Early Abandon to Accelerate Exact Dynamic Time Warping

Li Junkui and Wang Yuanzhen
College of Computer Science and Technology, University of Science and Technology, China

Abstract: *Dynamic time warping is one of the important distance measures in similarity search of time series; however, the exact calculation of dynamic time warping has become a bottleneck. We propose an approach, named early abandon dynamic time warping, to accelerate the calculation. The method checks if values of the neighbouring cells in the cumulative distance matrix exceed the tolerance, and if so, it will terminate the calculation of the related cell. We demonstrate the idea of early abandon on dynamic time warping by theoretical analysis, and show the utilities of early abandon dynamic time warping by thorough empirical experiments performed both on synthetic datasets and real datasets. The results show, early abandon dynamic time warping outperforms the dynamic time warping calculation in the light of processing time, and is much better when the tolerance is below the real dynamic time warping distance.*

Keywords: *Data mining, time series, similarity search, dynamic time warping, early abandon.*

Received July 8, 2007; accepted December 20, 2007

1. Introduction

Time-series data naturally occur in a wide range of applications, examples include computational biology, astrophysics, geology, multimedia, and 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 [5], and has wide applications in classification [12], pattern detection [9], and others. Recently, searching for similar time series has been one of the hot topics in time series mining, and many similarity search methods have been proposed.

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 [1, 3, 4].

Recently, more and more researchers examined the superiority of Dynamic Time Warping (DTW) over Euclidean distance in searching similar time series. DTW is a distance allowing stretching on time axis, which provides a way to optimally align time series that are intuitively better than Euclidean distance does. Since it was first introduced into the time series mining community by Berndt *et al.* [1], 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. 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. The real calculation of DTW is usually based on the dynamic programming method, which constructs a distance matrix to reuse the already known values, and improves efficiency.
- Indexing based on DTW distance. Since DTW was proved to not obey the property of triangle inequality [3], and cannot be used to the exactly indexing time series sequences, some researchers introduced the lower bounding functions, such as LB_Kim [8], LB_Keogh [3, 4], LB_PAA [15], *etc,* and indexed sequences with these functions. As lower bounding functions are generally fast, they gain a good efficiency in large scale computations. The lower bounding theorem in [2] ensured that the index methods will not incur the false negative (i.e., the final result set will contain all the qualified sequences, and no qualified sequence will be missed), however, 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 result set). 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 a post processing step, which 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 [7] tested the indexing approaches on subsequence matching with thorough experiments, and the results show, the step of post processing has been the performance bottleneck in the query. Similarly, there is a sequential scan on the post posting step of previous proposed methods on querying by DTW based indexing, thus it is desirable to devise methods to accelerate the exact calculation of DTW in refining step. Overall, our contributions in this work can be simply summarized as follows:

- We introduce the idea of early abandon in the exact 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 exact calculation of DTW. We 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 provides a background for our work. In section 3 we present the method of EA_DTW for the exact calculation of DTW. Section 4 presents the experimental results. We discuss some related work in section 5. Finally we offer conclusions and future work in section 6.

2. Preliminary

In this section, we first give some definitions to confine our problem scope, and then discuss the exact dynamic time warping as well as the early abandon technique on Euclidean distance calculation.

2.1. Some Definitions

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

Definition 1: Time series: time series is a sequence of data measured by the time, denoted as $T = \{t_1, t_2, ..., t_n\}$, 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 $C = \{c_1, c_2, ..., c_p\}$, the query sequence q, the tolerance given by users $\varepsilon(\varepsilon > 0)$, the distance measure d, find all the qualified sequences $c \in C$, that $d(q,c) \le \varepsilon$.

There are two types of similarity search in time series, one is the whole sequence search, and the other is the subsequence search, i.e., searching all the occurrences of sub sequences that are qualified. While it is possible to convert the subsequence search into whole sequence search by the methods of sliding window [6], segmentation etc, we consider whole sequence search in this work.

