ICIC Express Letters
Part B: Applications
Volume **10**, Number **10**, October **2019**

ICIC International ©2019 ISSN 2185-2766
pp. **867–874**

MOTIF IDENTIFYING ON THE TIME SERIES DATA BASED ON THE HASHING TECHNIQUE

Van-Chung Pham[1] and Thoi-Duong Nguyen[2]

[1]Faculty of Information Technology
Industrial University of HCM City
No. 12, Nguyen Van Bao, Ward 4, Go Vap District, Ho Chi Minh City, Vietnam
pchung@iuh.edu.vn

[2]Saigon Thuong Tin Commercial Join Stock Bank
No. 927, Tran Hung Dao, Ward 1, District 5, Ho Chi Minh City, Vietnam
nguyenduongthoi.iuh.cs@gmail.com

Received April 2019; accepted July 2019

Abstract. *Motifs in time series data are similar subsequences which appear repeatedly several times in data set. Motif discovery is one of the general data mining techniques. In this paper, we propose a new algorithm for identifying motifs based on a hashing technique. By experiment, we compare our algorithm with Brute-Force and MK algorithms according to the execution time and accuracy of identifying motif.*
Keywords: Time series, Motif, Hashing algorithm, Motif candidate, Bucket

1. **Introduction.** Time series data arise in many fields, from science and technology to economics and finance, etc. Search for similar subsequences in a time series called motifs. This is a very necessary task in many practical applications.

Identifying motifs in time series data is used to solve problems in a variety of application areas since 2002, such as using motifs to signature verification [1], to detect duplicate images in the shape database [2], to forecast stock prices [7], to classify time series data [3] and to give useful information for the user in order to model or analyze the data [4].

There are many algorithms and trends in identifying motifs in recent times [4]. The algorithm for identifying the exact motifs (Brute-Force) is quadratic in n, the number of individual time series (or the length of the single time series from which subsequences are extracting) [5]. To increase the time efficiency in identifying motifs, some approximation algorithms are proposed [6-10]. These algorithms have the cost of $O(n)$ or $O(n \log n)$; however, they need some predefined parameters. In the few years recently, some algorithms have identified all the motifs [11] or motifs of variable lengths on very large time series (millions of points) [7].

In this paper, we introduce an algorithm to improve the effective time in the identity motif approximation on time series (the length of the motifs is known in advance); it consists of two procedures named BuildHashTable and Find_Motif. The approach of the paper is based on the random projection method introduced by Buhler and Tompa [12] and hashing techniques, the time complexity of the method in [12] is quadratic [4], but the time complexity of our approach is better (see Section 4).

The original data was normalized then reduced the number of dimensions and discrete into the form of sequences. Use a sliding window of size w (user-defined w) to slide through all the characters in the sequence [13]. The subsequences generated from the sliding window are call words [14], each word being considered as a feature. A hash table used to hold these features, and the two match features store in the same bucket.

DOI: 10.24507/icicelb.10.10.867

Looking for the largest size bucket, the components of this bucket will be motif candidates. With a given threshold (from the user), use the Euclidean distance function to select approximate motifs from candidate motifs.

The rest of the paper is organized as follows. In Section 2, we give some essential background knowledge for identifying motifs on time series. Section 3 introduces our hashing technique algorithm. Section 4 reports on the experiments of the hashing technique method in comparison to the Brute-Force algorithm and the MK algorithm [5]. Section 5 gives some conclusions and future work.

2. Background.

In this section, we provide much background knowledge as well as some related works to identify the motifs on the time series data.

