Approximately Recurring Motif Discovery Using Shift Density Estimation

Yasser Mohammad[1,2] and Toyoaki Nishida[2]

[1] Assiut University, Egypt
[2] Kyoto University, Japan
yasserm@aun.edu.eg, nishida@i.kyoto-u.ac.jp

Abstract. Approximately Recurring Motif (ARM) discovery is the problem of finding unknown patterns that appear frequently in real valued timeseries. In this paper, we propose a novel algorithm for solving this problem that can achieve performance comparable with the most accurate algorithms to solve this problem with a speed comparable to the fastest ones. The main idea behind the proposed algorithm is to convert the problem of ARM discovery into a density estimation problem in the single dimensionality shift-space (rather than in the original time-series space). This makes the algorithm more robust to short noise bursts that can dramatically affect the performance of most available algorithms. The paper also reports the results of applying the proposed algorithm to synthetic and real-world datasets.

1 Introduction

Consider a robot watching a human communicating with another using free hand gestures to achieve some task [13]. The ability to automatically discover recurring motion patterns allows the robot to learn important gestures related to this domain. Consider an infant listening to the speech around it. The ability to discover recurring speech patterns (words) can be of great value in learning the vocabulary of language. In both of these cases, and in uncountable others, the patterns do not recur exactly in the perceptual space of the learner. These cases motivate our search for an unsupervised algorithm that can discover these kinds of approximately recurring motifs (ARMs) in general time-series. Several algorithms have been proposed for solving this problem [10] [7] [2],[15],[4] [6], [17].

In this paper we propose a novel algorithm for solving ARM discovery directly. The proposed algorithm achieves high specificity in discovered ARMs and high correct discovery rate and its time and space complexities can be adjusted as needed by the application. The main insight of the proposed algorithm is to convert the problem of subsequence density estimation which is multidimensional in nature into a more manageable single dimensional shift density estimation. This allows the algorithm to discover complete ARMs (with potential don't-care sections). The paper also reports a quantitative 6-dimensions evaluation criteria for comparing ARM discovery algorithms.

2 Problem Statement and Related Work

A time series $x(t)$ is an ordered set of T real values. A subsequence $x_{i,j} = [x(i) : x(j)]$ is a contiguous part of a time series x. Given two subsequences $\alpha_{i,j}$ and $\beta_{k,l}$, a distance function $D(.,.)$ and a positive real value R, we say the two subsequences *match up to R* if and only if $D(x_{i,j}, x_{k,l}) < R$. We call R the range following [16]. In this paper, we assume that the distance function $D(.,.)$ is normalized by length of its inputs. Moreover, our algorithm will only apply D to pairs of subsequences of the same length. In most cases, the distance between overlapping subsequences is considered to be infinitely high to avoid assuming that two sequences are matching just because they are shifted versions of each other (these are called trivial motifs [5]).

An approximately recurrent motif (ARM) or motif for short is a set of subsequences that are similar in some sense. In most cases similarity between subsequences is measured as the inverse of their distance [14]. Either the Euclidean distance or dynamic time wrapping (DTW) could be used for this calculation. Relying on these distance functions in the definition of a motif implies that a predefined motif length must be given to the algorithm. Several algorithms were suggested to discover distance based motifs [10] [7] [2] [15] [4] [6] [8] [17]. Many of these algorithms are based on the PROJECTIONS algorithm proposed in [18] which uses hashing of random projections to approximate the problem of comparing all pairwise distances between n subsequences to achieve linear rather than quadratic space and time complexities. Because this algorithm works only with discrete spaces, the time series must be discretized before applying any of PROJECTIONS variants to it. A common discretization algorithm employed for this purpose is SAX [6]. An unsupervised method for finding a sensible range parameter for these algorithms was proposed in [7]. The proposed algorithm differs from all of these approaches (even with automatic range estimation) in requiring no discretization step and being able to discover motifs in a range of lengths rather than a single length. The proposed algorithm also has adjustable space and time complexity and is linear in the worst case, while all PROJECTIONS algorithms require good selection of the discretization process parameters to lead to sparse collision matrices in order to avoid being quadratic.

