EA_DTW: Early Abandon to Accelerate Exactly Warping Matching of Time Series

Junkui Li Yuanzhen Wang

College of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan 430074, China

Abstract

Dynamic Time Warping (DTW) is one of the important distance measures for time series, however, the exact calculation of DTW has become a bottleneck. We propose an approach, named *Early Abandon DTW* (EA_DTW) to accelerate the calculation. We demonstrate the idea of early abandon by theoretical analysis, and show the utilities of EA_DTW by thorough experiments both on synthetic and real datasets. The results show, EA_DTW outperforms the dynamic DTW calculation in the light of process time, and is much better when the threshold is below the real DTW distance.

Keywords: Data mining, Time series, Similarity search, Dynamic time warping, Early abandon

1. Introduction

Time-series data naturally occur in a wide range of applications, examples include computational biology, astrophysics, geology, multimedia, economics, to name a few. Therefore, there has been great research efforts devote to mining time series in the last decade. Similarity search in time series is useful in its own right to explore the properties of time series data. On one hand, similarity search is one of the important tools for mining time series; on the other hand, it usually acts as a subroutine in other time series mining tasks [1], and has wide applications in clustering, classification, pattern detection, and others. Recently, searching similar time series has been one of the hot topics in time series mining, and various similarity search methods have been proposed due to the ubiquitous use.

A large body of earlier work have been based on the Euclidean distance. The distance is calculated in a point by point manner, and works well when the time series have the same unit of scale. However, there is an increasing awareness that Euclidean distance suffers from the distortion in time axis and shows poor accuracy in searching [2].

Recently, more and more researchers examined the superiority of *Dynamic Time Warping*(DTW) over Euclidean distance in the similar time series search. DTW is a distance allowing stretching on time axis, which provides a way to optimally align time series that are intuitively more better than Euclidean distance does, thus since it was first introduced into the time series mining community by Berndt et al. in [3], the distance has attracted great attention. However, the calculation of DTW has long been a research topic, though there are some lower bounding functions to approximate the calculation of DTW, as we will show later, the exact DTW calculation is in reality unavoidable and has become the bottleneck in the similarity search, thus it is of urgent importance to accelerate the exact calculation of DTW.

Overall, our contributions in this work can be simply summarized as follows:

- We introduce the idea of early abandon in the calculation of DTW, The early abandon has been used in previous work for the calculation of Euclidean, and we apply it to the calculation of DTW.
- We propose the Early Abandon DTW (EA_DTW) to accelerate the calculation of DTW. We formally demonstrate the process of early abandon in DTW.
- We show the superiority of EA_DTW over plain dynamic DTW calculation by thorough empirical experiments, and the results validate the utility of EA_DTW.

The rest of the paper is organized as follows. Section 2 reviews the related work, and provides a background for our work. In Section 3 we present the method of EA_DTW in the exact calculation of DTW. In Section 4 we give experimental results and make discussions about the results. Finally we offer conclusions and future work in Section 5.

2. Related Work and Backgrounds

2.1. Related Work

A variety of work on DTW calculation mainly focused on the following aspects:

- Calculation of DTW with dynamic programming. The original calculation of DTW is a recursive routine, and may incur many redundant computations during the process. While the real calculation of DTW is usually based on the dynamic programming method, which constructs a dynamic warping matrix to reuse the already known values, and thus improves the efficiency.
- Exact indexing based on DTW distance. Since DTW is proved to not obey the property of triangle inequality [1], and cannot be used to the exactly indexing on time series sequences, some researchers introduced the lower bounding functions, such as *LB_Kim* [4], *LB_Keogh* [1], etc, and indexed the sequences with these functions. As the lower bounding functions are generally fast, they gain a good efficiency in large scale computations. The lower bounding theorem in [5] ensured that the index methods will not incur the *false negative*(i.e., the final resultset will contain all the qualified sequences, and no qualified sequence will be missed), however, the lowering functions may cause the *false positive*(i.e., the sequences returned by the index may not be the qualified sequences, and the unqualified sequences may be contained in the resultset). To get the exact results, the query on the index proceeds in two steps: the first step is to retrieval in the index space, and find all the candidate sequences; the second step is to refine the candidate sets of sequences from the first step by exact calculation of DTW, and discard all the unqualified sequences.