2.1. Definitions.

Definition 2.1. *Time Series: A time series $T = t_1, \ldots, t_m$ is an ordered set of n real-valued variables (Elements in the set can be repeated).*

Definition 2.2. *Subsequence: Given a time series T of length m, a subsequence C of T is a sampling of length $n < m$ of contiguous position from T, that is, $C = t_p, \ldots, t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.*

Definition 2.3. *Trivial Match: Given a time series T, containing a subsequence C beginning at position p and a matching subsequence M beginning at q, we say that M is a trivial match to C if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.*

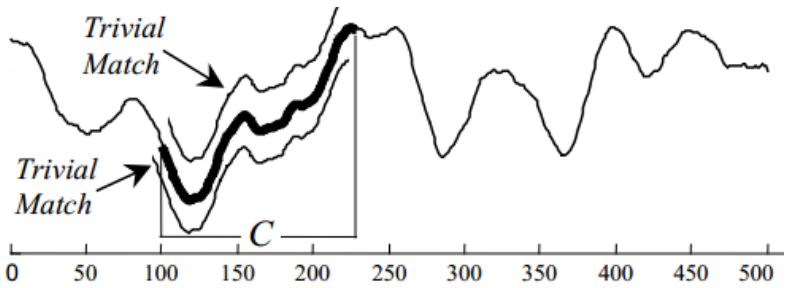

FIGURE 1. For almost any subsequence C in a time series, the best matches are the trivial subsequences immediately to the left and right of C [14].

2.2. Effectively identify motif.

Based on the experimental results, some researchers have proposed a method for determining the effectiveness of a motif search algorithm. This option is based on how the value of the efficiency coefficient decides (the smaller the coefficient, the higher the efficiency of the algorithm), as follows [14]:

$$Efficiency = \frac{Number\ of\ Time\ calls\ Euclidean\ dist}{Number\ of\ Times\ Brute\text{-}Force\ calls\ Euclidean\ dist}$$

2.3. Similar distance measurement [14,16].

Euclidean distance: The distance of two subsequences Q and C is defined as (d1).

$$D(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \quad \text{(d1)},$$

$$DR\left(\overline{Q}, \overline{C}\right) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} \left(\overline{q}_i - \overline{c}_i\right)^2} \quad \text{(d2)},$$

$$MINDIST\left(\widehat{Q}, \widehat{C}\right) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} \left(dist\left(\widehat{q}_i, \widehat{c}_i\right)\right)^2} \quad \text{(d3)}$$

When Q and C transform into PAA representations as $\overline{Q}$ and $\overline{C}$, the distance between $\overline{Q}$ and $\overline{C}$ will be (d2). $\overline{Q}$ and $\overline{C}$ continue to transform into the symbolic (words) representation of $\widehat{Q}$ and $\widehat{C}$, then MINDIST function (d3) returns the minimum distance between $\widehat{Q}$ and $\widehat{C}$.

Dynamic Time Warping Distance (DTW): Berndt and Clifford [15] introduce DTW, a point from this sequence can map into many points in the other sequence. Figure 2 illustrates this; DTW gives more accurate results but time costs are higher than Euclidean.

FIGURE 2. Using DTW, the number of times to calculate the distance between them is higher than that using the Euclidean distance [16].

2.4. **Time series data representation.** The size of the time series data is often very large. Therefore, it needs to transform into shorter and simpler data. The dimensional reduction is the representation of the n-dimensional time series data $X = (x_1, \ldots, x_n)$ into the fundamental k-dimensional lines $Y = (y_1, \ldots, y_k)$, Y is the baseline and k is a coefficient of the baseline. From the basic Y, the data can completely restore the initial X data.

The Piecewise Aggregate Approximation method (PAA) is proposed by E. Keogh et al. [17]. This method approximates k points of contiguous values into the same mean value of k points. The process does from left to right and the result is a line ladder.

The most commonly used discretization method is Symbolic Aggregate Approximation (SAX) that converts time series data into strings of characters. This method was proposed by J. Lin et al. [18]. The original data was discretized by the PAA method, each fragment in the PAA subsequence mapped to a corresponding letter based on a Gaussian distribution.

2.5. **Motif identification algorithm.** The motif search in time series data has two main approaches: exact motifs and approximate motifs. Exact motifs take a lot of time to calculate the distance between subsequences but give exact results. Approximate motifs often use methods like PAA to reduce the data size so the number of times needed to calculate the distance between the two subsequences is greatly reduced but their results deviate from the correct results.