2.2. Exact Calculation of DTW

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

$$U = \{u_1, u_2, ..., u_m\}, \quad (1)$$

$$V = \{v_1, v_2, ..., v_n\}. \quad (2)$$

To align the two sequences with DTW, we can construct an $m \times n$ distance matrix, where the cell $(i, j)(1 \le i \le m, 1 \le j \le n)$ 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*. Though there are many ways to define the base distance, we use the distance $d_{base}(u_i, v_j) = (u_i - v_j)^2$ as the base distance in accordance with other work [3]. However, other distance measures such as $L1$ ($d_{base}(u_i, v_j) = |u_i - v_j|$) will not affect our discussions.

After constructing the distance matrix, we calculate a warping path W from cell (1,1), and to cell (m,n): $W = \{w_1, w_2, ..., w_c\}(\max(m,n) \le c \le m + n - 1)$, as shown in Figure 1.

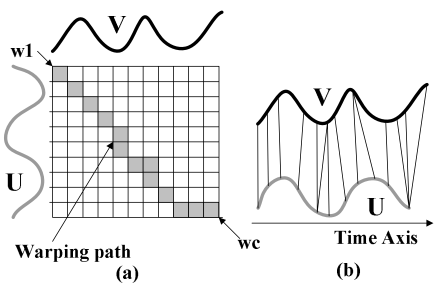


Figure 1. Illustration of classic DTW calculation (a) Construct warping matrix and search for the minimal warping path, denoted with grey squares; (b) The DTW alignments between sequences.

We can formally define a mapping function $f_w : (U, V) \to W$ to represent the alignment between U and V in the warping path W as: $w_k = f_w(u_i, v_j)$, where $1 \le k \le c$, $i \in \{1, 2, ..., m\}$, $j \in \{1, 2, ..., n\}$, 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 is considered to subject to several constraints:

- Endpoint: the warping path starts from the beginning of both sequences, and ends to the ending of both sequences, i.e.,

$$w_1 = f_w(u_1, v_1), w_c = f_w(u_m, v_n). \quad (3)$$

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

$$\begin{cases} w_k = f_w(u_i, v_j) \\ w_{k+1} = f_w(u_{i'}, v_{j'}) \end{cases} \Rightarrow i' \le i+1, j' \le j+1. \quad (4)$$

- Monotonicity: the warping path spaces monotonically in time, i.e.,

$$\begin{cases} w_k = f_w(u_i, v_j) \\ w_{k+1} = f_w(u_{i'}, v_{j'}) \end{cases} \Rightarrow i \le i', j \le j'. \quad (5)$$

The DTW distance between U, V is calculated from the optimal warping path with the minimum distance: $DTW(U,V) = \arg\min_W(\sum_{k=1}^{c} w_k)$. Though a variety of techniques can be applied to the calculation of DTW, the most established one is the dynamic programming method, which is calculated as follows:

$$\begin{aligned} & DTW(U,V) = \gamma(m,n), \\ & \gamma(i,j) = d_{base}(u_i, v_j) + \\ & \quad \min\{\gamma(i-1,j), \gamma(i-1,j-1), \gamma(i,j-1)\}. \quad (6) \\ & \gamma(0,0) = 0, \gamma(0,\infty) = \infty, \gamma(\infty,0) = \infty, \\ & (i = 1,2,...,m; j = 1,2,...,n). \end{aligned}$$

The value in cell $\gamma(i,j)$ is the cumulative sum of values in the warping path from cell $(1,1)$ to (i,j) .Given the warping path $W = \{w_1, w_2, ..., w_k\}$, the value in cell is $\gamma(i,j) = \sum_{i=1}^{k} w_i$. The matrix containing the cells $\gamma(i,j)(1 \le i \le m, 1 \le j \le n)$ is called cumulative distance matrix. Figure 2 shows an example of distance matrix and the corresponding cumulative distance matrix.