The work in [6] tested the indexing approaches with thorough experiments, and the results show, as there is a sequential scan in the second step(the exact calculation of DTW), the efficiency is very low, which has become a bottleneck in the query. Thus it is desirable to devise methods to accelerate the exact calculation of DTW.

2.2. Backgrounds

2.2.1. Some definitions

We are now in the position to give the formal definition on the problem we are considering, the similarity search. We first define the data type we are interested of, the time series.

Definition 1 *Time Series.* Time series is a sequence of data measured by the time, denoted as $T = \{t_1, t_2, \ldots, t_n\}$, where $n(n > 0)$ is the length of time series, i.e., $|T| = n$.

Definition 2 *Similarity Search.* Given the set of time series sequences $C = \{c_1, c_2, \ldots, c_p\}$, the query sequence q, the threshold $\epsilon(\epsilon > 0)$, and the distance measure d, find all the qualified sequences $c \in C$, such that $d(q, c) \leq \epsilon$.

There are two types of similarity searching in time series, one is the *whole sequence searching*, and the other is the *subsequence searching*, i.e., searching all occurrence of subsequences that are qualified. While it is possible to convert the subsequence searching into whole sequence searching by the methods of sliding window [7], segmentation, we consider whole sequence searching in this work.

2.2.2. Exact calculation of DTW

Suppose there are two sequences with length of m and n, respectively:

$$U = u_1, u_2, \ldots, u_m, \tag{1}$$

$$V = v_1, v_2, \ldots, v_n. \tag{2}$$

To align the two sequences with DTW, we can construct an $m * n$ dynamic warping matrix, where the cell $(i, j)(1 \leq i \leq m, 1 \leq j \leq n)$ corresponds to the aligning of data point u_i and v_j, and the value in the cell is the distance between u_i and v_J, also called *base distance.* In accordance with other work [1], we use the Euclidean distance as the base distance, i.e.,

$$d_{base}(u_i, v_j) = (u_i - v_j)^2. \tag{3}$$

However, other distance measures such as $L1$ will not affect our discussions.

After constructing the dynamic warping matrix, we can calculate a warping path W from the cell $(1, 1)$, and to cell (m, n):

$$W = w_1, w_2, \ldots, w_c, \tag{4}$$

as shown in Fig. 1, and the DTW alignment between U and V is shown in Fig. 2.

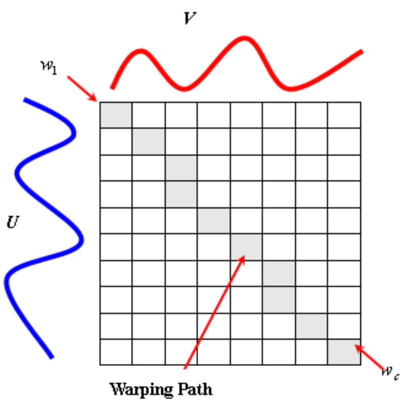

Fig. 1: Illustration of classic DTW calculation: construct a warping path from the start point to the end point, and search for the minimal warping path, denoted with the grey squares.

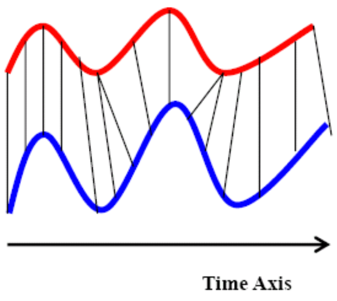

Fig. 2: The DTW alignment between the two sequences.

We can formally define a mapping function $f_w : (U,V) \rightarrow W$ to represent the alignment between U and V in the warping path W as

$$w_k = f_w(u_i, v_j), \tag{5}$$

where $1 \leq k \leq c, i \in \{1,2,\dots,m\}, j \in \{1,2,\dots,n\}$, u_i and v_j are aligned, and the cell (u_i, v_j) is the kth cell w_k in the warping path.