Identifying motifs of all lengths can be done by repeatedly running the algorithm to find the motif of a given length. The method in [11] is an order of magnitude faster than running MK [5] for every length. Therefore, it is necessary to improve the execution time of the identity motifs algorithm with a given length. [7] is to find all motifs whose length is greater than l (l given) by a greedy algorithm. The highlight of this method is the execution time of only a few seconds on the time series with millions of points. However, greedy algorithms may not lead to optimal solutions.

3. **Motif Identifying by Hashing Technique Algorithm.** We present the content of identifying motifs by hashing techniques. Figure 3 shows the steps in identifying motifs and the procedure that builds the hash tables from data strings described as in Figure 4.

After doing Steps 1, 2, 3 and 4, a user-defined sliding window w extracts all subsequences, termed features. These features are inserted into a hash table. Two matching

Step 1: Normalize initial data by zero-mean method
Step 2: Determine the length of the subsequence and cut the string
Step 3: Dimensional reduction by PAA method
Step 4: Discretization by SAX method
Step 5: Identify motifs by hashing technique
Step 6: Post-test

FIGURE 3. Steps to identify motif by hash technique

```
Algorithm BuildHashTable
/*T: discreted series data, w: the sliding window size, HashTable: storage features*/
Input:  T, w
Output:  HashTable
1   for each feature in T //Travel through each of the features in T found = false;
2     for each key in HashTable //Looks in all the hash keys of hashing table
3      if( Match(feature, key))//Find the key that matches the feature
4        found = true;  trivial = false; //Confirmed found
5          for each index in HashTable[key].bucket
6            if(|index -feature.index|<w) //Examination is trivial match
7              trivial = true;//The trivial match case  break;
8          if(trivial == false)//The non-trivial match case
             // Add the feature to the corresponding hashing bucket of the key
9             HashTable[key].bucket.Add(feature.index);
10     if (found = false)
        /*If no key matches the feature,create a new hashing table containing that feature*/
11       HashTable[key].bucket.Add(feature.index);
```

FIGURE 4. The algorithm builds the hash table.

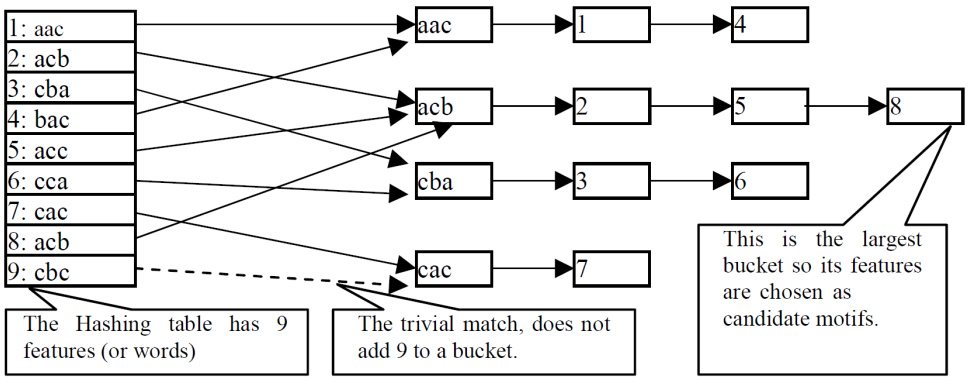

FIGURE 5. An example that extracts candidate motifs from a hash table (T = aacbaccacbc, $w = 3$)

```
Algorithm Find_Motif
    /* T: discreted series data, Threshold_Max, w: maximum distance threshold and sliding window size defined
    by the user, Hashtable: the hash table contains features, instances: set of motifs found */
    Input: T, w, Threshold_Max
    Output: L1, L2: the positions of a motif
    0  Bsf = ∞;
    1  for each feature in T is_ok = true;
    2  for each instance in instances
    3    if(isTrivial(feature,instances) or MINDIST(feature,instances)) > Threshold_Max
    4     is_ok = false; // feature is trivial match or has a greater distance than Threshold_Max
    5           break;
    6  if(is_ok = true)// The feature is a motif
         motif_instances.Add(feature);//add feature into instances
    7  for each feature in T
    8    for each instance in motif_instances
    9     d = MINDIST(feature, instance);
    10     if (isNonTrivial (feature, instance) and d <= Bsf)  Bsf = d; L1 = Loc_feature; L2 = Loc_instance;
```

FIGURE 6. The algorithm for extracting motif candidates

features will place in the same bucket (see Figure 5). The procedure that identifies the motif candidate from hash tables is described as in Figure 6.

