Parallel Variable-Length Motif Discovery in Time Series Using Subsequences Correlation

Chuitian Rong[1,2(✉)], Lili Chen[1], Chunbin Lin[3], and Chao Yuan[1]

[1] School of Computer Science and Technology, Tiangong University, Tianjin, China
{chuitian,yuanchao}@tiangong.edu.cn
[2] Tianjin Key Laboratory of Autonomous Intelligence Technology and Systems, Tianjin, China
[3] Amazon AWS, Seattle, USA
lichunbi@amazon.com

Abstract. The repeated patterns in a long time series are called as time series motifs. As the motifs can reveal much useful information, time series motif discovery has been received extensive attentions in recent years. Time series motif discovery is an important operation for time series analysis in many fields, such as financial data analysis, medical and health monitoring. Although many algorithms have been proposed for motifs discovery, most of existing works are running on single node and focusing on finding fixed-length motifs. They cannot process very long time series efficiently. However, the length of motifs cannot be predicted previously, and the Euclidean distance has many drawbacks as the similarity measure. In this work, we propose a parallel algorithm based on subsequences correlation called as PMDSC (Parallel Motif Discovery based on Subsequences Correlation), which can be applied to find time series motifs with variable lengths. We have conducted extensive experiments on public data sets, the results demonstrate that our method can efficiently find variable-length motifs in long time series.

Keywords: Time series · Motif discovery · Parallel · Spark

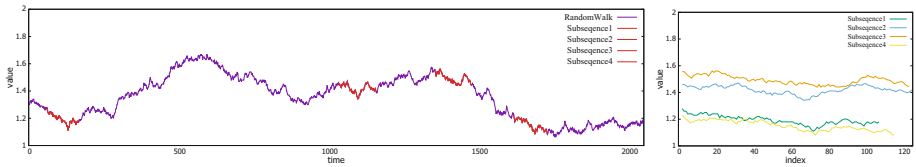
1 Introduction

Time series Motif Discovery is a task to discover repeated and similar (correlated) patterns in time series. In recent years, time series motif discovery has been received extensive attentions. It has been applied in many time series data mining and analysis tasks, such as data classification, clustering, activity recognition and outlier detection.

In the past decade, a large number of motif discovery algorithms have been proposed. Most existing algorithms aim to find fixed-length motifs. While, the length of motifs cannot be predicted previously in most cases. So, some interesting motifs with different lengths will be lost. Mostly, finding variable-length

X. Wang et al. (Eds.): APWeb-WAIM 2020, LNCS 12318, pp. 164–175, 2020.
https://doi.org/10.1007/978-3-030-60290-1_13

motifs in the time series can reveal more latent patterns than fixed-length motifs. So, there are some works such as [3,4] have tried to discover the variable-length motifs. The biggest challenge of variable-length motifs discovery is the massive computations. The complexity of variable-length motif discovery is 10 times higher than the fixed-length motif discovery in [17]. For example, if the lengths of motifs are ranging from 300 to 10300, the brute-force algorithm will take 5×10^{18} Euclidean distance calls [3]. In fact, many industrial applications generate very large time series. While, most algorithms use a single compute node to analyze large-scale time series. So, that is difficult to complete the analysis in a feasible time. Therefore, in recent years, distributed and parallel computing platforms are widely used in the data mining and analysis of large-scale time series.



(a) Motifs in Time Series (b) Discovered Motifs

Fig. 1. Variable-length motifs discovered by PMDSC algorithm on random walk dataset

In addition, there are some other limitations in these motif discovery algorithms. For example, the algorithms apply Euclidean distance as the similarity and require the two compared subsequences with the same length, which is not suitable in most applications. So, we propose a variable-length motif discovery algorithm using Pearson Correlation Coefficient as the similarity measure. The Pearson Correlation Coefficient is a commonly used similarity measure for time series data mining due to its multiple beneficial mathematical properties, such as it is invariant to scale and offset. The Pearson Correlation Coefficient can reveal the true similarity of two time series. Therefore, we argue that the Pearson Correlation Coefficient is a good similar measure in time series motif discovery. In Fig. 1, we showed one of the results found by our proposed algorithm.

In order to solve the existing drawbacks and improve the efficiency of motif discovery in long time series, we introduced a parallel motif discovery algorithm on Spark platform. This algorithm applies the Pearson Correlation Coefficient as the similarity measure to find variable-length motifs in large-scale time series.