In the calculation of DTW, the warping path should subject to several constraints:

- **Endpoint.**The warping path starts from the beginning of both sequences, and ends to the end of both sequences, i.e.,

$$w_1 = f_w(u_1, v_1), w_c = f_w(u_m, v_n). \tag{6}$$

- **Continuity.** Neighboring cells in the warping path are adjacent to each other(including the diagonally adjacent cells), i.e.,

$$\begin{cases} w_k = f_w(u_i, v_j) \\ w_{k+1} = f_w(u_{i'}, v_{j'}) \end{cases} \Rightarrow i' \leq i+1, j' \leq j+1. \tag{7}$$

- **Monotonicity.** The warping path spaces monotonically in time, i.e.,

$$\begin{cases} w_k = f_w(u_i, v_j) \\ w_{k+1} = f_w(u_{i'}, v_{j'}) \end{cases} \Rightarrow i \leq i', j \leq j'. \tag{8}$$

The DTW distance between U, V is calculated from the optimal warping path with the minimum distance:

$$DTW(U,V) = \underset{W}{argmin}(\sqrt{\sum_{i=1}^{c} f_w(u_i, v_j)}). \tag{9}$$

Though a variety of techniques can be applied to the calculation of DTW, the most established one is the dynamic programming method. The exact DTW distance can be calculated as:

$$DTW(U,V) = \gamma(m,n)$$

$$\gamma(i,j) = d_{base}(u_i, v_j) + min \begin{cases} \gamma(i-1, j) \\ \gamma(i-1, j-1) \\ \gamma(i, j-1) \end{cases} \tag{10}$$

$$\gamma(0,0) = 0, \gamma(0,\infty) = \infty, \gamma(\infty, 0) = \infty$$

$$(i = 1,2,\dots,m; j = 1,2,\dots,n)$$

the value in cell $\gamma(i,j)$ can be seen as the cumulative sum of values in the warping path from cell $(1,1)$ to (i,j), we call the cells with value of $\gamma(i,j)(1 \leq i \leq m, 1 \leq j \leq n)$ as *cumulative distance matrix.* Fig. 3 shows an example of cumulative distance matrix.

	0	3	6	0	6	1
2	**4**	**5**	21	25	41	42
5	29	8	**6**	31	26	42
2	33	9	22	**10**	26	27
5	58	13	10	35	**11**	27
3	67	13	19	19	20	**15**

Fig. 3: Illustration of cumulative distance matrix between two sequences: $U = \{2,5,2,5,3\}$,$V = \{0,3,6,0,6,1\}$. The DTW distance is 15.

2.2.3. Early Abandon Technique

Early abandon is a technique in the constraint distance calculation. Here constraint means the calculation is not for the exact distance, but for a comparison between distances(such as to determine one distance is bigger than the other). The similarity search of time series needs to check $d(q, c) \leq \epsilon$, which is a comparison and if the distance exceeds the ϵ, the candidate sequence c is not qualified, and will be excluded in the final resultset. With this in mind, if current calculation already exceeds the ϵ, we should terminate, since if we have $d(q', c') > \epsilon(|q'| < |q|, |c'| < |c|)$, then we will come to the conclusion that $d(q, c) > \epsilon$, the sequence c should not be qualified.

The work in [8] and [9] applied early abandon to eliminate redundant calculations of Euclidean distance between sequences, the idea is illustrated in Fig. 4. Since the calculation of Euclidean distance is a forward step-wised process, and once the calculation exceeds the threshold, the afterward calculations will be interrupted and terminated, and report no-match, thus save the computational power and improve the efficiency.

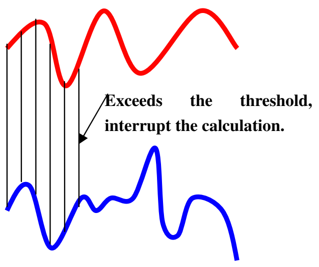

Fig. 4: Early abandon on Euclidean distance calculation.