Another approach for finding these motifs was proposed in [1] that uses random sampling from the time series (without any disretization). This algorithm requires an upper limit on the motif length and also is not guaranteed to discover any motifs or to discover them in order. An explicit assumption of this algorithm is that the motifs are frequent enough that random sampling will has a high probability of sampling two complete occurrences in candidate and comparison windows of lengths just above the maximum motif length. The sampling process was improved in MCFull [10] by utilizing a change point discovery algorithm to guide the sampling process with reported significant increase in discovery rate. Even though no clear definition of what is actually discovered by these algorithms (other than being frequent), they actually discover ARMs. The proposed algorithm has higher discovery rate than MCFull with comparable speed as will be shown in section 4.

The MK algorithm for discovering exact motifs was proposed in [14] and it is the most cited motif discovery algorithm since its appearance. In MK, the Euclidean distance between pairs of subsequences of length l is used to rank motif candidates. This has the problem of requiring a predefined motif length (while our approach requires only a motif length range). It is also sensitive to short bursts of noise that can affect the distance. The main difference between the proposed approach and this algorithm is that we rely on multiple distance estimations between short subsequences rather than a single distance calculation of the predefined motif length. This has three major advantages: First we need not specify a specific length. Secondly, the distance function is not required to be a metric (i.e. it is not required to satisfy the triangular inequality). Finally, the proposed algorithm can *ignore* short bursts of noise inside the subsequences (because of its multi-distance calculations) which is not possible if MK is directly used. Nevertheless, the MK algorithm can be used to speedup finding best matches during shift-density estimation. This was not tried in this work but will be compared with the current implementation in future work. Hereafter, we will use the word *motif* and ARM interchangeably as long as the context is clear.

3 Proposed Algorithm

The algorithm uses three types of windows. The candidate window is a subsequence s that is being considered for similar subsequences in the time series. The candidate window should be wide enough to contain a complete occurrence of any ARM to be discovered. The length of this window is called w. The random window is a time series of the same length as the candidate window and is constructed by randomly selecting w values from the time-series. This window is used by the algorithm to discover an upper limit of distances that can be considered *small* during processing. The idea of using a random window for this purpose can be found in [1]. The third type of windows is the comparison window. Comparison windows are subsequences of w points that are to be compared to the current candidate window in search for ARM occurrences within them.

The algorithm proceeds in three major steps: firstly, candidate locations of ARM occurrences are discovered using a change point discovery algorithm [11] and candidate windows are sampled around these points. This set is called $\mathtt{C}$ hereafter. Secondly, each one of these windows is compared with the rest of them (acting as comparison windows) and best matching windows are found as well as the best time-shifts in the best comparison windows to get it to best match the candidate window. This is the core step of the algorithm and is the point at which shift-density estimation is carried-out. Finally, the shifts required are analyzed in order to remove partial ARM occurrences, multiple ARM occurrences, and out-of-ARM parts of the candidate window and a new ARM is announced if a long enough occurrence could be found at that stage. The following subsections present the final two stages. For more details about the first stage please refer to [10].

3.1 Finding Best Matches

This step is the core of the proposed algorithm. Each member of $\mathtt{C}$ is treated as a candidate window while the ones after it are treated as a comparison window set until all members of $\mathtt{C}$ are considered. The current candidate window (c) is divided into $w - \bar{w}$ ordered overlapping subwindows ($x_{c(i)}$ where w is the length of the window, $\bar{w} = \eta l_{min}$, $1 \leq i \leq w - \bar{w}$ and $0 < \eta < 1$) The discovery accuracy is not sensitive to the choice of η and we select $\eta = 0.5$ for the rest of this paper. The same process is applied to every window in the current comparison set. The following steps are then applied for each comparison window (j) for the same candidate window (c):