In short, the main works of this paper are:

1. We propose a parallel algorithm for variable-length motifs discovery based on subsequences correlation using Spark.
2. In order to compute the correlation efficiently, we proposed a parallel FFT algorithm using Spark.
3. In order to improve the concurrency of parallel jobs, we proposed a time series segmentation method and dot product matrix partition method.

4. We demonstrate the efficiency and scalability of the proposed algorithm using extensive experiments.

The rest of the paper is organized as follows. Section 2 discusses Related Works of motif discovery. In Sect. 3, we introduce the problem definition and background concepts used in this paper. In Sect. 4, we introduce our proposed parallel PMDSC algorithm. The experimental results are shown in Sect. 5 and the conclusions are given in Sect. 6.

2 Related Works

Time series motif discovery approach was proposed in 2002 [7]. Then, time series motif discovery has received extensive attentions and many motif discovery algorithms have been proposed. These methods discover the motif as the most similar subsequences using some similar measures. Time series motif discovery is a basic operation in time series data analysis, so a large amount of motif discovery research works have been proposed in recent years.

The existing algorithms for time series motif discovery are mainly divided into two strategies: fixed-length and variable-length. The fixed-length algorithms proposed in [15] are based on SAX (Symbolic Aggregate approXimation) [6], which was applied to represent the time series with symbols. These methods mainly focus on fixed-length motif discovery. [10] proposed a MK algorithm to find the most similar pair subsequences as motifs, which is adopted a pruning method to speed up the Brute Force Algorithm. In [5], authors introduced a Quick-Motif algorithm, whose calculation speed is increased by 3 orders of magnitude compared with the traditional fixed-length motif discovery algorithm in [10]. Several recent works focus on fixed-length motif discovery, [16] introduces an algorithm named STAMP combined with MASS algorithm[1] to find exact motifs for a given length. An algorithm named STOMP was proposed in [17], which reduced the time complexity of STAMP from $O(n^2 \log n)$ to $O(n^2)$.

As variable-length motifs can reveal much more interesting latent patterns than fix-length do, many research works have focused on variable-length motif discovery in recent years. In 2011, the VLMD algorithm [11] was proposed by calling fixed-length motif discovery algorithm to find K pair-motifs with variable-length. In [9], the authors proposed an algorithm using Euclidean distance as the similarity measure for Z-Normalized segments. It applied a lower-bound to reduce the computing time of variable-length motifs discovery. In [14], the authors proposed a novel method, which incorporated the grammar induction to find approximate motifs with variable-length. Its running time is faster than other algorithms. However, the idea of this algorithm is based on grammar induction, so this method may be limited in some applications. [2] proposed a method based on discretization and the subsequences do not overlap with their adjacents, which may lead to loss some real results. [3] introduced an algorithm named HIME based on SAX and Induction Graph to find variable-length motifs. This

[1] https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

approach can find exact motifs in an acceptable time. However, this method is difficult to implement.

In summary, the existing motif discovery algorithms mostly find fixed-length motifs and mainly using Euclidean distance as a measure of similarity. As illustrated above, these algorithms have many limitations. In this paper, we propose an efficient parallel time series motif discovery method on Spark. Our approach is using subsequences correlation, combined with parallel FFT algorithm and time series segmentation method, can efficiently find motifs with variable-length.

Table 1. Symbols and Definitions

Symbol	Definition
T	Time Series T
$\|T\|$	The length of time series T
L	The length of motif
L_{min}	The smallest length of motif
len	The length of the subsequence in each segment
$Corre(T', T'')$	Correlation coefficient between subsequences T' and T''
$W(n)$	Butterfly coefficient of FFT
$X(k)$	The result of FFT
I	The index of a data point in the time series
P	The segment index during time series segmentation
$\mathcal{Z}$	The Dot-Product Matrix of two time series subsequences
$\mathcal{Z}'$	A number of consecutive columns(one block) in $\mathcal{Z}$

3 Problem Definition and Background

In this section, we present the problem definitions and introduce the background concepts used in this paper. The symbols used in this paper are listed in Table 1.

