

Accelerating Time Series Shapelets Discovery with Key Points

Zhenguo Zhang^{1,2}, Haiwei Zhang^{1(✉)}, Yanlong Wen¹, and Xiaojie Yuan¹

¹ College of Computer and Control Engineering, Nankai University,
38 Tongyan Road, Tianjin 300350, People's Republic of China
{zhangzhenguo,zhanghaiwei,wenyanlong,yuanxiaojie}@dbis.nankai.edu.cn

² Department of Computer Science and Technology, Yanbian University,
977 Gongyuan Road, Yanji 133002, People's Republic of China

Abstract. Shapelets are discriminative subsequences in a time series dataset, which provide good interpretability for time series classification results. For this reason, time series shapelets have attracted great interest in time series data mining community. Although time series shapelets have satisfactory performance on many time series datasets, how to fast discover them is still a challenge because any subsequence in a time series may be a shapelet candidate. There are several methods to speed up shapelets discovery in recent years. However, these methods are still time-consuming when dealing with the large datasets or long time series. In this paper, we propose a preprocessing step with time series key points for shapelets discovery which make full use of the prior knowledge of shapelets. Combining with shapelets discovery method based on SAX(Fast-Shaplets), we can find shapelets quickly on all benchmark datasets of UCR archives, while the classification accuracy is almost the same as the current methods.

Keywords: Time series · Shapelets · Classification · Key points

1 Introduction

Time series data mining has attracted significant interest in recent years because the series data are common in a wide range of daily life such as finance, medical treatment, motion, meteorology, etc. As a fundamental research work, classification of time series has been studied most commonly. Many research focus on distance measures for 1-Nearest Neighbor (1-NN) classifiers [1, 2]. Although the evidence show that 1-NN classifier with Euclidean Distance (ED) or Dynamic Time Warping (DTW) has good classification accuracy, it requires storing and searching the entire dataset which have high time and space complexity. Moreover, 1-NN classifiers do not give a clear insight to exhibit the most important patterns between two different classes. To solve these problems, a shapelet-based time series classification algorithm is proposed by *Ye et al.* [3].

A shapelet is a time series subsequence with highly discriminative between two classes. The shapelets discovery algorithm searches shapelets on raw data

and calculates the distance between shapelet candidates and time series. By the measure of information gain, the candidates are ordered and then the candidate with highest score is select as a shapelet to split dataset into two parts. For each part, the same process is used to get the corresponding shapelets. Shapelets are interpretable to the problem domain because they can tell us whether a shapelet is a common pattern in one class or not. The advantage has been confirmed by many researchers [4–6]. To some datasets, this algorithm can achieve more accurate classification results than other methods [3,10]. It has been applied to many domains such as medical care [7], gesture recognition [8], electrical power demand [9]. However, shapelets discovery is very time-consuming. All the subsequences with any length in time series dataset are shapelet candidates. Although there are some pruning strategies to speed up shapelets discovery like admissible entropy pruning [3], intermediate result reuse [10] and reducing the distance calculation with SAX method (called Fast-Shapelets) [11], it still needs a lot of time when the dataset is large or time series is very long. For example, when testing all candidates in the dataset *Non-Invasive Fetal ECG Thorax* of UCR time series archives [13], the Fast-Shapelets algorithm still needs over 10 h to get the results.

In this paper, we analyze the nature of time series shapelets and propose a preprocessing stage before searching shapelets to reduce the shapelet candidates. In our algorithm, the shapelet candidates are not all time series subsequences but the subsequences generated from the key points. Combining with the SAX representation of time series to find potential shapelet candidates, we can get the shapelets in less time than other algorithms. The experiments show that our preprocessing method for decreasing shapelet candidates is useful. In general, we make two contributions.

1. We propose a method to find the key points of a time series which is used for searching shapelet candidates. It can fast exclude the obviously impossible candidates from all subsequences.
2. An algorithm of extracting time series subsequences based on key points is presented that allows to generate shapelet candidates without repetition.

The rest of this paper is organized as follows. In Sect. 2, we review the related works about time series shapelets discovery algorithm. We present some notations and definitions of time series in Sect. 3. Section 4 shows our algorithm in details. We demonstrate the performance of our algorithm in Sect. 5 and this work is concluded in Sect. 6.