Consider the Find_Motif algorithm, and the example in Figure 5. In Figure 5, select the largest bucket: acb → acc → acb (*) with the BuildHashTable algorithm (Two for loops in line 2 and line 5 in BuildHashTable algorithm will find the entire candidate motif in T and put in the bucket. Therefore, candidate motifs are placed in the largest bucket). In the Find_Motif algorithm: lines 2 to 6 calculate the distance of candidate motifs in (*) with features with indexes from 1 to 9 in Hashtable (Use Table 4 of [14] to calculate the distance). If any feature matches candidate motif, add the motif_intances set. Lines 7 to 10 find the position of the motifs in the motif_instances set on the original time series, L1 and L2 are the beginning and end positions of the motif in the time series. In this process, we will use distance functions (d2) and (d3) as in Section 2.3 (for detailed calculation of distance see Section 3.3 of [14]).

4. **Experimental Evaluation.** We implemented the motif identification algorithms: Brute-Force, MK and our Hashing algorithm with Microsoft Visual C # along with the Intel graphics library ZedGraph on an Intel® Core TM i2 CPU T5870, 2 GHz, RAM 4 GB, Window 7. We tested on three datasets: Stock (800, 4000, 8000, 15000 and 30000 data points), Wind (200, 4000, 8000, 12000 and 16000 data points) and Federal-Fund (1500, 2000, 4000, 8000, 15000 data points). These datasets are obtained from UCR Time Series Data Mining Archive Series.

4.1. **Experiment on Stock data.** The shapes of a motif pair on the three algorithms are shown in Figure 7. The positions of a consecutive pair of motifs identified on the time series are shown in Table 1. In Table 1: best-so-far (bsf) is the nearest distance of a consecutive pair of motifs [12], *Loc1* and *Loc2* are the starting positions of a contiguous pair of motifs on the time series, $Loc1_{avg}$ and $Loc2_{avg}$ are the average deviations of the motif positions between Hashing and Brute-Force ($Loc1_{avg}$ and $Loc2_{avg}$ are calculated based on Table 1). $Loc1_{avg} = (7+1+9+27+2)/5/256 * 100 = 3.5\%$, $Loc2_{avg} = (7+2+9+28+2)/5/256 * 100 = 3.7\%$. Table 2 shows the execution times of three algorithms on Stock data.

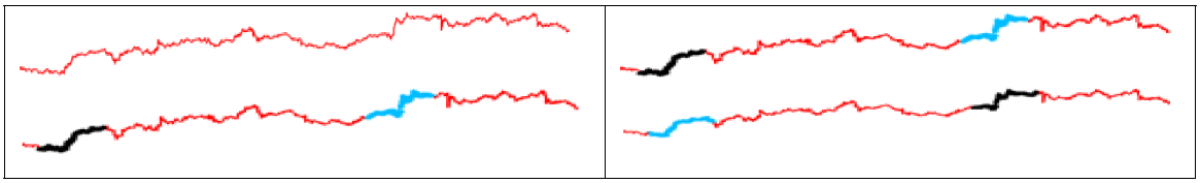

FIGURE 7. The shape of the identified motifs (bold) on the time series has 8000 points.