Definition 1 *(Time Series).* *A Time Series T is a sequence of real numbers observed in the same time interval. $T = [t_1, t_2, \ldots, t_n]$, where n is the length of time series T.*

Definition 2 *(Subsequence).* *A Subsequence with the length of m in time series T is a set of continuous points $T[j : j + m] = [t_j, t_{j+1}, \ldots, t_{j+m-1}]$ starting at position j.*

Definition 3 *(K-Frequent Motif). Given a time series T and a minimum length L_{min} of motif, K-frequent motif of T is defined as a set of subsequences that have at least K matches and denoted as ϕ, $|\phi| \geq K$. $\phi = \{T'' \,|\, \forall T' \subset T, \exists T'' \subset T \wedge Corre(T', T'') \geq \theta\}$. T' and T'' are subsequences of the time series T with $|T'| \geq L_{min}$ and $|T''| \geq L_{min}$. $Corre(T', T'')$ is the correlation coefficient between T' and T'', θ is a similarity threshold of the correlation given by user.*

Definition 4 *(Pearson Correlation Coefficient). Pearson Correlation Coefficient is a measure of the correlation between two variables. It can reflect the degree of similarity between two subsequences. For two subsequences T' and T'', the Pearson Correlation Coefficient can be computed as following.*

$$Corre(T', T'') = \frac{(E[(T' - E(T'))(T'' - E(T''))])}{(\sigma_{T'}\sigma_{T''})} \tag{1}$$

The above calculation formula of the Pearson Correlation Coefficient can also be defined as Formula 2, where T' and T'' are the subsequences of T, $u_{T'}$, $\mu_{T''}$ and $\sigma_{T'}$, $\sigma_{T''}$ are the mean and Standard deviation of T' and T'' respectively, and the $\sum T'T''$ can be calculated from the dot product between the subsequences.

$$Corre(T', T'') = \frac{(\sum T'T'' - m\mu_{T'}\mu_{T''})}{(m\sigma_{T'}\sigma_{T''})} \tag{2}$$

FFT (*Fast Fourier Transform*) is an efficient algorithm to compute the DFT (*Discrete Fourier Transform*) of a sequence. In many applications, we are interested in finding motifs or the similar shapes. Before computing the correlation coefficient, we normalized two subsequences by *Z-Normalization*. After *Z-Normalization*, FFT can be used to compute the cross products of arbitrary subsequences of two sequences. By doing this, the computational time complexity of $\sum T'T''$ in Formula 2 can be reduced to $O(n \log n)$. In order to improve the efficiency of motif discovery, we implemented a parallel FFT on Spark. In order to avoid redundant computations, we computed the dot products of all time series subsequences previously using parallel FFT and organized them as a Dot-Product-Matrix $\mathcal{Z}$.

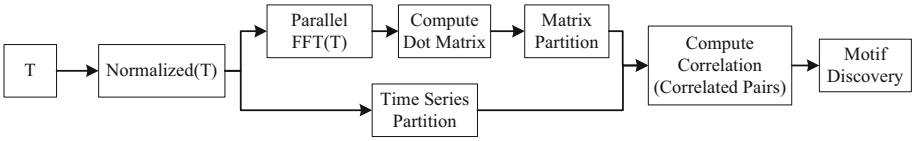


Fig. 2. The Framework of PMDSC Algorithm

Algorithm 1: `Segmentation`$(T, P_{max}, len, L_{min})$

Input : T: the normalized time series
P_{max}: the number of partitions
len: the length in each partition
L_{min}: the minimum length of subsegments

1 **foreach** $T(i) \in T$ **do**
2 $newValue \leftarrow < i, T(i) >$;
3 **for** $P = 0 \rightarrow P_{max}$ **do**
4 **if** $key \geq P * (len - L_{min}) \&\& key \leq (P+1) * len - P * L_{min}$ **then**
5 $midresult \leftarrow \langle P, newValue \rangle$;
6 $partitions \leftarrow midresult.partitionBy(\_.1)$;
7 $segments \leftarrow partitions.gorupByKey(\_.\_.1)$;

4 Parallel Variable-Length Motifs Discovery

In this section, we present the implementation details of the parallel motif discovery using subsequences correlation using Spark. Our approach is based on new parallel FFT and data segmentation techniques. Figure 2 shows the framework of our motif discovery algorithm. In following, we first introduce the time series segmentation method. Then, we describe the Dot-Product-Matrix computation and partition method. Finally, we describe the details of motif discovery.

