids, is essential for protein function. In recent years, several human and animal pathologies, such as cystic fibrosis, Alzheimer's and mad cow disease, among others, have been identified as protein folding or protein unfolding disorders. Over the years, many experimental and computational approaches have been used to study protein folding and protein unfolding. Here we use the proposed algorithm to assist in the study of the unfolding mechanisms of Transthyretin (TTR), a human plasma protein involved in amyloid diseases such as Familial Amyloid Polyneuropathy. Our goal is to find approximate motifs, i.e. simultaneous events on variations of molecular properties characterizing the unfolding process of TTR. The data analyzed is constituted by changes observed in molecular properties calculated from Molecular Dynamics (MD) protein unfolding simulations of TTR [1, 3]. In the present case, the dataset consists of 127 time series, each representing the variation over time of the Solvent Accessible Surface Area (SASA) of each amino acid in the protein. Each time series is a collection of 8000 data points, one data point per picosecond (ps) for a total of 8 nanoseconds (ns) of simulation. Before the algorithm is applied, pre-processing of the data is performed. The 8000 data points were scaled to 160 intervals. Each

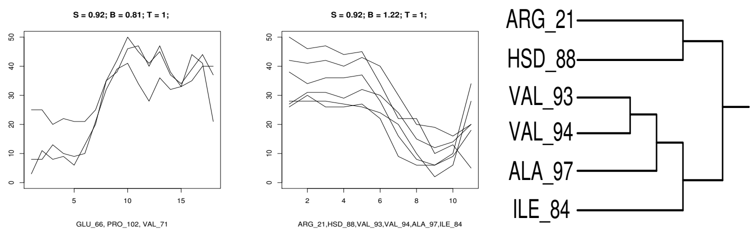

Fig. 2. (a) Example of a motif with an increasing SASA; (b) Example of a motif with a decreasing SASA and (c) respective cluster of Amino-Acids.

interval represents the average variation of SASA over 50ps. Symbolization was performed by rounding each value to its closest integer.

In the unfolding process of a protein, it is expected that the SASA increases for most amino-acids, i.e. they tend to become more exposed to the solvent upon protein unfolding. However, identifying how and when the SASA increases, and which amino-acids have similar (positive correlation) or opposite behavior (negative correlation), may reveal important details of the unfolding mechanism and which amino-acids constitute structural intermediates essential in the unfolding process. Figure 2 (a) depicts a motif that represents a synchronized increase of the SASA values for three amino-acids. Around data point 5, an unidentified event triggered the increase of solvent exposure. This is an example of a motif that is worthwhile to be investigated. Figure 2 (b) shows a motif with an overall tendency of SASA decreasing. Since this motif opposes to the expected behavior, it is also interesting to be further investigated. The motifs in figure 2 were obtained respectively with the parameters $\sigma = 3$, $w = 15(750ps)$ and $\sigma = 4$, $w = 10(500ps)$, with $deltaW = 5(250ps)$.

4.2 Stock Market Analysis and Synthetic Control Charts

The use of stock market datasets for the evaluation of time series algorithms is almost classic. We analyzed the Standard and Poor (S&P) 500 index historical stock data to demonstrate another possible application domain of our algorithm. This dataset contains 515 stocks with daily quotes fluctuations recorded over the course of the year 2005. It was obtained from http://kumo.swcp.com/stocks. We have analyzed the volume data for each day. The data contained a variable length (between 50 and 252) of points, since no scaling was done in this case. The size of the symbols alphabet was 100. We have made several runs on data, and we choose the results obtained with the following parameters: $\sigma = 0.01\%$; $R_{min} = 0.95$; w = 15 and deltaW = 5. It resulted in 9 motifs. From these we choose the 4 motifs with the highest statistical significance, according to the LogOdds and Information Gain measures. For each motif in figure 3 (upper row) it is presented the identifier of the company where the motif occurs and the respective statistical scores. Note that motif (d) has a overall similarity below R_{min}. This happen since this is a motif with two types of correlation (T=0).