3. Early Abandon on Exact DTW Calculation

The exact DTW distance can be calculated with equation (10) recursively, for two sequences of m and $n(m > 1, n > 1)$, respectively, we can construct an $m*n$ cumulative distance matrix, and the value $\gamma(m, n)$ is the DTW distance between the sequences. However, it should point out that we can apply the idea of early abandon to accelerate the calculation. For simplicity, for the given threshold $\epsilon > 0$, if the value of $\gamma(i, j) > \epsilon$, we call cell $\gamma(i, j)$ *overflows.*

3.1. Theoretical Analysis

Now we examine the properties of the cumulative distance matrix.

Theorem 1 ***The overflow transmission theorem.*** *If the cells $\gamma(i-1, j)$, $\gamma(i-1, j-1)$, $\gamma(i, j-1)$ in cumulative distance matrix overflow, then t he cell $\gamma(i, j)$ will overflow.*

Proof.

$$\because \gamma(i-1, j) > \epsilon, \gamma(i-1, j-1) > \epsilon, \gamma(i, j-1) > \epsilon,$$

$$\therefore min \begin{cases} \gamma(i-1, j) \\ \gamma(i-1, j-1) \\ \gamma(i, j-1) \end{cases} > \epsilon.$$

Moreover,

$$d_{base}(u_i, v_j) \geq 0.$$

Hence, from equation (10) we have:

$$\gamma(i, j) > \epsilon,$$

the cell $\gamma(i, j)$ overflows.

Theorem 2 ***The overflow omit calculation theorem.*** *If the cells $\gamma(i-1, j)$, $\gamma(i-1, j-1)$, $\gamma(i, j-1)$ in cumulative distance matrix overflow, then the calculation of base distance $d_{base}(u_i, v_j)$ can be omitted.*

Proof. From theorem 1, we have the cell $\gamma(i, j)$ will overflow without regarding what the value of $d_{base}(u_i, v_j)$ is, thus the value of $d_{base}(u_i, v_j)$ takes no effect on the overflow of $\gamma(i, j)$ and can be omitted.

From theorem 2, we know that if the cell $\gamma(i, j)$ overflows, we do not have to calculate the base distance for the cell, thus save the computation.

Theorem 3 ***The no transmission with replacement theorem.*** *If cell $\gamma(i, j)$ overflows, then replacing the value of cell $\gamma(i, j)$ with a value that is above ϵ will not affect the overflow of cell $\gamma(i', j')(i' \geq i, j' \geq j)$.*

Proof. We prove the theorem with two cases:

- the value $\gamma(i', j')$ depends on the value of $\gamma(i.j)$. As $\gamma(i, j) > \epsilon$, thus $\gamma(i', j') > \epsilon$ and if replace $\gamma(i, j)$ with a value $\epsilon^*(\epsilon^* > \epsilon)$, the $\gamma(i', j') > \epsilon$ holds.
- the value $\gamma(i', j')$ has no relation with the value of $\gamma(i, j)$. Surely in this case, the replacement will not affect the overflow of cell $\gamma(i', j')$.

Theorem 4 ***The replacement equality theorem.*** *If the cell $\gamma(i,j)$ overflows, then replacing the value of cell $\gamma(i,j)$ with a value that is above ϵ will not affect the overall overflow of the DTW distance.*

Proof. Similarly, we consider the proof with two cases:

- The minimal warping path crosses the cell (i,j). As $\gamma(i,j) > \epsilon$, then from the meaning of γ and the equation (9), we have $DTW(U,V) > \epsilon$, if we replace $\gamma(i,j)$ with an above ϵ value, the same conclusion holds.
- The minimal warping path does not cross the cell (i,j). The value in cell (i,j) will not affect the $DTW(U,V)$ and can be omitted. From the theorem 3, provided that $\gamma(i,j) > \epsilon$, the replacement will not affect the overflow of remaining cells.