TABLE 1. Positions of a pair of motifs and their deviation on Stock data

Position of a pair of motifs									**Deviation of motifs**			
Brute-Force (BF)			**MK**			**Hashing**			**Hashing – BF**		**Hashing – MK**	
Loc1	***Loc2***	***bsf***	***Loc1***	***Loc2***	***bsf***	***Loc1***	***Loc2***	***bsf***	***Loc1***	***Loc2***	***Loc1***	***Loc2***
321	772	6.03	321	772	6.03	314	765	4.39	7	7	7	7
298	1682	3.24	298	1682	3.24	297	1680	2.19	1	2	1	2
298	1682	3.24	298	1682	3.24	289	1673	2.23	9	9	9	9
5233	14203	2.67	5233	14203	2.67	5260	14231	2.36	27	28	27	28
18792	29712	1.05	18792	29712	1.05	18794	29714	0.02	2	2	2	2

TABLE 2. The efficiency ratios of Hashing/MK on Stock data

Execution time (second)					Efficiency			
BF	**MK**	**Hashing**	**BF/ Hashing**	**MK/ Hashing**	**BF**	**MK**	**Hashing**	**Hashing /MK**
0.43	0.21	0.20	2.11	1.04	1	0.18	0.005	33.81
7.2	1.37	3.89	1.85	0.35	1	0.04	0.002	21.52
28.16	5.91	17.32	1.63	0.34	1	0.05	0.001	33.01
135	18.53	80	1.69	0.23	1	0.04	0.0004	101.98
158	27.684	97.13	1.63	0.29	1	0.08	0.004	19.43

From the experimental results on Stock data, we can see that:

- The positions of a pair of motifs have average deviations are $Loc1_{avg} = 3.5\%$ and $Loc2_{avg} = 3.7\%$ when compared to Brute-Force and these deviations increase as the number of dimensions decreases.
- The execution time of the Hashing is slower than the MK but the position deviation of the motif is almost the same as MK.
- Changing parameters with PAA = 64 and PAA = 32, we also obtained similar results.

4.2. **Experiment on Wind data.** The experimental results obtained from the experiment on the Wind data are shown in Table 3. The average deviations of the motif positions between Hashing and Brute-Force are $Loc1_{avg} = 114\%$ and $Loc2_{avg} = 34\%$. The shape of the pair of consecutive motifs and execution time on Wind data of the three algorithms are shown in Figure 8.

TABLE 3. The starting positions of a pair of motifs on Wind data

Position of a pair of motifs									Deviation of motifs			
BF			**MK**			**Hashing**			**Hashing – BF**		**Hashing – MK**	
Loc1	***Loc2***	***bsf***	***Loc1***	***Loc2***	***bsf***	***Loc1***	***Loc2***	***bsf***	***Loc1***	***Loc2***	***Loc1***	***Loc2***
1154	1446	6.31	1154	1446	6.31	1228	1515	4.86	74	69	74	69
1154	1446	6.31	1154	1446	6.31	1228	1515	4.68	74	69	74	69
4998	5960	5.99	4998	5960	5.99	3725	6225	4.63	1273	265	1273	265
11672	9016	5.39	11672	9016	5.39	11652	8997	4.09	20	19	20	19
9016	11672	5.39	9016	11672	5.39	8997	11652	4.09	19	20	19	20

From the experimental results on Wind data, we can see that:

- The execution time of Hashing algorithm is better than the MK algorithm, while the position deviation of the motif varies quite low.
- In the case of the motif of the Hashing algorithm, there is more deviation (time series with 8000 points), when a time series has many extremes (Figure 8(a)). Hashing algorithm using PAA method, in addition to advantages, the PAA also has some disadvantages; it minimizes dimensionality by the mean values of equal-sized frames. This mean value may cause a possibility to miss some important points in the time series [19]. Furthermore, using Euclidean measurements, in this case, can cause the accuracy of poor motifs.
- Changing parameters with PAA = 64, subsequence length = 256 and PAA = 128, subsequence length = 512, we also obtained similar results.