2 Related Work

The straightforward way for finding shapelets is the brute force algorithm that generates all possible candidates and tests these candidates by information gain. The running time is $O(n^2m^4)$, where n is the number of time series in dataset D , m is the length of a time series. In the first paper of shapelets discovery, the

author proposes several methods to speed up the searching work, like early abandon pruning and entropy pruning technique. However, it is still time-consuming. An improved algorithm called Logical Shapelets is given by [10]. One technique is that it precomputes the sufficient statistics to compute the distance between a shapelet and a candidate of a time series in amortized constant time. It is a way of trading space for time. The other is to use a novel admissible pruning technique to skip the costly computation of entropy for the vast majority of candidates. The worst-case of running time is $O(n^2m^3)$ and it requires a lot of memory space as large as $O(nm^2)$.

Rakthanmanon et al. [11] propose a way to find shapelets quickly, where the raw real-valued and high dimensional data are transformed to a discrete and low dimensional representation by using Symbolic Aggregate approxIimation (SAX) method [14]. In the discrete representation, they first hash the SAX representation of candidates by the random projecting method and then use the collision history to give a rough selection for the overall candidate set. By this way, the candidates that cannot be a shapelet are excluded quickly. The remaining candidates are still confirmed by information gain. The time complexity can reduce to $O(nm^2)$ according to the paper.

3 Notations and Definitions

3.1 Time Series

A time series T is a sequence of m real-valued variables recorded in temporal order at fixed intervals of time: $T = (t_1, \dots, t_i, \dots, t_m)$, $t_i \in R$. For the problem of time series data mining, a dataset of n time series can be expressed as $D = \{T_1, T_2, \dots, T_n\}$. For classification, each time series in the dataset D has a class label of c . Given a time series T of length m , a subsequence S is a series of length l ($l < m$) consisting of contiguous time instants of T : $S = (t_i, t_{i+1}, \dots, t_{i+l-1})$, where $i \in [1, m - l + 1]$, noted as T_i^l .

3.2 Time Series Key Points

We define the key points of a time series as the *inflection points*, *local minimum points*, *local maximum points* in a time series curve. They are useful for shapelets discovery and we will discuss them in detail in Sect. 4.

3.3 Distance Measure

In this paper, we take Euclidean distance as the distance measure between two time series. Suppose S and S' are two time series subsequences with the same length l , the Euclidean distance is calculated by the following formula:

$$d(S, S') = \sqrt{\frac{1}{l} \sum_{i=1}^l (s_i - s'_i)^2} \quad (1)$$

The distance between a time series T of length m and a subsequence S of length l ($l < m$) is defined as the minimum distance between the S and all subsequences of T that have the same length with S , i.e.

$$d(S, T) = \min d(S, T^l) \quad (2)$$

3.4 Time Series Shapelets

As primitives, shapelets can be used to determine the similarity of two time series. Because any subsequence in all time series of a dataset are probable to be a shapelet, finding a shapelet requires us to generate the candidates and to calculate the distances between candidates and a time series. We should also define a measurement to estimate the quality of a shapelet.

Shapelet Candidates. Given a dataset D , a shapelet candidate is the subsequence of length l in a time series. If dataset D contains n time series and the length of a time series is m , we can get $(m - l) + 1$ distinct subsequences in a time series and $n(m - l + 1)$ subsequences in D . We denote the set of all subsequences of length l to be $M_l : \{M_{1,l}, \dots, M_{i,l}, \dots, M_{n,l}\}$, where, $M_{i,l}$ is the subsequences of the i -th time series in D . The length l can be changed from 1 to m , so the overall subsequences set is: $M = \{M_1, \dots, M_l, \dots, M_m\}$.

Generally, if the length of a subsequence is too small or close to m , it may not be a shapelet. So we can give a minimum and maximum length before calculating the distances: $minlen$ and $maxlen$, i.e. $M = \{M_{minlen}, \dots, M_l, \dots, M_{maxlen}\}$. Note that M is very large. The most of shapelet research focus on how to efficiently prune M [3, 10, 11].