Theorem 5 ***The early abandon on DTW theorem.*** *If cells in one row or one column of the cumulative distance matrix all overflow, then we have $DTW(U,V) > \epsilon$.*

Proof. From the constraints on the warping path, (6),(7) and (8), warping path is a continuous curve from cell $(1,1)$ to cell (m,n), thus warping path must cross each row and column in the distance matrix. If all cells in one row or column overflow, then there must be a cell in the cumulative distance matrix with the value exceeds the threshold, from the meaning of γ and the equation (9), we have $DTW(U,V) > \epsilon$.

3.2. Early Abandon on DTW

From the above analysis, we know that if one cell overflows, we can replace the value of the cell with an on-the-fly value that is above the threshold and save the calculation of base distance without affecting the final result; furthermore, if the cells in one row or column overflow, we should terminate the calculation DTW, and return no-match. The idea of early abandon on DTW is illustrated in Table 3.2.

EA_DTW returns false if the two sequence are not matched, i.e, the distance is above the given threshold, and returns true if they are matched. In table 3.2, the algorithm EA_DTW constructs an $m*n$ matrix for the cumulative distance and initializes the margin of the matrix(in lines 6 to 8). Lines 3 to 5 check if the first point alignment exceeds the threshold, if so, we just cancel the calculation and return false. Lines 9 to 23 do the early abandon and calculation of DTW, before calculating the value of each cell, we first check if the value will overflow according to the theorem 1, if the cell will overflow, replace the cell value with the max possible double value according to theorem 3 and theorem 4, and omit the calculation of base distance according to theorem 2, otherwise calculate the value in normal way. Lines 19 to 22 check if the whole row overflow, and if so, break out the loop and terminate the further calculation according to theorem 5. The time complexity of EA_DTW is $O(mn)$, where m, n are the length of the two sequences, respectively.

```
Algorithm: EA_DTW
Input:
$U = \{u_0, u_1, \ldots, u_{m-1}\}$,
$V = \{v_0, v_1, \ldots, v_{n-1}\}$,
$\epsilon$: threshold for the query.
Output:
$true$ if $DTW(U,V) \le \epsilon$, otherwise $false$.

1. construct an $m*n$ matrix $M$;
2. $M[0][0] \leftarrow d_{base}(u_0, v_0)$;
3. if $M[0][0] > \epsilon$ then
4.    return $false$;
5. end if;
6. Initialize the $M[0][.]$ and $M[.][0]$ with
7. $M[i][0] \leftarrow M[i-1][0] + d_{base}(u_i, v_0)$
8. $M[0][i] \leftarrow M[0][i-1] + d_{base}(u_0, v_i)$
9. for $i = 1$ to $m-1$ do
10.   $overflow \leftarrow true$;
11.   for $j = 1$ to $n-1$ do
12.      $v \leftarrow min\begin{cases} M[i-1][j-1] \\ M[i-1][j] \\ M[i][j-1] \end{cases}$;
13.      if $v > \epsilon$ then
14.         $M[i][j] \leftarrow DOUBLE\_MAX$;
15.      else
16.         $M[i][j] \leftarrow v + d_{base}(u_i, v_j)$;
17.         $overflow \leftarrow false$;
18.      end if;
19.   end for;
29.   if $overflow$ then
21.      break;
22.   end if;
23.end for;
24.if $overflow$ then
25.   return $false$;
26.end if;
27.return $M[m-1][n-1] \le \epsilon$;
```

Table 1: Early Abandon on DTW

4. Experimental Study

In this section, we test the improvement of EA_DTW on the efficiency with a comprehensive sets of experiments.

4.1. Experiment Setup

For these experiments, we used a PC and the configuration is listed in Table 2.

Configuration item	Item value
Processor	Intel Pentium 3-866
Operating System	Ubuntu Linux 4.1.1-13
RAM	256 megabytes
Disk space	40 gigabytes
Implement language	ANSI C
Compiler	GNU gcc 4.1.2 20060928

Table 2: Experiment configuration

Note that in order to allow reproducibility, all the source code and datasets are freely available, interested readers may email the authors.