	0	3	6	0	6	1
2	4	1	16	4	16	1
5	25	4	1	25	1	16
2	4	1	16	4	16	1
5	25	4	1	25	1	16
3	9	0	9	9	9	4

(a)

	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

(b)

Figure 2. $U = \{2,5,2,5,3\}, V = \{0,3,6,0,6,1\}$. (a) The distance matrix of two sequences. (b) The cumulative distance matrix of two sequences. The DTW distance is 15.

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 $d(q,c) \le \varepsilon$, 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 result set. With this in mind, if current calculation already exceeds the ε, we should terminate, since if $d(q',c') > \varepsilon(|q'|<|q|, |c'|<|c|)$, and then we will come to the conclusion that $d(q,c) > \varepsilon$, the sequence c should not be qualified.

The work in [3], [5] applied early abandon to eliminate redundant calculations of Euclidean distance between sequences. The idea is illustrate in Figure 3.

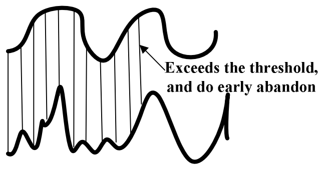


Figure 3. Early abandon on Euclidean distance calculation.

Since the calculation of Euclidean distance is a forward step-wised process, and once the calculation exceeds the threshold, the calculations unfinished will be interrupted and terminated, and report no-match, thus saving the computational power and improving the efficiency.

3. Early Abandon on Exact DTW Calculation

The exact DTW distance can be calculated with equation 6 recursively, for two sequences with length of m and $n (m > 1, n > 1)$, respectively, we can construct a $m \times n$ 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 $\varepsilon > 0$, if the value of $\gamma(i,j) > \varepsilon$, we call cell $\gamma(i,j)$ *overflows*.

3.1. Theoretical Analysis

Now we examine the properties of the cumulative distance matrix.

Lemma 1: (overflow transmission) If cells $\gamma(i-1,j), \gamma(i-1,j-1), \gamma(i,j-1)$ in cumulative distance matrix overflow, then the cell $\gamma(i,j)$ will overflow.

Proof:

□ $\gamma(i-1,j) > \varepsilon, \gamma(i-1,j-1) > \varepsilon, \gamma(i,j-1) > \varepsilon,$

□ $\min\{\gamma(i-1,j), \gamma(i-1,j-1), \gamma(i,j-1)\} > \varepsilon$ Moreover we have $d_{base}(u_i, v_j) \ge 0$. Hence with equation 10, we get $\gamma(i,j) > \varepsilon$.

Lemma 1 depicts the transmission of overflow of cells in cumulative distance matrix, as illustrated in

Figure 4. We check the overflow of neighbouring cells before calculating current cell in the cumulative distance matrix, and if all neighbouring cells overflow, the current cell should overflow.

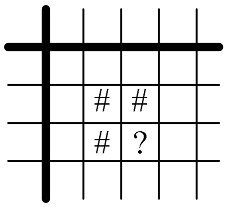


Figure 4. Illustration of transmission of overflow of cells.

Lemma 2: (overflow omission) If 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 Lemma 1, we have that the cell $\gamma(i,j)$ will overflow without regarding what the value of $d_{base}(u_i,v_j)$ is, thus $d_{base}(u_i,v_j)$ takes no effect on the overflow of $\gamma(i,j)$ and can be omitted.

By Lemma 2, we know that if the cell overflows, we should not be bothered to calculate the base distance for the cell, and save computations.

Lemma 3: (overflow replacement) If cell $\gamma(i,j)$ overflows, then replacing the value of cell $\gamma(i,j)$ with any value $\varepsilon^*(\varepsilon^*>\varepsilon)$ will not affect the overflow of cell $\gamma(i',j')(i'\ge i, j'\ge j)$.