The Quality of a Shapelet. An effective measurement of discriminating the quality of a shapelet is information gain [3]. It involves the concept of *entropy* of a dataset and a *split*. Suppose that the dataset D contains c different classes, the number of time series in class i is n_i , so the entropy of the dataset is $E(D) = -\sum_{i=1}^c p_i \log(p_i)$, where $p_i = n_i/n$. A *split* is a tuple $\langle s, d \rangle$ of a subsequence s and distance threshold d which can separate the dataset into two subsets, D_1 and D_2 with n_1 and n_2 time series, respectively. The information gain of a given split sp in a dataset can be express as

$$IG(sp) = E(D) - \frac{n_1}{n} E(D_1) - \frac{n_2}{n} E(D_2) \quad (3)$$

It is possible that two splits have the same information gain. To solve this problem, the distance between two subsets divided by the given split called a separation gap is used as a measurement.

$$gap(sp) = \frac{1}{n_1} \sum_{t_1 \in D_1} d(s, t_1) - \frac{1}{n_2} \sum_{t_2 \in D_2} d(s, t_2) \quad (4)$$

So the quality of a shapelet candidate is measured by the value of information gain and separation gap. The large information gain and separation gap value indicate high discriminatory power of a shapelet candidate.

4 Shapelets Discovery with Key Points

4.1 A Motivating Observation

Note that a shapelet is a subsequence that can discriminate time series from different classes. So if a subsequence of a time series with no variations in its values or gradients is considered as a shapelet, the corresponding subsequence in another time series from a different class must have some variations in values or gradients. Generally, these variations correspond to some actions or movements of observation, which can be regarded as features. Therefore, the subsequence with variations is a more reasonable choice as a shapelet than subsequence with no variations. The results of previous work about shapelets discovery demonstrate that all of extracted shapelets contain some variations. Figure 1 shows some examples from [3].



Fig. 1. (*left*) Coffee dataset and its shapelet; (*right*) Gun point dataset and its shapelet

From Fig. 1, we find that all shapelets contain one or some of key points in time series. This character can be used to prune unnecessary candidates when searching shapelets.

4.2 Extracting Key Points

The key points of a time series are special points in a time series curve, so we can use mathematical methods to find them. Due to the small changes resulting from the process of data collection, a smooth step is necessary to eliminate these meaningless changes before we extract the key points from a time series. We use a rectangle sliding window along with a time series to do this preprocessing. Figure 2 gives a simple sketch of this method.

The width of rectangle sliding window is preset according to the specific situation. A large width means a rough sketch which only find the obvious changes while a small width means that the tiny changes will be found. For a given width, we scan the points along with time series. If the point is not in the rectangle sliding window, we pause the scan process and fit a straight line using the points

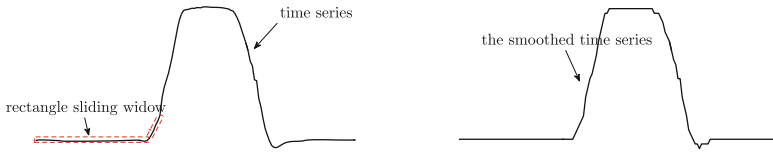


Fig. 2. Time series smoothing. (*left*) Rectangle sliding window; (*right*) The smoothed time series

Algorithm 1. Extract_KeyPoints

Input: T : a time series; th : gradient threshold

Output: $keypoints$: the key points in a time series

```

1:  $keypoints = []$ 
2:  $T = TS\_smooth(T)$ 
3: for each point in  $T$  do
4:    $gradient\_1 = Calculate\_Gradient(t)$ 
5:   for each value in  $gradient\_1$  do
6:     if  $value > th$  then
7:        $keypoints \leftarrow$  the index of value
8:     end if
9:   end for
10:   $g\_sign\_1 = sign(gradient\_1)$ 
11:   $gradient\_2 = Calculate\_Gradient(g\_sign\_1)$ 
12:   $g\_sign\_2 = sign(gradient\_2)$ 
13:   $index = find(abs(g\_sign\_2) == 2) + 1$ 
14:   $keypoints \leftarrow index$ 
15: end for

```

in the sliding window. Then, another sliding window is used to do the same thing until all points of the time series is smoothed. After finishing the smoothing step, we can extract the key points from a time series.