4.1 Time Series Segmentation

In order to make better utilize the properties of RDD (Resilient Distributed DataSet), we introduced a partition approach that divides time series into multiple segments (subsequences) with equal length *len*. To assure the exact of results, we keep an overlap between the adjacent segments with the length of L_{min}. For each segment, we can get the range of indices for its data points by using Formulas 3 and 4, in which I_{min} and I_{max} are the minimum and maximum data points indices in S_P, respectively. Also, we can get the number of segments P_{max} for a time series based on the *len* and L_{min} using Formula 5. The Algorithm 1 describes its implementation using Spark.

$$I_{min} \geq P * (len - L_{min}) \tag{3}$$

$$I_{max} \leq (P+1) * len - P * L_{min} \tag{4}$$

$$P_{max} = \frac{n - L_{min}}{len - L_{min}} + 1 \tag{5}$$

In Spark, we use the *map*, *partitionBy* and *groupByKey* operators to complete the time series segmentation task as presented in Algorithm 1. As presented in Algorithm 1, the *map* operator is used to transform the time series data points into $\langle key, value \rangle$ pairs (Lines 1–2). Here, the *value* is the data point *T(i)* and the *key* is the index i of the data point. For each data point, we assign its partition number P as its new key according to Formulas 3 and 4 and transform it

Algorithm 2: $\mathcal{Z}-$ComputationAndPartition$(T, P_{max}, len_1, len_2)$

```
   Input  : T: the normalized time series
            P_max: the partition number of time series
            len_1, len_2: the length of two subsequences
   Output: Z': the blocks of dot product matrix
1  N ← T.length;
2  for i = 0 → N do
3  |  T_i ← T.takeRight(m − i);
4  |  for j = 0 → 2*N do
5  |  |  T ← Array[T, 0];
6  |  |  T_i ← Array[T_i, 0];
7  |  T ← FFT(T), T_I ← FFT(T_i);
8  |  Z ← T * T_I;
9  for P_1 = 0 → P_max do
10 |  for P_2 = 0 → P_max do
11 |  |  if P_1! = P_max&&P_2! = P_max then
12 |  |  |  v = N + (P_2 − P_1 + 1) * L_min;
13 |  |  |  v_1 = (v + (P_1 − 1) * len_1 − P_2 * len_2);
14 |  |  |  v_2 = (v + P_1 * len_1 − (P_2 − 1) * len_2);
15 |  |  |  for j = v_1 → v_2 do
16 |  |  |  |  Z' ← iFFT(Z(j));
17 |  |  |  Z' ← < (P_1, P_2), Z' >
```

into $\langle P, (i, T(i))\rangle$ (Lines 3–5). Then, the *partitionBy* operator is used to partition the new key/value pairs into multiple partitions according to the partition number (Line 6). Finally, the *groupByKey* operator is used to aggregate the new key/value pairs in multiple groups with the same order as in the original time series (Line 7). By doing this, the segmentation is completed.

Each time series is processed in the similar way. After that, in order to pair all possible subsequences. First, we use the *map* operator to assign a new key to the divided subsequences according to the maximum partition number P_{max} of each time series. Then, we joined the subsequences from different two time series using the *join* operator. This step generates records in the form $\langle (P_1, P_2), (Iterable[T'], Iterable[T''])\rangle$, where P_1 and P_2 are the indices of two subsequences.

4.2 Dot-Product-Matrix Computation and Partition

Variable-length motifs discovery can reveal much more interesting latent patterns than fix-length motifs discovery. However, it is a very expensive in time cost. As the motif discovery is a pairwise operation to compute the correlations of all possible subsequences, there are many redundant computations [8]. In order to avoid redundant computations, we compute the correlations previously using parallel FFT and store the results in the Dot-Product-Matrix $\mathcal{Z}$. In fact, the matrix partition is implemented during its computation and stored in multiple distributed blocks, as shown in Algorithm 2.