Proof:
we prove the theorem with two cases:

- The value of $\gamma(i',j')$ is determined by the value of $\gamma(i,j)$. As $\gamma(i,j)>\varepsilon$, so $\gamma(i',j')>\varepsilon$, and if replacing value of $\gamma(i,j)$ with a value $\varepsilon^*(\varepsilon^*>\varepsilon)$, the $\gamma(i',j')>\varepsilon$ holds.
- The value of $\gamma(i',j')$ has no relation with the value of $\gamma(i,j)$.Surely in this case, the replacement will not affect the overflow of cell $\gamma(i',j')$.

By Lemma 3, if the cell overflows, the exact value of the cell can be replaced with any value that is above the threshold. As we will see in later part, the solution of EA_DTW replaces the value with the most max double value on one machine ($DOUBLE\_MAX$).

Lemma 4: If cell $\gamma(i,j)$ overflows, then replacing the value of cell $\gamma(i,j)$ with any value $\varepsilon^*(\varepsilon^*>\varepsilon)$ will not affect the overflow of the DTW distance.

Proof:
similarly, we consider the proof with following two cases:

- The minimal warping path crosses the cell (i,j). $\gamma(i,j)>\varepsilon$, γ is the cumulative distance and with the equation 6, we have $DTW(U,V)>\varepsilon$. If we replace $\gamma(i,j)$ with a value above ε, the same conclusion holds.
- The minimal warping path does not cross the cell (i,j). From theorem 3, provided that $\gamma(i,j)>\varepsilon$, the replacement will not affect the overflow of remaining cells. The value in (i,j) will not affect the value of $DTW(U,V)$ and can be omitted.

Lemma 5: (condition for early abandon on DTW) If one cell in the minimal warping path of cumulative distance matrix overflows, we have $DTW(U,V)>\varepsilon$

Proof:
before we go on, we first prove following fact:
in the minimal warping path W^* of cumulative distance matrix, for two cells $\gamma(i,j)$ and $\gamma(i',j')(i<i', j<j')$, the following equation holds. $\gamma(i,j)\le\gamma(i',j')$. Suppose the minimal warping path is $W^*=\{w_1,w_2,...,w_c\}$, and $w_k=f_w(i,j)$, $w_{k'}=f_w(i',j')(k'>k)$. Thus, $\gamma(i,j)=\sum_{t=1}^{k}w_t$, $\gamma(i',j')=\sum_{t=1}^{k'}w_{t'}=\sum_{t=1}^{k}w_t+\sum_{t=k+1}^{k'}w_t$.
As w_i is the value in the warping path, $w_i\ge 0$, thus, $\sum_{t=k+1}^{k'}w_t\ge 0$ .Hence we have $\gamma(i,j)\le\gamma(i',j')$.

Note that, $DTW(U,V)=\gamma(m,n)$, if one cell in the warping path overflows, say $\gamma(i,j)>\varepsilon(i\le m, j\le n)$, we have $DTW(U,V)\ge\gamma(i,j)>\varepsilon$.

The implication of Lemma 5 is exceedingly important. It provides the theoretical basis of the argument of early abandon on dynamic time warping. By Lemma 5, as long as one cell in the minimal warping path of cumulative distance matrix overflows, one can terminate the DTW calculation.

Lemma 6: If cells in one row or one column of the cumulative distance matrix all overflow, then we have $DTW(U,V)>\varepsilon$.

Proof:
from the constraints on the warping path (3) (4) and (5), warping path is a continuous curve from cell $(1,1)$ to cell (m,n) ,thus warping path must cross each row and each column in the distance matrix (though not each cell). If all cells in one row or column overflow, then there must be a cell in the warping path of cumulative distance matrix with the value exceeds ε .By Lemma 5, we have $DTW(U,V)>\varepsilon$.