The Algorithm 1 shows the process of extracting key points. Line 2 executes the smooth process that is a key part for finding the key points. Line 4 keeps the variation information of a smoothed time series. The large variation points are stored in lines 6 and 7. The local minimum and maximum points are found from lines 10 to 14. Figure 3 shows the key points of a time series.

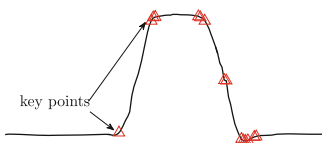


Fig. 3. Key points of a time series

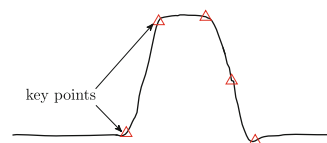


Fig. 4. Key points after removing unnecessary points

As we can see, there are too many key points after finishing the previous step and some key points are close enough. It is unnecessary to keep all points because the points that is close to each other will generate overlapped subsequences. So we search each point’s neighborhood and remove the other points. By an appropriate neighborhood threshold, we can get the reasonable key points. The result of the above example is expressed in Fig. 4 with a threshold of 10 points.

4.3 Finding the Best Candidates in Key Points Neighborhood

Generating Subsequences in Key Points Neighborhood. A shapelet can be of any length, so all the subsequences that contain a key point should be generated and measured by information gain. In order to avoid subsequences with overlapped parts, we check all key points in the current key point’s neighborhood. Suppose that the current key point is p , the length of subsequences to be generated is len , there are two cases when we get shapelets candidates (Fig. 5).

Case 1: If there are no other key points in the area of $[p-len, p]$, the candidates’ starting points are from point $p-len$ to p and we can get len candidates.

Case 2: If there are some key points in the area of $[p-len, p]$, the candidates’ starting points are from point k to p and we can get $p-k$ candidates. Note that k is the closest point to p .

Using SAX Method to Evaluate the Candidates. If we first calculate the distance of each candidate and all time series in a dataset and then find a split for computing the information gain value, it is very time consuming. *Rakthanmanon et al.* [11] give us a useful way to avoid unnecessary distance computation. Here, we use this method to evaluate the shapelet candidates generated by key points.

We first transform a subsequence to a SAX word. Here, the SAX process will not be discussed in details. Everyone who wants to know the process can refer to [14]. For all SAX words, a random projections method [11] is used to find the *topK* candidates with the most discriminating ability. The process is shown below.

In line 1, we define a matrix to record the masked SAX word occurred in time series. A random projection method is used to mask the SAX words in line 4. We do r times hashing for a SAX word and get the final matrix. In line 10, we

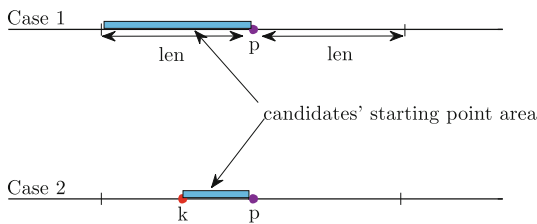


Fig. 5. A visualized representation of candidates’ starting point area

Algorithm 2. RandomProjections_and_CalculateBestSAX

Input: *SAXlist*: SAX representation of subsequences
k: the number of symbols when masking
r: the number of hashing

Output: *topK_candidates*: the *topK* discriminating candidates

- 1: *discriminate_matrix* = []
- 2: **for** each *saxword* in *SAXlist* **do**
- 3: **for** *i*=1 to *r* **do**
- 4: *sub_saxword* \leftarrow *RandomProjections*(*saxword*,*k*)
- 5: **for** all *saxword* contain *sub_saxword* **do**
- 6: *discriminate_matrix*[*saxword*][*ts_label*][*i*] $++$
- 7: **end for**
- 8: **end for**
- 9: **end for**
- 10: *topK_candidates* = *CalculateBestSAX*(*discriminate_matrix*)

use the function *CalculateBestSAX* to get the *topK* candidates by counting the occurrence number of a SAX word when we do hashing step in each class. The detailed explanation of *CalculateBestSAX* can be found in [11].