The Dot-Product-Matrix $\mathcal{Z}$ stores the shift-cross products of all possible subsequences from tow time series. For parallel processing, the naive way to use the $\mathcal{Z}$ is to send it to all worker nodes. In fact, just a little of columns in $\mathcal{Z}$ is used to compute the correlations of each pair subsequences. In order to avoid unnecessary data transfer and improve the efficiency, we proposed a partition technique for $\mathcal{Z}$. For each pair of subsequences, we can get the corresponding blocks in $\mathcal{Z}$ according to their partition numbers. Without losing generality, we assuming the partition number P_1 and P_2 of two subsequences T' and T'' with lengths len_1 and len_2. The necessary corresponding part $\mathcal{Z}' \in \mathcal{Z}$ for computing Corre(T'', T') is from $\mathcal{Z}[*][|T| + (P_2 - P_1 + 1) * L_{min} + P_1 * len_1 - (P_2 + 1) * len_2]$ to $\mathcal{Z}[*][|T| + (P_2 - P_1 - 1) * L_{min} + (P_1 + 1) * len_1 - P_2 * len_2]$.

In Algorithm 2, the first part is to compute the $\mathcal{Z}$ (lines 1–8). It gets all subsequences of T (lines 2–3)and extend their length to the twice of length T (lines 4–6). Then, each subsequence T_i of T and itself will be transformed using FFT (line 7) and get their cross products (line 8). Finally, a part of the Dot-Product-Matrix is retrieved by doing inverse Fourier transform (line 9). This algorithm returns a two-dimensional array, which contains the sum of the products of the elements in T and T_i for different shifts of T. The *FFT()* (line 7) ensures that this process can be done in $O(n \log n)$ time. The *FFT* is a parallel implementation [13]. The parallel FFT algorithm is implemented as the following four steps.

1. Get the original index for each element in time series.
2. Compute the binary code $\mathcal{B}(\mathcal{I})$ for each element according to its original index and the length of time series.
3. Compute the butterfly coefficient using Formula 6 based on the values of bits in $\mathcal{B}(\mathcal{I})$.
4. Compute the final result for each element using Formula 7.

$$W(n) = \prod W_{2^j}^k * (-1)^t, k \in [0, 2^{j-1}) \quad t = \begin{cases} 0 & \text{n = k} \\ 1 & \text{n = k + } 2^{j-1} \end{cases} \tag{6}$$

$$X(n) = \sum_{i=0}^{n-1} x(i) * W(n) \tag{7}$$

The last part of Algorithm 2 (lines 9–16) shows the details of the matrix $\mathcal{Z}$ partition. When computing the $\mathcal{Z}$, we assign a *key* to the elements belongs to the same block according to the partition number of the two subsequences. The $\mathcal{Z}$ is organized and stored in the form of key/value pairs. By doing this, we completed the matrix partition task. When performing motif discovery and computing the correlations of each pair of subsequences, the subsequence pair along with the corresponding block of $\mathcal{Z}' \in \mathcal{Z}$ will be grouped together and shuffled to the same worker node according to their assigned *key* in Algorithm 3.

Algorithm 3: `CorrelationComputation`$(list, L_{min}, \mathcal{Z}', \theta)$

Input : $list$: $\langle (Segment_{T'}, Segment_{T''}) \rangle$
L_{min} : the minimum length of motif
$\mathcal{Z}'$: the corresponding block in $\mathcal{Z}$
θ : the threshold of correlation
Output: *result*: the motifs with correlation $\geq \theta$

1 $n \leftarrow list.\_1.length$, $m \leftarrow list.\_2.length$;
2 $m_{len} = L_{min}$;
3 **for** $i = 0 \rightarrow m$ **do**
4 **for** $j = 0 \rightarrow n$ **do**
5 $maxLength \leftarrow \min(m - i + 1, n - j + 1)$;
6 $len \leftarrow L_{min}$;
7 **while** $len < maxLength$ **do**
8 $\sum T'T'' \leftarrow \mathcal{Z}'$;
9 /* $S_i \in Segment_{T'}, S_j \in Segment_{T''}$ */
10 $mean \leftarrow getMean(S_i, S_j)$;
11 $stdv \leftarrow getStdv(S_i, S_j)$;
12 $C \leftarrow \frac{\sum T'T'' - len * mean}{len * stdv}$;
13 **if** $C > \theta \&\& m_{len} \geq len$ **then**
14 $m_i = i, m_j = j$;
15 $m_{len} = len$;
16 $result \leftarrow (m_i, m_j, m_{len}, C)$;
17 $len \leftarrow len + +$;
18 /*filter out the covered subsequence pairs*/
19 $result \leftarrow result.maxBy(m_{len})$;

4.3 *K*-Frequent Motif Discovery

In this part, we introduce the algorithms for discovering K-Frequent motifs, as shown in Algorithm 3. Algorithm 3 is used to compute the correlation of all possible subsequence pairs and filter out the covered subsequence pairs to get the longest ones.