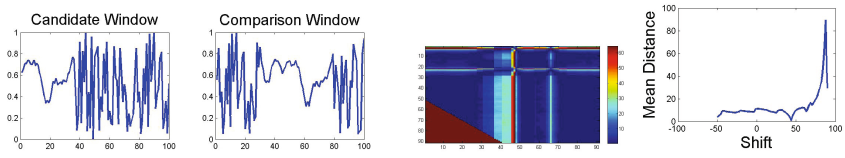
Firstly, the distances between all candidate subwindows and comparison subwindows are calculated ($D\left(x_{c(i)}, x_{j(k)}\right)$ for $i, k = 1 : w - \bar{w}$). The distance found is then appended to the list of distances at the shift $i - k$ which corresponds to the shift to be applied to the comparison window in order to get its k-th subwinodw to align with the i-th subwindow of the candidate window. By the end of this process, we have a list of distances for each possible shift of the comparison window. Our goal is then to find the best shift required to minimize the summation of all subwindow distances between the comparison window and the candidate window. Our main assumption is that the candidate and comparison windows are *larger* than the longest ARM occurrence to be discovered. This means that some of the distances in every list are not between parts of the ARM occurrences (even if an occurrence happens to exist in both the candidate and comparison windows). For this reason we keep only the distances considered *small* from each list. This can be achieved by keeping the smallest K distances from the list (where K is a user-defined parameter). In this paper, we utilize a different approach that was first proposed in [1]. The idea is to generate a window of length w from the time series by concatenating randomly selected samples from it. This window which is called the random window, is then compared to all the candidate windows and the mean distances between the $\bar{w}$ subwindows is then used as a measure of smallness. The algorithm also keeps track of the comparison subwindow indices corresponding to these *small* distances.

Finally, the comparison windows are sorted according to their average distance to the candidate window, with the best shift of each of them recorded. Comparison windows with an average distance greater than the *small* distance limit (found as described in the previous paragraph) are removed from the list to reduce the required processing time.

At the end of this process and after applying it to all candidate windows, we have for each member of $\mathtt{C}$ a set of best matching members with the appropriate shifts required to align the ARM occurrences in them (if any).

An important advantage of this technique over the one proposed in [1] is that we need not have the complete ARM inside both the candidate and comparison windows because even if a part of the ARM occurrence is contained in one of them, the alignment process implicit in calculating the shifts will still discover the similarity between contained parts of the two occurrences. This is an important advantage of the proposed algorithm because it remedies any localization inaccuracies in the change point discovery step. All what we need is that the CPD algorithm discovers locations within $w - \bar{w}$ points from the true beginning or ending of the ARM occurrence. Even if parts of multiple occurrences are contained within the candidate or comparison windows (or both), the algorithm can automatically select the appropriate occurrence to consider from each.

In practice, it is not necessary to use the complete $w - \bar{w}$ set of subwindows, as long as the number of subwindows selected is large enough to cover the complete window. As a limiting case, we can select the number of subwindows to be $\frac{w}{\bar{w}}$.



(a) Comparison and Candidate Windows (b) Dist. Matrix (c) Dist. vs. Shift

Fig. 1. Processing Steps During Best Matches Finding

Fig. 1 shows the processing steps of the proposed algorithm. Fig. 1-a shows a candidate window and a comparison window during the execution of the algorithm. The ARM occurrence in the candidate window is partial, yet the algorithm will be able to find the best fit between the two windows. Fig. 1-b shows the distances between pairs of $\bar{w}$ subwindows. Distances that correspond to subwindows of the occurrence in the two windows are much smaller than the distances elsewhere. During actual execution, this matrix need not be built but is shown here for illustration only. Fig. 1-c shows the mean distance as a function of the shift needed to align the subwindows. It is clear that the minimum of the distance happens when shifting the comparison window left by 47 positions and considering Fig. 1-a this is the correct shift required to match the two occurrences.

After finishing this step, we have for each one of the candidate subwindows a list of nearest comparison subwindows and the shift required to minimize the distance between them. This will be needed in the final step of the algorithm.

3.2 Stitching ARM Occurrences

The final step of the algorithm is to generate a set of ARMs each containing two or more ARM occurrences from the outputs of the best match finding stage. The output of this stage is an ARM graph where each clique corresponds to an ARM and each node to an occurrence. This graph is initialized to an empty graph and is filled incrementally as will be shown in this section.

The core data structure of this stage is the *matching matrix* which is constructed for each candidate window in order. Assuming we have m candidate windows and n candidate subwindows in every candidate window ($n = w - \bar{w}$), then this matrix is a square $m - 1 \times n$ matrix with each row corresponding to a comparison window and each column corresponding to a subwindow. The value at element (i,j) is the shift required to align the subwindow represented by the column i with its nearest subwindow in the comparison window j.

Using the matching matrix, we calculate the number of *contiguous* subwindows of each comparison window that match a contiguous set of subwinodws in the candidate window (e.g. having the same shift value in the matrix for more than one column). If the lengths of the resulting comparison and candidate subsequences (which is called a group) are larger than or equal to l_{min} (the minimum acceptable ARM length), a node is added (if not existing) to the ARM graph representing the comparison subsequence and the corresponding candidate subsequence and an edge connecting them is added. It is at this step that we can ignore small *gaps* of different shift values to implement don't-care sections of any predefined length.