4.4 Shapelets Discovery Algorithm with Key Points

Our shapelet discovery algorithm with key points is presented in Algorithm 3.

The process contains two phase for the subsequence of length *len*. In the first phase (lines 4–15), we first extract all key points of a time series *ts* in line 5 and then keep the necessary points in line 6. In lines 7–9, all the shapelet candidates are generated based on key points. The discrete representation of candidates is gained in lines 11–14. When the first phase finished, the candidates of length *len* from all time series in *D* and their discrete representation are generated. In the second phase (lines 16–21), we used the random projection method to evaluate the quality of a candidates SAX representation. This method is proposed in [11, 12], which describe it in detail. We get *topK* candidates by this way in line 16. For the *topK* candidates, we still use the information gain to select the best candidate in lines 18–21. The reason is that the random masking method can effectively reduce the impossible candidates, but it is imprecise to find the best shapelets. After doing all admissible candidate length from *minlen* to *maxlen*, we can get the shapelets for the current dataset *D*.

5 Experimental Evaluation

We will demonstrate two points with our experiments. One is that our shapelets discovery techniques is useful to find shapelets for time series classification. The other is to show that our algorithm is faster than the other algorithms when discovering shapelets. We use the datasets from the UCR Time Series archives [13] to do our experiments. The UCR Time Series archives contains 85 datasets

Algorithm 3. Shapelet_with_Keypoints**Input:** D : time series Dataset; $maxlen, minlen$: max and min length of shapelet**Output:** sh : shapelet

```

1:  $sh \leftarrow []$ 
2:  $best\_gain = 0$ 
3: for  $len$  from  $minlen$  to  $maxlen$  do
4:   for each  $ts$  in  $D$  do
5:      $keypoints = Exact\_KeyPoints(ts)$ 
6:      $keypoints = Remove\_Closed\_Point(keypoints)$ 
7:     for each  $p$  in  $keypoints$  do
8:        $subseries = Generate\_Subseries(ts, p, len)$ 
9:     end for
10:     $saxlist \leftarrow []$ 
11:    for each  $s$  in  $subseries$  do
12:       $sax = Generate\_SAX(s, len)$ 
13:       $saxlist \leftarrow sax$ 
14:    end for
15:  end for
16:   $topK\_candidates = RandomProjections\_and\_CalculateBestSAX(saxlist)$ 
17:   $t\_series = Remap\_to\_ts(topK\_candidates)$ 
18:   $[gain, shapelet] = Calculate\_Gain(t\_series)$ 
19:  if  $gain > best\_gain$  then
20:     $best\_gain = gain \ \& \ sh \leftarrow shapelet$ 
21:  end if
22: end for

```

and provides diverse characteristics with various lengths and number of the classes, numbers of time series instances. We also use a decision tree classifier to calculate the classification accuracy by the selected shapelets as the other works do. In the experiments, the width of rectangle sliding window is set to $(max(T) - min(T))/50$, the cardinality of SAX is 4 and the size of key points' neighbourhood is 10.

5.1 Effectiveness of Key Points

Compared with Logical Shapelets. To validate the effectiveness of shapelets generated by key points, we reproduce the paper's [10] algorithm called Logical Shapelets (LS) which is known as the best algorithm for finding time series shapelets and modify it with key points. Because the Logical Shapelets algorithm needs a lot of time to get the results in large datasets (over one day), we compare the classification accuracy between two algorithms using 33 small datasets from the UCR Time Series archives. Figure 6 shows the results.

The points in Fig. 6 are classification accuracy ratios of two algorithms on the same datasets. The horizontal axis is the accuracy by using shapelets from key words and the vertical axis is Logical Shapelets' results. The line in the figure stands for that two algorithms have the same classification accuracy on a