In Algorithm 3, the input *list* contains two segments, $Segment_{T'}$ and $Segment_{T''}$, generated by using segmentation method 1 on two time series. In order to get variable-length motifs, we should compute the correlations of all possible subsequences contained in these two segments (Lines 3–11). According to the Dot-Matrix-Partition method, only a little block of $\mathcal{Z}' \in \mathcal{Z}$ is needed for each pair of segments to compute the correlations of contained subsequences (Line 8). For each pair of subsequences (S_i, S_j), in which $S_i \in Segment_{T'}$ and $S_j \in Segment_{T''}$, the mean and standard deviation of data points contained by S_i and S_j should be computed (Lines 9–10). After that, we can use Formula 2 to compute the correlation coefficient of two subsequences (Line 11). Then, we will apply the filtering method to select the required subsequence pairs (Lines 12–15). It's to be noted that, some found short subsequence pairs can be covered by the longer ones. So, it is necessary to remove the covered subsequence pairs. At the last step, we filter out the short ones and keep the long ones (Line 17). Finally, Algorithm 3 returns the longest subsequences pair contained in each segments pair.

After that, we use *groupByKey* operator to aggregate the motifs with the same key, and we can get a series motifs of the form $\langle Pid_1, Iterable(Pid_2, len, Correlation)\rangle$. So, we can find the motifs whose frequency is more than K easily.

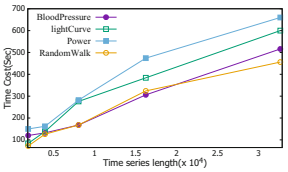


Fig. 3. Time series length.

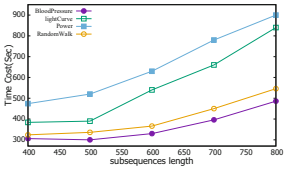


Fig. 4. Partition segment length.

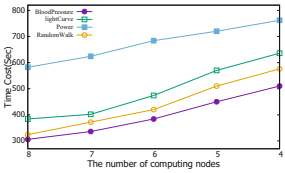


Fig. 5. Computing nodes number

5 Experiments

We have implemented our proposed algorithm using the Scala programming language on Spark. Our experiments are conducted on Spark-2.1.0 cluster, in which there is one master node and 8 worker nodes. Each node in the cluster are equipped with Linux CentOS 6.2, 4 cores, 6 GB of memory and 500 GB of disk storage. In this section, we evaluated the experimental results with the variations of different parameters, including the segment length of motif, the length of time series and the threshold θ. We also testified the scalability by changing the number of computing nodes. All the experiments are tested on public datasets that are downloaded from the web site[2]. There are four datasets used in our experiments, including *Blood Pressure* [1], *Light Curve* [12], *Power* [8] and *Random Walk*.

5.1 Effect of Time Series Length

In this experiment, we verify the efficiency of PMDSC by changing the length of time series from 2000 to 32,768. In this experiment, we set the minimum motif length $L_{min} = 100$, the correlation threshold $\theta = 0.9$ and the partitioned segment length $len = 400$. The experimental results are shown in Fig. 3.

From the Fig. 3, we can observe that the time cost is increasing near linearly on the four different data sets as the length of time series increasing. Compared with the time costs on these four data sets, we find that the time cost on the *Power* data is the most. It is because the *Power* dataset has large spikes that cause increased time to compute the correlation in each partition. The time costs on the two datasets *Blood Pressure* and *Random Walk* is smaller and near the same. The reason is that after normalization, the spikes and changing frequencies of the two datasets are like to each other.

[2] https://files.secureserver.net/0fzoieonFsQcsM.

5.2 Effect of the Segment Length

In this test, we test the time costs by changing the segment length len from 400 to 800 with the interval 100. In this experiment, we set the time series length to 16,384, the correlation threshold $\theta = 0.9$ and $L_{min} = 100$. Figure 4 shows the effects of segment length variations on time costs.