For completeness, we implemented all the methods proposed in this work, which are the EA_DTW, calculation of DTW with dynamic programming(denoted as *Dyn_DTW*), and raw calculation of DTW with recursive(denoted as *Raw_DTW*). There are a step for construction of cumulative distance matrix in EA_DTW and Dyn_DTW, while this is not a must in Raw_DTW, the method calculates the DTW in a recursive way. We have taken great care to create high quality implementations of all techniques. All approaches are optimized as much as possible.

In our experiments, we evaluated the efficiency of different techniques using two metrics. We measured the *elapsed time* as the performance metric directly perceived by the user, and the *skip_rate* as the a factor of saving computations.

- **Elapsed Time.** We used wall-clock time to measure the elapsed time during the calculation. As we repeated each experiments in several times, the resulting elapsed time is the average of the experiments with the same parameters configured.
- **Skip Rate.** The steps the EA_DTW skipped is a factor for the improvement, we use the following equation to denote the skip rate during the early abandon:

$$skip\_rate = 1 - \frac{calculated\_cells\_number}{total\_cells\_number} \tag{11}$$

4.2. Evaluation on Random Generated Datasets

The data sets used in this experiment were created using a random time series generator that produces n time series that confirm to the Simple, Uniform, Exponent Distributions. A fraction of the data are shown in Fig. 5.

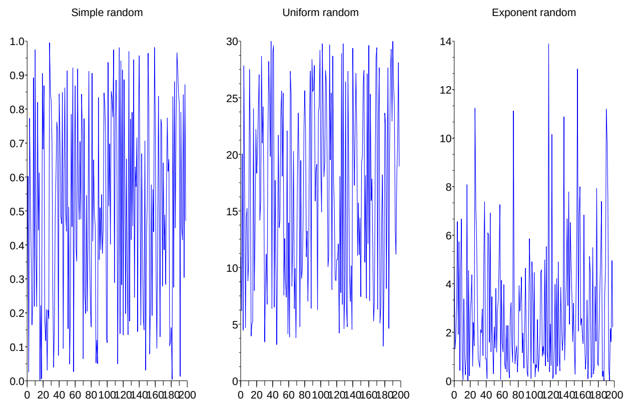

Fig. 5: Plotting a fraction of the random data.

We found that there were many redundant calculations in RAW_DTW compared with Dyn_DTW, and RAW_DTW showed a poor efficiency, the results are shown in Table 4.2. For this reason, we mainly compare our method, EA_DTW, with the method Dyn_DTW for the rest of the experiments.

RAW_DTW	Dyn_DTW	
Simple random	7,418,316.0	141.8
Uniform random	3,929,671.3	140.9
Exponent random	251,528.6	130.3

Table 3: The comparison of RAW_DTW and Dyn_DTW on elapsed time (in ms).

The comparisons of the elapsed time on the random generated data are shown in Fig. 6, Fig. 7 and Fig. 8, respectively. As seen from the results, the EA_DTW outperforms the dynamic calculation of DTW in the elapsed time, the method wins in all the threshold configurations, even when the threshold exceeds the exact distance.

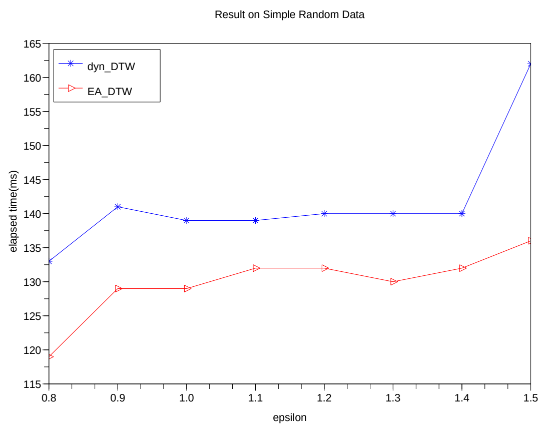

Fig. 6: Comparison of elapsed time on Simple Random data, the exact DTW distance is 1.3, the size of cumulative distance matrix is $10 * 13$.