3.2. Early Abandon on DTW Algorithm

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 omit 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 of DTW, and return no-match. The idea of early abandon on exact DTW is illustrated in Figure 5. EA_DTW returns false if the two sequences are not matched, i.e., the distance exceeds the given threshold, and returns true if they are matched.

```
Input: $U = \{u_0, u_1, ..., u_{m-1}\}$ $V = \{v_0, v_1, ..., v_{n-1}\}$ $\varepsilon$: the threshold for the query.
Output: $true$ if $DTW(U,V) \le \varepsilon$ ,otherwise $false$ .
1. construct an $m \times n$ matrix $M$ ;
2. $M[0][0] \leftarrow d_{base}(u_0, v_0)$;
3. if $M[0][0] > \varepsilon$ then
4.    return $false$;
5. end if;
(Initialize the $M[0][.]$ and $M[.][0]$)
6. if $M[i-1][0] > \varepsilon$ then
7.     $M[i][0] \leftarrow DOUBLE\_MAX$;
8. else
9.     $M[i][0] \leftarrow M[i-1][0] + d_{base}(u_i, v_0)$
10. end if;
  (Repeat lines 6~10 similarly on $M[0][.]$)
11. for $i = 1$ to $m-1$ do
12.   $overflow \leftarrow true$;
13.   for $j = 1$ to $n-1$ do
14.      $v \leftarrow \min\{M[i-1][j-1], M[i-1][j], M[i][j-1]\}$;
15.         if $v > \varepsilon$ then
16.            $M[i][j] \leftarrow DOUBLE\_MAX$;
17.         else
18.            $M[i][j] \leftarrow v + d_{base}(u_i, v_j)$;
19.       if $M[i][j] > \varepsilon$ then
20.          $M[i][j] \leftarrow DOUBLE\_MAX$;
21.       else
22.              $overflow \leftarrow false$;
23.       end if;
24.         end if;
25.   end for;
26.   if $overflow$ then
27.          break;
28.   end if;
29. end for;
30. if $overflow$ then
31.   return $false$;
32. end if;
33. return $M[m-1][n-1] \le \varepsilon$;
```

Figure 5. Early abandon on DTW.

For more details, in Figure 5, the algorithm EA_DTW constructs an $m \times n$ matrix for the cumulative distance and initializes the margin of the matrix (in lines 6 to 10). Lines 3 to 5 check if the first point alignment exceeds the threshold, and if so, we just cancel the calculation and return false. The rational here is that with the Endpoint constraint of warping path equation 6, the first point in the matrix will be in the warping path.

Lines 11 to 25 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 Lemma 1. If the cell overflows, then we replace the cell value with the max possible double value $(DOUBLE\_MAX)$ according to Lemmas 3 and 4, and omit the calculation of base distance according to Lemma 2; otherwise we calculate the value in normal way. Lines 26 to 28 check if the whole row overflows and if so, break out the loop and terminate the further calculation according to Lemma 6. The time complexity of EA_DTW is $O(mn)$, where m, n is the length of the two sequences, respectively.

3.3. Case Study

To further demonstrate the process of early abandon on DTW, we give a case study in this section. We use the same sequences $U = \{2,5,2,5,3\}$, $V = \{0,3,6,0,6,1\}$ in Figure 2 for example. Figure 6 shows the process of early abandon on DTW, and the tolerance ε is 9. Early abandon on DTW proceeds as follows:

- *Step 1*: Figure 6 (a) calculate the distance between first points, and check if it overflows. Since 4 < 9, we continue to step 2;
- *Step 2*: Figure 6(b) calculate the margin for the matrix, and fill with the overflowed cells with $DOUBLE\_MAX$ (denoted in the Figure 6 as "#");
- *Step 3*: Figure 6(c) calculate the body of the matrix sequentially and do the early abandon. This is the main step for the calculations.

For comparative study of the effects on early abandon, we draw the whole cumulative distance matrix in Figure 6(d). As the figure shows, there are many overflowed cells in the early abandon method, where the calculations of base distance are omitted. Another important, yet interesting observation in Figure 6(c) is that, the whole cells in 4th row of the matrix overflow, and after the detection, the remaining calculations are terminated (denoted as "##" in Figure 6(c)), which will improve the efficiency.