In order to test our algorithm in a set of time series with well differentiated characteristics we applied it to the SCC dataset. This dataset was obtained from the UCI repository [8] and contain 600 examples of control charts synthetically generated, divided in six different classes of control charts. We mined each class individually and according to the parameters: $\sigma = 0.05\%$, $R_{min} = 0.9$, w=15, $deltaW = 15$. The average runtime was 3.67 secs and average number of motifs: 12.8. Figure 3 (lower row) shows examples of motifs with the highest similarity value for each of the four classes.

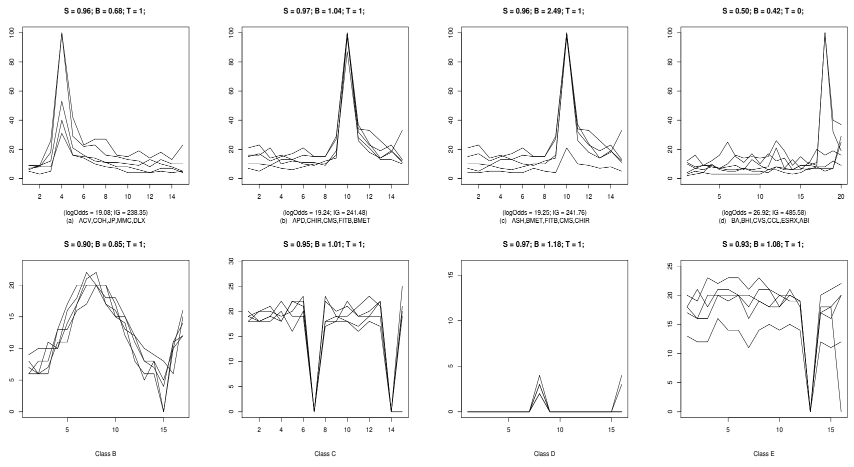

Fig. 3. (Upper row) Example motifs from the SP500 dataset; (Lower row) Selected motifs from the synthetic control dataset from class B, C, D, E.

4.3 Performance Evaluation

Due to the lack of larger datasets we used synthetic random walk sequences to evaluate the performance of our algorithm in several dimensions. The formula used to generate the points of the sequences was: $S(j+1) = S(j) + RandNum(10)$.

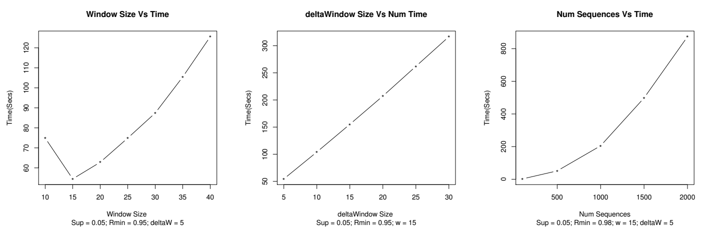

Fig. 4. Performance of the algorithm with different databases and parameter settings: Runtime w.r.t: (a) window size;(b) deltaW size;(c) number of sequences.

Due to its different phases and steps our algorithm has also different levels of complexity. Step 1 of Phase 1 is quadratic with respect to the length of the sequences. For N possible windows in database, this step requires approximately N^2 calls to the similarity function. Depending on the size of the window frame the number of tested pairs of windows can be greatly reduced with a consequently diminution in the runtime (see figure 4(b)). The complexity of Step 2 depends on the output of Step 1, namely it is proportional to number of 2-Clusters. This number directly depends on the R_{min} and on w values (see figure 4(a)). The initial value in the (see figure 4(a)) is explained by the significative difference in the number of 2-clusters obtained with $w = 10$ and with $w = 15$. The Phase 2, which corresponds to the length extension of the motifs is the less demanding. It typically represents an insignificant part of the overall time. Finally, we give some insights about the algorithm scalability. Figure 4(c) shows that our algorithm scales nearly in quadratic time in relation to the database size, which in practice is a is a reasonable performance for small or medium size datasets.

As it is expected in this type of mining applications, the interest of the results is a direct outcome of parameters values. Therefore, an iterative and interactive application of the algorithm is necessary. Heuristic or statistical methods, like the MDL principle used in [17], may provide initial values for the parameter setting.