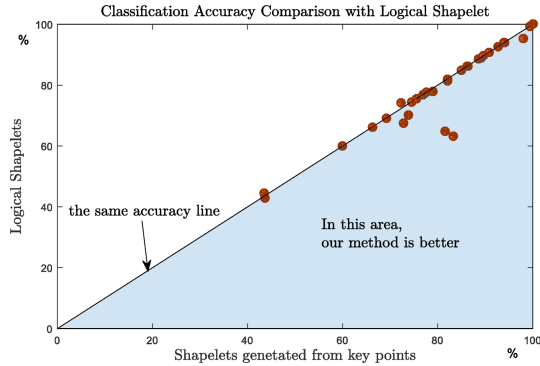


Fig. 6. Classification accuracy of LS and our modified LS by key points

dataset. The lower area under the line states that our algorithm is better. From the results, we can figure out that the shapelets selected by our algorithm is effective to build a classifier. In most datasets, we get the same classification accuracy with Logical Shapelets algorithm. In several datasets, we get better results. Note that this experiment is used to measure the quality of the selected shapelets with key points, it shows that our algorithm does not miss the eligible shapelets although we overlook a lot of candidates when generating the shapelet candidates. So our time series shapelets discovery algorithm with key points is feasible.

Compared with Fast Shapelets. Fast Shapelets (FS) [11] use a symbolic technique to fast prune unnecessary candidates which can speed up the shapelets discovery process and its accuracy is not perceptibly different with Logical Shapelets. To verify the effectiveness of our algorithm in large datasets, we compare our algorithm with Fast Shapelets with the same parameters in all datasets of UCR Time Series archives. The parameters including the number of random projections and the size of the set of potential candidates are referred to [11]. The classification results are shown in Fig. 7. As well as the above experiments, we also take the classification accuracy as axes.

We draw two dot lines on the same accuracy line side. The points between the two lines mean that classification results of two algorithms have little difference. We can find that two algorithms have not quite different difference in most of datasets. In almost half datasets (41 of 85), we get better results than FS algorithm. This experiments demonstrate that our shapelets discovery method with key points is effective even if we use the discrete representation of time series to find shapelets.

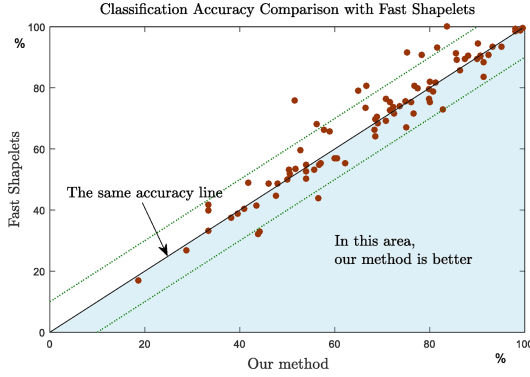


Fig. 7. Classification accuracy of our method and Fast Shapelets on 85 benchmark datasets

5.2 Running-Time Comparison

For all shapelets discovery algorithm, the main factors of finding shapelets are the number of time series in the dataset and the length of time series. To test the time efficiency of our method, we do the experiments on a large time series dataset in UCR time series archives, the *NonInvasiveFetalECGThorax1* dataset and compare our results with FS algorithm. It contains 42 classes, 1800 training time series of length 750 and 1965 test time series. In this experiments, the parameters are same for two algorithm: the ratio of random projections is set to 0.25 for a SAX word and the size of the set of potential candidates is set to 10, *i.e.* *topK* in Algorithms 2 and 3 is 10. Due to the long run-time (over 10 h) of FS algorithm when searching all candidates in this dataset, we take 10 points interval when two adjacent candidates are generated as the FS algorithm do. This way don't reduce the performance in finding shapelets [11].

Figure 8 shows the running time changes when the number of time series is varied from 100 to 1800 with a fixed length of 750 for all time series. From the figure, we can find that the running time of FS algorithm increases from 203 s to over 4600 s while our algorithm only need 40 s when the number of time series is 100 and less than 1056 s when the number is 1800. So the speedup factor is 4X-5X in this dataset.

Figure 9 shows that time consumption of our algorithm and the FS algorithm when the length of time series increasing. In our experiments, we fix the number of time series at 1800 and change their length from 50 to 750. The running time of our algorithm is varied form 14 s to 1033 s while the FS algorithm need 26 s at the beginning and over 4600 s when the length is 750.