4. Experimental Study

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

4.1. Experiment Setup

For these experiments, we used a PC and the configurations are listed in Table 1.

	0	3	6	0	6	1
2	4					
5						
2						
5						
3						

(a)

	0	3	6	0	6	1
2	4	5	#	#	#	#
5						
2						
5						
3						

(b)

	0	3	6	0	6	1
2	4	5	#	#	#	#
5	#	8	6	#	#	#
2	#	9	#	#	#	#
5	#	#	#	#	#	#
3	#	##	##	##	##	##

(c)

	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

(d)

Figure 6. Illustration of early abandon on DTW (a) Calculate the first point and check if it overflows; (b) Calculate the margins of the matrix; (c) Calculate the body of the matrix with early abandon; (d) The comparative calculation results without early abandon. ("#"=Overflow, all cells denoted in bold are overflowed, "##"=Omit the calculation.)

Table 1. Experiment configurations.

Items	Value
Processor	Intel Pentium 3-866
Operating System	Ubuntu Linux 4.1.1-13 (Kernel version 2.6.2)
RAM	256 Megabytes
Disk Space	40 Gigabytes
Implement Language	ANSI C
Compiler	GNU gcc 4.1.2.20060928

In order to allow reproducibility, all the source codes 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 calculation of DTW recursively (denoted as raw_DTW). There is a step for construction of cumulative distance matrix in EA_DTW and dyn_DTW, while this is not a must in raw_DTW, which 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 a factor of saving computations.

- Elapsed Time: we used wall-clock time to measure the elapsed time during the evaluation. This time includes both CPU and IO time. As we repeated each experiment in several times, the elapsed time reported is the averaging value with the same parameters configured.
- Skip Rate: the steps that EA_DTW skips during calculation is a factor for the improvement, we use the following equation to denote the skip rate during early abandon:

$$skip\_rate = 1 - \frac{calculated\_cells\_number}{total\_cells\_number} \qquad (7)$$

We performed the experiments on both of synthetic and real data sets.

- Synthetic Datasets: the synthetic data sets were created using a random time series generator that produces n time series that confirm to simple, normal and exponential distributions, respectively.
- Real Datasets: our real dataset is the synthetic control chart, which was obtained from http://www.db-isc.uci.edu.

Sample data of data sets are shown in Figure 7. We randomly extracted 1 percent of the entries from each data set, the other subset of objects become our query set. This practice allowed us to use a query object that is in the same domain as the candidate query set.

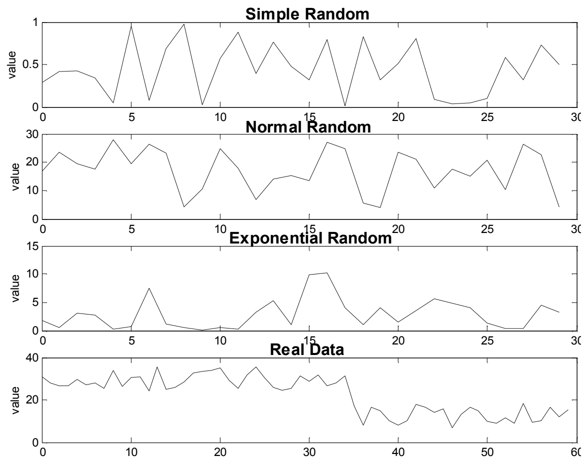


Figure 7. Plotting a fraction of the data sets.

4.2. Evaluation on Random Generated Datasets

The configurations for the random generated data are listed in Table 2.

Table 2. Parameters for experiments on random generated data sets.