From Fig. 4, we can find that the time costs are increasing on four different datasets as the segment length increasing. The reason is that increasing the segment length will bring more subsequence pairs to be processed in each segment. We can also find that the time costs increasing trend is different on four data sets. When the segment length is more than 500, the time costs increase mostly on *Power* data set followed by *Light Curve* data set. It is caused by the large variations of data values contained in the two time series. While, the distribution of data point values in other two data sets are relatively stable. So, the time costs changing on *Blood Pressure* and *Random Walk* is relatively smaller.

5.3 Effect of the Number of Computing Nodes

In this part, we test the scalability of our proposed method by changing the number of computing nodes in the cluster. In this experiment, we set the length of time series to 16,384, the segment length $len = 400$ and the correlation threshold $\theta = 0.9$. The experimental results are shown in Fig. 5. In Fig. 5, the time costs on four data sets are decreasing when the computing nodes number is increased from 4 to 8. The more computing nodes in the cluster means more computing power and higher parallel concurrency.

6 Conclusion

Time series motif is the repetitive similar patterns in time series. In this paper, we introduce a parallel algorithm to discover the time series motifs with variable-length. This algorithm can process large-scale time series in an acceptable time. Experimental results demonstrate that our algorithm can efficiently and precisely find motifs in large-scale time series. In the future, we will improve our method to find motifs from multivariate time series.

Acknowledgment. This work was supported by the project of Natural Science Foundation of China (No. 61402329, No. 61972456), the Natural Science Foundation of Tianjin (No. 19JCYBJC15400) and Natural Science Foundation of Tianjin-Science and Technology Correspondent Project (No. 18JCTPJC63300).

References

1. Bugenhagen, S.M., Cowley Jr., A.W., Beard, D.A.: Identifying physiological origins of baroreflex dysfunction in salt-sensitive hypertension in the Dahl SS rat. Physiol. Genomics **42**, 23–41 (2010)

2. Castro, N., Azevedo, P.J.: Multiresolution motif discovery in time series. In: SIAM, pp. 665–676 (2010)
3. Gao, Y., Lin, J.: Efficient discovery of variable-length time series motifs with large length range in million scale time series. CoRR abs/1802.04883 (2018)
4. Gao, Y., Lin, J., Rangwala, H.: Iterative grammar-based framework for discovering variable-length time series motifs. In: ICMLA, pp. 7–12 (2016)
5. Li, Y., U, L.H., Yiu, M.L., Gong, Z.: Quick-motif: an efficient and scalable framework for exact motif discovery. In: ICDE. pp. 579–590 (2015)
6. Lin, J., Keogh, E., Li, W., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Discov. **15**, 107–144 (2007). https://doi.org/10.1007/s10618-007-0064-z
7. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proceedings of 2nd Workshop on Temporal Data Mining at KDD, pp. 53–68 (2002)
8. Mueen, A., Hamooni, H., Estrada, T.: Time series join on subsequence correlation. In: ICDM, pp. 450–459 (2014)
9. Mueen, A.: Enumeration of time series motifs of all lengths. In: ICDM, pp. 547–556 (2013)
10. Mueen, A., Keogh, E.J., Zhu, Q., Cash, S., Westover, M.B.: Exact discovery of time series motifs. In: SIAM, pp. 473–484 (2009)
11. Nunthanid, P., Niennattrakul, V., Ratanamahatana, C.A.: Discovery of variable length time series motif. In: EEE, pp. 472–475 (2011)
12. Rebbapragada, U., Protopapas, P., Brodley, C.E., Alcock, C.: Finding anomalous periodic time series. Mach. Learn. **74**, 281–313 (2009). https://doi.org/10.1007/s10994-008-5093-3
13. Rong, C., Chen, L., Silva, Y.N.: Parallel time series join using spark. Concurr. Comput. Pract. Exp. **32**(9), e5622 (2020)
14. Senin, P., et al.: GrammarViz 2.0: a tool for grammar-based pattern discovery in time series. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD. LNCS, vol. 8726, pp. 468–472. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44845-8_37
15. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time-series motif from multi-dimensional data based on MDL principle. Mach. Learn. **58**, 269–300 (2005). https://doi.org/10.1007/s10994-005-5829-2
16. Yeh, C.C.M., Yan, Z., Ulanova, L., Begum, N., Keogh, E.: Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: ICDM, pp. 1317–1322 (2016)
17. Zhu, Y., Zimmerman, Z., Senobari, N.S., et al.: Matrix profile II: exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: ICDM, pp. 739–748 (2016)