5 Related Work

Our work can be viewed as a fusion of two research areas of data mining. The sequence and motif mining area where we emphasize the motivation provided by the algorithms in bioinformatics for mining overrepresented patters in a set of related and comparable biological sequences (proteins/dna) [18, 2]. A second area corresponds to all the research made in time series, where particular attention is given to recent advances in the algorithms for pattern extraction. In the context of time series, we start by emphasizing some earlier work that have been done in the mining of recurrent subseries/patterns throughout a particular sequence [4, 7]. Recently, Keogh et al. has introduced the issue of mining motifs in time series [16] and proposed algorithms [16, 5] for this task. In [5] is described a probabilistic algorithm inspired on another algorithm from bioinformatics, called random projection. The difference from our work to the previous work starts with the definition of a motif. We are interested in discovering groups of subsequences that reveal the same trend, possibly in a symmetric way (approximate motifs). Additionally, we search for motifs in a set of related time sequences, eventually confined to a certain window frame, and not only in one time sequence. One work that has also inspired us is [13]. Here, the Pearson's correlation is used in the context of linear regression analysis in order to cluster time series.

6 Conclusions

In this work we have formulated the problem of mining approximate motifs in a set of related sequences. We also propose an algorithm that allows discovering all the motifs in the database, thus ensuring no false dismissals. The application of the algorithm to the problem that first motivated its development has already proved to be a valuable tool to assist biologists. However, the application of this method is not limited to this case study. As we have demonstrated it is an interesting method to be applied in other application domains.

References

1. P. Azevedo, C. Silva, J. Rodrigues., N. Loureiro-Ferreira, and R. Brito. Detection of hydrophobic clusters in molecular dynamics protein unfolding simulations using association rules. *LNCS*, (3745), 2005.
2. T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proc. of the 2th ISMB*, 1994.
3. R. Brito, W. Dubitzky, and J. Rodrigues. Protein folding and unfolding simulations: A new challenge for data mining. *OMICS: A Journal of Integrative Biology*, (8):153–166, 2004.
4. J. Caraca-Valente and I.Lopez-Chavarrias. Discovering similar patterns in time series. In *Proc. of the 6th ACM SIGKDD*, 2000.
5. B. Chiu, E. Keogh, and S. Lonardi. Probablistic discovery of time series motifs. In *Proceedings of the 9th ACM SIGKDD*, Washington DC, USA, August 24-27 2003.
6. D. Gunopulos and G. Das. Time series similarity measures (tutorial pm-2). In *Tutorial notes of the 6th ACM SIGKDD*, 2000.
7. J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proc. of the 15th ICDE*, 1999.
8. S. Hettich and S. D. Bay. The uci kdd archive irvine,[http://kdd.ics.uci.edu], ca: University of california, department of information and computer science, 1999.
9. E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empricial demonstration. In *Proc. of the 8th ACM SIGKDD*, 2002.
10. E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proc. of the 5th IEEE ICDM*, 2005.
11. E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. of the 6th ACM SIGKDD*, 2000.
12. Anders Krogh. *An Introduction to Hidden Markov Models for Biological Sequences*, chapter 4, pages 45–63. Elsevier, 1998.
13. H. Lei and V. Govindaraju. Grm: A new model for clustering linear sequences. In *Proc. of SIAM Int'l Conference on Data Mining*, 2004.
14. J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. of the 8th ACM SIGMOD workshop DMKD'03*, 2003.
15. H. Mannila, H. Toivonen, and A. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
16. P. Patel, E.Keogh, J.Lin, and S.Lonardi. Mining motifs in massive time series databases. In *Proc. of 2th IEEE ICDM*, December 2002.
17. Y. Tanaka and K. Uehara. Discover motifs in multi-dimensional time-series using the principal component analysis and the mdl principle. In *Proc. of 3th MLDM*, 2003.
18. W. Thompson, E. Rouchka, and C. Lawrence. Gibbs recursive sampler: finding transcription factor binding sites. *Nucleic Acids Research*, 31(13):3580–3585, 2003.
19. J. Yang and P.S. Yu W. Wang. Mining surprising periodic patterns. In *Proc. of the 7th ACM SIGKDD*, 2001.
20. J.H. Zar. *Biostatistical Analysis (4th Edition).* Prentice Hall, 1998.