A comparison window and the corresponding candidate window may have multiple groups which means that more than one ARM occurrence is at least partially available in these windows (of the same or different ARMs). Each one of these groups is added the graph (a new node is added only if the subsequence it represents is not existing in the graph).

By the end of this process, the ARM graph is populated and each clique of this graph represents an ARM.

4 Evaluation

The proposed algorithm was evaluated in comparison with other ARM discovery algorithms using synthetic data for which the exact locations of ARM occurrences is known. The algorithm was then applied to detection of gestures from accelerometer data and motion pattern discovery for a mobile robot simulation. This section presents these experiments. The source code of the proposed algorithm and the other four algorithms it was compared with as well as the data of these experiments are available from the authors as a part of a complete change point discovery and ARM discovery MATLAB/Octave toolbox in the supporting page of this paper at [3].

Comparing ARM discovery algorithms is not a trivial task due to the large number of possible errors that these algorithms can fall into. Discovered ARMs may cover a complete real ARM, a part of it, multiple real ARMs or nothing at all. Another problem is that an algorithm may succeed in covering all real ARMs but on the expense of adding extra parts from the time series around their occurrences to its discovered ARMs. There is no single number that can capture all of these possible problems. Nevertheless, a quantitative comparison is necessary to assess the pros and cons of each algorithm for specific tasks or types of time series. In this paper we compare algorithms along six performance dimensions.

Assume that we have a time-series x with n_T embedded ARMs ($\{\Xi_i\}$) where $1 \leq i \leq n_T$ and each ARM (Ξ_i) contains μ_i occurrences ($\{\xi_k^i\}$) for $1 \leq k \leq \mu_i$. Assume also that applying an ARM discovery algorithm to x generated n_D

ARMs ($\{M_j\}$) where $1 \leq j \leq n_D$ and each *discovered* ARM (M_j) contains o_j occurrences ($\{m_p^j\}$) for $1 \leq p \leq o_i$. Given this notation we can define the following performance measures for this algorithm:

Correct Discovery Rate (CDR): The fraction of discovered ARMs (M_js) for which each occurrence m_p^j is overlapping one and only one true ARM occurrence ξ_k^i and all these covered true ARM occurrences (ξ_k^is) are members of the same true ARM (Ξ_i).

Covering Partial-ARMs Rate (CPR): The fraction of discovered ARMs (M_js) not covered in CDR because at least one occurrence is not covering any real motif occurrence.

Covering Multiple-ARMs Rate (CMR): The fraction of discovered ARMs (M_js) for which at least one m_p^j is overlapping one real motif Ξ_i and at least one other occurrence m_q^j is covering a different motif Ξ_k .

Covering No-ARMs Rate (CAR): The fraction of discovered ARMs (M_js) for which all occurrences m_p^j are overlapping no true ARM occurrences.

Covered (C): The fraction of the time-series sequences represented by true ARM occurrences (ξ_k^is) that are covered by at least one discovered ARM occurrence m_p^j. C will always be between zero and one and represents the sensitivity of the algorithm.

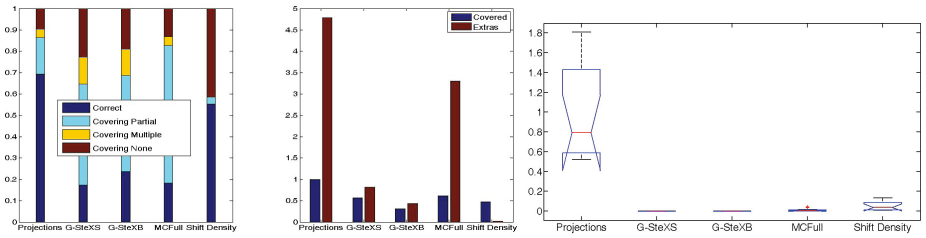
Extras (E): The length of the time-series sequences represented by discovered ARM occurrences (m_p^js) that cover no true ARM occurrence ξ_k^i. E will always be a positive number and represents the specificity of the algorithm.

The higher CDR and C and lower CPR, CMR, CAR, and E, the better the algorithm.