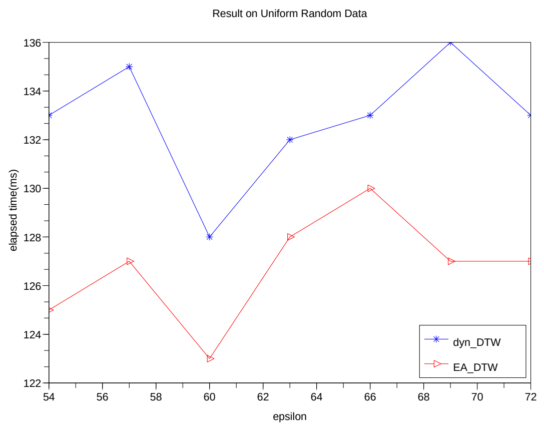

Fig. 7: Comparison of elapsed time on Uniform Random data, the exact DTW distance is 66.4, the size of cumulative distance matrix is $8 * 11$.

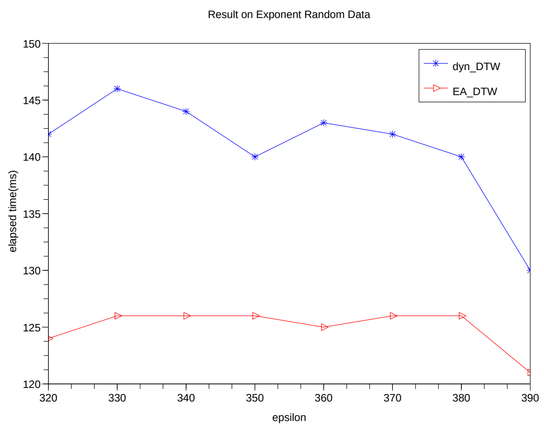

Fig. 8: Comparison of elapsed time on Exponent Random data, the exact DTW distance is 370.2, the size of cumulative distance matrix is $10 * 12$.

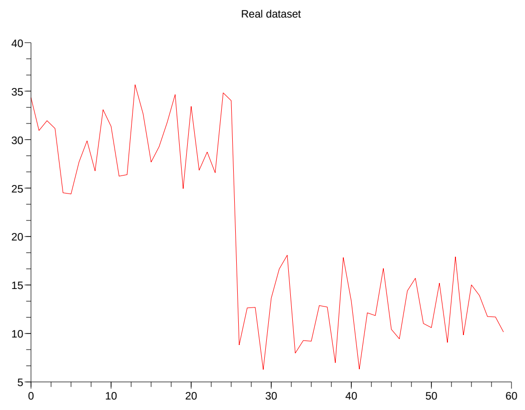

Fig. 9: Plotting a fraction of the real data.

4.3. Evaluation on Real Dataset

Our real dataset is the Synthetic Control, which was downloaded from UCI [10]. Fig. 9 shows typical examples of the dataset.

The result of elapsed time is shown in Fig. 10. The same trend is observed from the result. With the same parameter configured, the EA_DTW finishes the calculation in less elapsed time, the pruning power of the EA_DTW is satisfying.

With all these experiments, we also measured the skip rate for each calculation of EA_DTW, Fig. 11 shows the results. As the interesting results indicate, the skip rate becomes lower with the threshold grows, and when the threshold exceeds the exact distance, we may have less cells to skip. However, one thing should be noted that, even when the threshold is above the exact distance, we have some improvement on the elapsed time, as well as the skip rate, because we can skip the calculation of base distance to save computations.