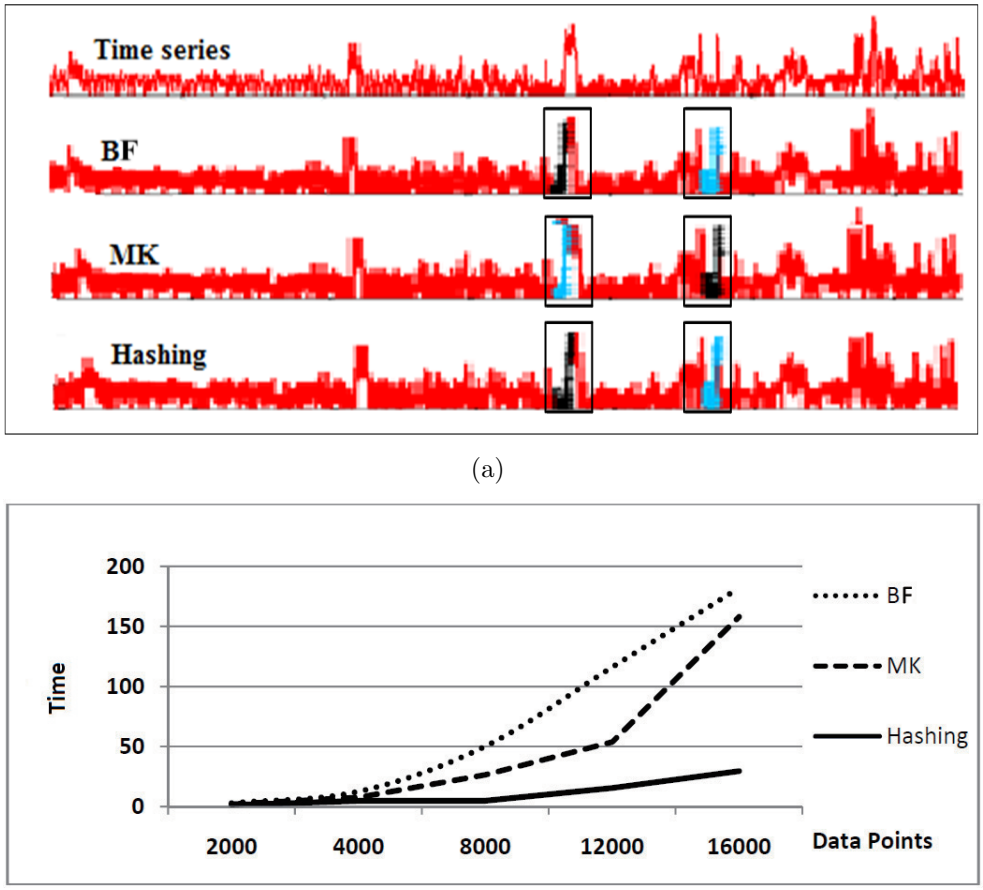

FIGURE 8. (a) The shape of the motifs (rectangular frame), (b) execution time on Wind data

4.3. **Experiment on Federal-Fund data.** Experiments on the Federal-Fund are like Stock data and Wind data, time series with subsequence length = 512 and PAA are 32, 64 and 128 respectively. We also obtained similar results. The execution time of Hashing algorithm is better than MK algorithm and the deviation of position of motif compared with Brute-Force algorithm is relatively low: $Loc1_{avg} = 6.32\%$, $Loc2_{avg} = 5.7\%$, (PAA = 32); $Loc1_{avg} = 9.58\%$, $Loc2_{avg} = 9.3\%$, (PAA = 64); $Loc1_{avg} = 9.41\%$, $Loc2_{avg} = 9.17\%$, (PAA = 128).

Through our empirical results on three datasets, we have the following observations.

- The length of the subsequences varies but with the same rate of reducing the dimension; the Hashing algorithm gives better execution time and efficiency than the MK algorithm.
- The thicker dataset, the Hashing algorithm gives the motifs closer to being correct.
- Hashing algorithm produces poor results when the dataset has many extreme points.
- Hashing algorithm runtime is better than MK algorithm because it uses hashing technique to group features in bottom-up. However, the motif place is only approximate when compared to Brute-Force algorithm.
- Deviation to the motif's place is relatively small and the shape of the motifs is almost identical to the Brute-Force and MK algorithms.