4.1 Synthetic Data

As a first experiment, we evaluated the proposed algorithm against four ARM discovery algorithms using synthetic data with embedded ARMs. The first comparison algorithm is MCFull which was proposed in [10] as an improvement of the basic sampling algorithm of [1]. The second and third algorithms are variations of the GSteX (Greedy Stem Extension) system proposed in [9]. This algorithm utilizes a different approach as it builds the distance graph directly from short subwindows without the shift estimation step. GSteX uses a large distance matrix that can easily become superlinear depending on the number of change points discovered. We also compare the proposed system with PROJECTIONS as explained in [16] and [2]. This algorithm utilizes SAX [5] to discritize the timeseries then applies random projections based on the work of [18]. This algorithm does not require the CPD step but it requires the specification of an exact ARM length as well as a range parameter of *near* distances.

The test data consisted of 50 timeseries of length between 2000 and 4000 points each (depending on the total number of embedded ARM occurrences) that were generated randomly from a uniform distribution ranging from -1 to 1. Depending on the experiment, a number of random ARM patterns were generated and embedded into the database and noise was then added to the complete time series. Random ARM patterns are very challenging for our CPD as there



(a) Average Performance (b) Execution Time in ms/point

Fig. 2. Average Performance of all Algorithms

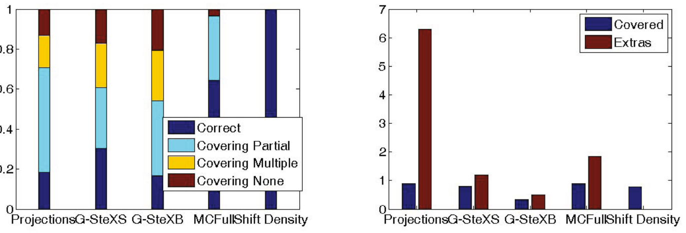


Fig. 3. Performance of all algorithms averaged over localization errors

is no underlying structure to be utilized in finding the change points. Because PROJECTIONS can only discover ARMs of a fixed predefined length, we selected $l_{min} = l_{max} = 60$ for all of our experiments.

Fig. 2(a) shows the overall performance of the five algorithms averaged over all noise levels in the previous data-set. The best performing algorithm in terms of correct discovery rate was PROJECTIONS with an average CDR of 0.69 followed by the proposed algorithm with 0.58 CDR. In terms of covered fractions/extras balance PROJECTIONS showed the highest covered fraction (highest sensitivity) but with highest extras fraction (lowest specificity). MCFull followed in terms of covered fraction but with still low specificity. The proposed algorithm showed comparable sensitivity to MCFull but with much higher specificity. Notice that these results are averaged over all noise levels. Fig. 2(b) shows the execution time in milliseconds per point for each algorithm. The proposed algorithm achieved an order of magnitude increase in speed compared with PROJECTIONS. For long time series that are encountered in real-world situations, this improvement in speed and the high specificity of the algorithm may compensate for the small reduction in correct discovery rate.

Fig. 3 shows the overall performance of all algorithms averaged over localization error. Even though the figure shows that the proposed algorithm outperforms the other algorithms in terms of correct discovery rate and specificity (E) and with comparable sensitivity to the other best performing algorithms (C), these results should not be taken at face value. Because in this experiment, CPD was performing better than what we would expect in real applications, PROJECTIONS was unfairly penalized due to its inability to utilize this information. These results represent more of an *asymptotic* behavior as the performance of the CPD algorithm improves. Nevertheless, the comparison of the proposed algorithm with MCFull, GSteXS, and GSteXB was fair and show that the proposed algorithm has higher potential of employing improvements in change point discovery accuracy.

4.2 Real World Evaluations

The first application of the proposed algorithm to real world data was in gesture discovery. Our task is to build a robot that can be operated with free hand-gestures without any predefined protocol. The way to achieve that is to have the robot *watch* as a human subject is guiding another robot/human using hand gestures. The learner then discovers the gestures related to the task by running our proposed ARM discovery algorithm to the data collected from an accelerometer attached to the tip of the middle finger of the operator's dominant hand. We collected only 13 minutes of data during which seven gestures were used. The data was sampled 100 times/second leading to a 78000 points 3D time-series. The time-series was converted into a single space time series using PCA as proposed in [12]. The proposed algorithm as well as GSteXS were applied to this projected time-series. The proposed algorithm discovered 9 gestures, the top seven of them corresponded to the true gestures (with a discovery rate of 100%) while GSteXS discovered 16 gestures and the longest six of them corresponded to six of the seven gestures embedded in the data (with a discovery rate of 85.7%) and five of them corresponded to partial and multiple coverings of these gestures.