5. Conclusions

In this work, we examined the problem of the exact calculation of DTW, and proposed a method called EA_DTW to accelerate the calculation. The approach adopted the idea of early abandon to skip the unnecessary calculations. The experiments show, EA_DTW works with less calculation time, compared with the dynamic programming method of DTW calculation, and the skip rate is more

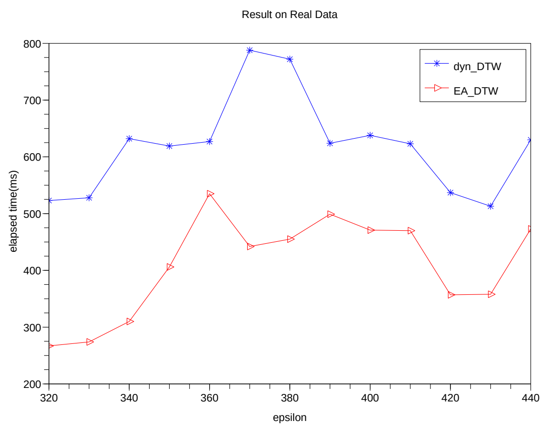

Fig. 10: Comparison of elapsed time on real data, the exact DTW distance is 413.1, the size of cumulative distance matrix is $60 * 60$.

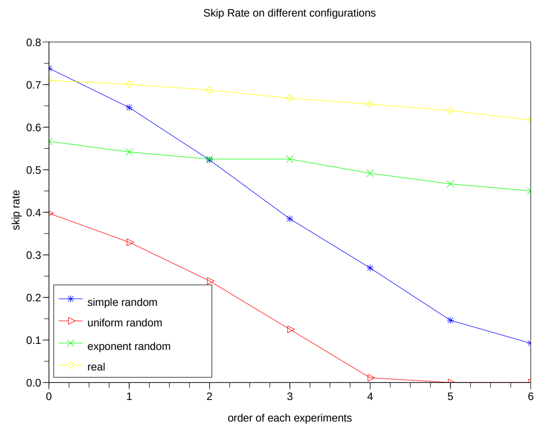

Fig. 11: Skip rate on different configurations: the skip rate of the first 7 epsilon configurations on each dataset

better when the threshold specified by the user is blow the exact distance. For future research, we plan to apply the method to the segmented-wised DTW calculation [11], where sequences are first segmented before the calculation of DTW, the difference with this work is that if we will skip one cell in the cumulative distance matrix, we may skip more cells in the same segment, since the values of cells in one segment are more the same.

References

[1] E. Keogh, Exact indexing of dynamic time warping. *Proc. of 28th International Conference on Very Large Data Bases*(VLDB), pp. 406–417, HongKong, China, 2002.

[2] E. Keogh, S. Kasetty, On the need for time series data mining benchmarks: a survey and empirical demonstration. *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 102–111, Edmonton,Canada, 2002.

[3] D. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series. *AAAI Workshop on Knowledge Discovery in Databases*, pp. 229–248, Washington DC, 1994.

[4] S. Kim, S. Park, W. Chu, An index-based approach for similarity search supporting time warping in large sequence databases. *Proc. of 17th International Conference on Data Engineering*(ICDE), pp. 607–614, Washington DC, 2001.

[5] C. Faloutsos, M. Ranganthan, Y. Manolopoulos, Fast subsequence matching in time-series databases. *Proc. of the ACM SIGMOD conference*, pp. 419–429, Mineapolis, 1994.

[6] S. Kim, B. Jeong, Performance bottleneck in time-series subsequence matching. *2005 ACM Sumposium on Applied Computing*, pp. 469–473, 2005.

[7] E. Keogh, M. Pazzani, A simple dimensionality reduction techinique for fast similarity search in large time series databases. *Proc. of the 4th Pacific-Asia conference on Dnowledge Discovery and Data mining*(PAKDD), pp. 122–133, Kyoto, Japan, 2000.

[8] W. Li, E. Keogh, H. Van et al, Atomic Wedgie: Efficient query filtering for streaming time series. *Proc. of the 5th IEEE International Conference on Data Mining*(ICDM), pp. 490–497, Houston, Texas, 2005.

[9] E. Keogh, W. Li, X. Xi, et al, LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. *Proc. of the 32nd International Conference on Very Large Data Bases*(VLDB), Seoul, Korea, 2006.

[10] http://kdd.ics.uci.edu/databases.

[11] M. Zhou and M. H. Wang, A segment-wise time warping method for time scaling searching. *Information Sciences*, pp. 237–253, 2004.