These results are not surprising because our algorithm decreases the number of shapelet candidates with key points. For all dataset of UCR Time Series archives, the average speedup factor is 3.5 and largest factor is 9.8. The time complexity of our algorithm is the same as the FS algorithm. But given that our

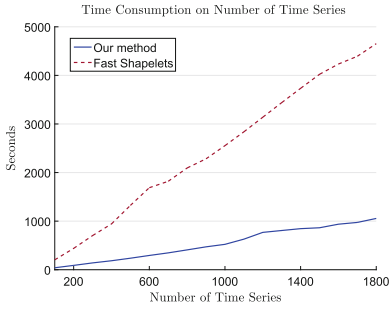


Fig. 8. Time consumption with number of time series

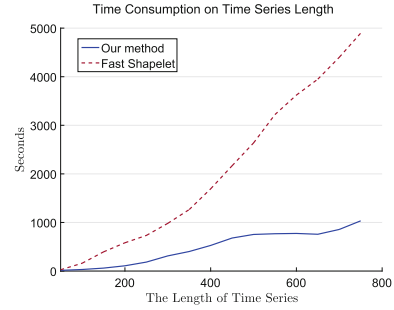


Fig. 9. Time consumption with the time series length

strategy is only to generate necessary shapelet candidates not all candidates, our algorithm is faster for finding shapelets.

6 Conclusions

For most time series, there are many subsequences with no or little changes. When discriminating two time series, most algorithm tend to use the varying subsequences than immutable subsequences. In this paper, we analyze the essential properties of time series shapelets and propose a preprocessing step to speed up the process of shapelets discovery. By using the key points, we decrease the number of shapelet candidates before measuring their quality as a shapelet. We have demonstrated that the subsequences generated by the key points do not miss eligible shapelets in the time series classification experiments compared with the current algorithm. Moreover, our algorithm for finding shapelets is significantly faster in almost all time series dataset of UCR archives than the current algorithm.

Acknowledgments. This work is partially supported by National 863 Program of China under Grant No. 2015AA015401, Tianjin Municipal Science and Technology Commission under Grant No. 14JCQNJC00200, 13ZCZDGX01098, as well as Research Foundation of Ministry of Education and China Mobile Under Grant No. MCM20150507. This work is also partially supported by Jilin NSF Under Grant No. 20130101179JC-18 and YBU development plan 2014–16.

References

1. Ding, H., Trajcevski, G., Scheuermann, P., et al.: Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endowment* **1**(2), 1542–1552 (2008)
2. Xi, X., Keogh, E., Shelton, C., et al.: Fast time series classification using numerosity reduction. In: *Proceedings of the 23rd International Conference on Machine Learning*, pp. 1033–1040. ACM, New York (2006)

3. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 947–956. ACM, New York (2009)
4. Chang, K.W., Deka, B., Hwu, W.M.W., Roth, D.: Efficient pattern-based time series classification on GPU. In: 12th International Conference on Data Mining, pp. 131–140. IEEE Computer Society, Washington DC (2012)
5. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 289–297. ACM, New York (2012)
6. Hartmann, B., Schwab, I., Link, N.: Prototype optimization for temporarily and spatially distorted time series. In: AAAI Spring Symposium: It's All in the Timing (2010)
7. Reiss, A., Weber, M., Stricker, D.: Exploring and extending the boundaries of physical activity recognition. In: 2011 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 46–50. IEEE Press, New York (2011)
8. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: accelerometer-based personalized gesture recognition and its applications. *Pervasive Mob. Comput.* **5**(6), 657–675 (2009)
9. Gordon, D., Hendler, D., Rokach, L.: Fast randomized model generation for shapelet-based time series classification. arXiv preprint [arXiv:1209.5038](https://arxiv.org/abs/1209.5038) (2012)
10. Mueen, A., Keogh, E., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1154–1162. ACM, New York (2011)
11. Rakthanmanon, T., Keogh, E.: Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the Thirteenth SIAM Conference on Data Mining (SDM), pp. 668–676. SIAM (2013)
12. Ulanova, L., Begum, N., Keogh, E.: Scalable clustering of time series with u-shapelets. In: SIAM International Conference on Data Mining (SDM 2015) (2015)
13. Chen, Y.P., Eamonn, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista G.: The UCR time series classification archive (2015). http://www.cs.ucr.edu/~eamonn/time_series_data/
14. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Disc.* **15**(2), 107–144 (2007)