As another proof of concept experiment, we employed a simulated differential drive robot moving in an empty arena of area 4m^2. The robot had the same dimensions as an e-puck robot and executed one of three different motions at random times (a circle, a triangle and a square). At every step, the robot selected either one of these patterns or a random point in the arena and moved toward it. The robot had a reactive process to avoided the boundaries of the arena. Ten sessions with four occurrences of each pattern within each session were collected and the proposed algorithm was applied to each session after projecting the 2D time-series into a 1-D time-series as in the previous case. The algorithm discovered 3 motifs corresponding to the three motion patterns. In this case there were no partial motifs or false positives and discovery rate was 100%.

5 Conclusions

In this paper, we proposed a new algorithm for discovering approximately recurrent motifs in time series based on shift density estimation. The main insight behind the algorithm is to convert the problem from a density estimation in the high-dimensionality time-series subsequences space into a more manageable density estimation in the single dimensional shift space. The proposed algorithm require only the specification of a lower limit on ARM occurrence lengths and can discover multiple ARMs at the same time.

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: KDD 2003, pp. 493–498. ACM, New York (2003)
3. CPMD Toolbox, `http://www.ii.ist.i.kyoto-u.ac.jp/~yasser/cpmd/cpmd.html`
4. Jensen, K.L., Styczynxki, M.P., Rigoutsos, I., Stephanopoulos, G.N.: A generic motif discovery algorithm for sequenctial data. BioInformatics 22(1), 21–28 (2006)
5. Keogh, E., Lin, J., Fu, A.: Hot sax: efficiently finding the most unusual time series subsequence. In: IEEE ICDM, p. 8 (November 2005)
6. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: The 2nd Workshop on Temporal Data Mining, pp. 53–68 (2002)
7. Minnen, D., Starner, T., Essa, I., Isbell, C.: Improving activity discovery with automatic neighborhood estimation. Int. Joint Conf. on Artificial Intelligence (2007)
8. Minnen, D., Essa, I., Isbell, C.L., Starner, T.: Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery. In: IEEE ICDM (2007)
9. Mohammad, Y., Ohmoto, Y., Nishida, T.: G-steX: Greedy stem extension for free-length constrained motif discovery. In: Jiang, H., Ding, W., Ali, M., Wu, X. (eds.) IEA/AIE 2012. LNCS, vol. 7345, pp. 417–426. Springer, Heidelberg (2012)
10. Mohammad, Y., Nishida, T.: Constrained motif discovery in time series. New Generation Computing 27(4), 319–346 (2009)
11. Mohammad, Y., Nishida, T.: Robust singular spectrum transform. In: Chien, B.-C., Hong, T.-P., Chen, S.-M., Ali, M. (eds.) IEA/AIE 2009. LNCS, vol. 5579, pp. 123–132. Springer, Heidelberg (2009)
12. Mohammad, Y., Nishida, T.: On comparing SSA-based change point discovery algorithms. In: 2011 IEEE/SICE IIS, pp. 938–945 (2011)
13. Mohammad, Y., Nishida, T., Okada, S.: Unsupervised simultaneous learning of gestures, actions and their associations for human-robot interaction. In: IEEE/RSJ IROS, pp. 2537–2544. IEEE Press, Piscataway (2009)
14. Mueen, A., Keogh, E., Zhu, Q., Cash, S.: Exact discovery of time series motifs. In: Proc. of 2009 SIAM (2009)
15. Oates, T.: Peruse: An unsupervised algorithm for finding recurring patterns in time series. In: IEEE ICDM, pp. 330–337 (2002)
16. Patel, P., Keogh, E., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. In: IEEE ICDM, pp. 370–377 (2002)
17. Tang, H., Liao, S.S.: Discovering original motifs with different lengths from time series. Know.-Based Syst. 21(7), 666–671 (2008)
18. Tompa, M., Buhler, J.: Finding motifs using random projections. In: 5th Intl. Conference on Computational Molecular Biology, pp. 67–74 (April 2001)