5. **Conclusions.** We use the hash technique to improve the execution time in identifying motif. Through the Hashing algorithm on three datasets, the execution time has an encouraging improvement. The Hashing algorithm also has some limitations; for example,

the results of motifs found are affected when the time series has many extremes; the length of the motif must be known in advance and the size of the time series is small. In the last few years there were some works finding approximate motif with variable length [7], finding all motifs of different lengths [11]. Current data is increasing very fast, so we need to experiment on a large time series, which also stimulates us to find appropriate algorithms that increase the accuracy of motif and shorten run time. For future work, we continue to research to solve the limits that we are facing.

REFERENCES

[1] C. Gruber, M. Coduro and B. Sick, Signature verification with dynamic RBF networks and time series motifs, *Proc. of the 10th Int. Workshop on Frontiers in Handwriting Recognition*, 2006.

[2] X. Xi, E. J. Keogh, L. D. Wei and A. Mafra-Neto, Finding motifs in a database of shapes, *Proc. of SIAM International Conference on Data Mining*, Minneapolis, MN, USA, 2007.

[3] K. Buza and L. S. Thieme, Motif-based classification of time series with Bayesian networks and SVMs, in *Advances in Data Analysis, Data Handling and Business Intelligences, Studies in Classification, Data Analysis, Knowledge Organization*, A. Fink et al. (eds.), Springer, Berlin, Heidelberg, 2010.

[4] S. Torkamani and V. Lohweg, Survey on time series motif discovery, *International Journal of Business Intelligence and Data Mining*, vol.1, no.1, 2017.

[5] A. Mueen, E. Keogh, Q. Zhu, S. Cash and B. Westover, Exact discovery of time series motifs, *Proc. of SIAM International Conference on Data Mining*, 2009.

[6] P. Beaudoin, M. Panne and S. Coros, Motion-Motif graphs, *Symposium on Computer Animation*, 2008.

[7] Y. Gao and J. Lin, Efficient discovery of variable-length time series motifs with large length range in million scale time series, *arXiv:1802.04883vl1 [cs.DS]*, 2018.

[8] Y. Li, *Subseries Join and Compression of Time Series Data Based on Non-uniform Segmentation*, Ph.D. Dissertation, School of Computer Science, University of Waterloo, 2008.

[9] J. Meng, J. Yuan, M. Hans and Y. Wu, Mining motifs from human motion, *Proc. of EUROGRAPHICS*, 2008.

[10] D. Minnen, C. L. Isbell, I. Essa and T. Starner, Discovering multivariate motifs using subsequence density estimation and greedy mixture learning, *The 22nd Conf. on Artificial Intelligence (AAAI'07)*, 2007.

[11] A. Mueen and N. Chavoshi, Enumeration of time series motifs of all lengths, *Knowledge and Information Systems*, vol.45, no.1, pp.105-132, 2015.

[12] J. Buhler and M. Tompa, Finding motifs using random projections, *Journal of Computational Biology*, vol.9, no.2, pp.225-242, 2002.

[13] Y. Li, J. Lin and T. Oates, Visualizing variable length time series motif, *Proc. of the SIAM International Conference on Data Mining*, 2012.

[14] J. Lin, E. Keogh, S. Lonardi and P. Patel, Finding motifs in time series, *Proc. of the 2nd Workshop on Temporal Data Mining (KDD'02)*, 2002.

[15] D. J. Berndt and J. Clifford, Using dynamic time warping to find patterns in time series, *AAA1-94 Workshop on Knowledge Discovery in Databases*, pp.359-370, 1994.

[16] E. Keogh and C. A. Ratanamahatana, Exact indexing of dynamic time warping, *Knowledge and Information Systems*, vol.7, pp.358-386, 2005.

[17] E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, *Knowledge and Information System*, vol.3, pp.263-286, 2001.

[18] J. Lin, E. Keogh, S. Lonardi and B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, *Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, 2003.

[19] B. Lkhagva, Y. Suzuki and K. Kawagoe, Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation, *DEWS2006 4A-i8*, 2006.