Data Sets	Exact DTW Distance	Size of Matrix
Simple Random	1.3	10 * 13
Normal Random	66.4	8 * 11
Exponent Random	370.2	10 * 12

We found that there were many redundant calculations in raw_DTW compared with dyn_DTW, and raw_DTW had a poor performance, as shown in Table 3. For this reason, we mainly compare our method, EA_DTW, with the method dyn_DTW for the rest of the experiments.

Table 3. Comparisons of raw_DTW and dyn_DTW on elapsed time (in milliseconds).

Data Sets	raw_DTW	dyn_DTW
Simple Random	7,418,316.0	141.8
Normal Random	3,929,671.3	140.9
Exponential Random	251,528.6	130.3

The comparisons of the elapsed time on the randomly generated data are shown in Figure 8, Figures 9 and 10, respectively. In general, the results show that, the EA_DTW outperforms the dynamic calculation of DTW in the elapsed time, and wins in all threshold configurations, even when the threshold exceeds the exact distance.

We present the results of step rate on the random data sets in Figure 11. As the interesting results show, the skip rate becomes lower with the threshold grows, and when the threshold exceeds the exact distance of dynamic time warping, we may have less calculations on cells to omit (the most obvious example is the normal random case in Figure 11, when threshold exceeds the threshold, the skip rate becomes zero.). This is, however, not surprising and consistent with our theoretical analysis presented in section 3.2. Since the threshold exceeds the exact distance (1.3 in Simple random, 66.4 in normal random and 370.2 in exponential random), the early abandon behaviors more similarly to the whole calculation with dynamic programming, and no cells in the whole row or column will be skipped.

Nevertheless, we note that the time series objects obtained from querying is far more less than the whole set of objects. When performing exact dynamic time warping calculation on the objects that are not qualified, we may have that the threshold is below the exact distance. In this sense, we can adopt early abandon to gain more pruning power.

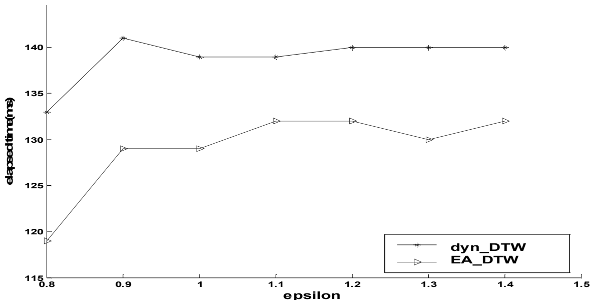


Figure 8. Comparison of elapsed time on simple random data.

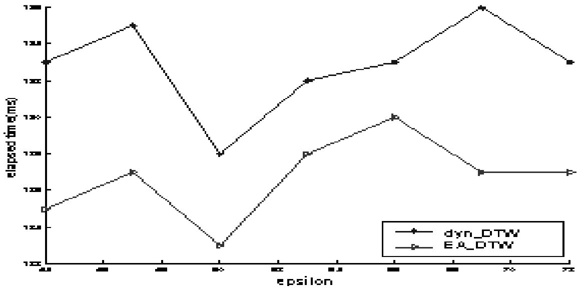


Figure 9. Comparison of elapsed time on normal random data.

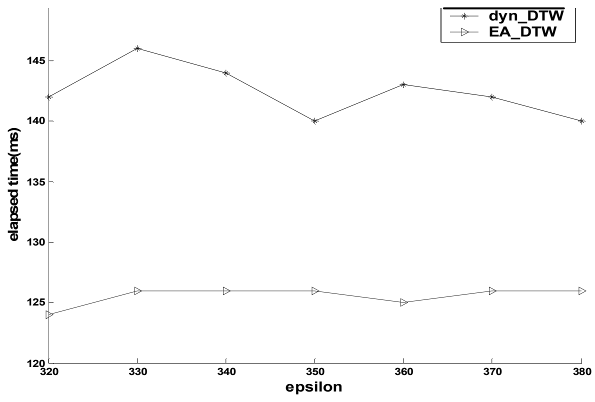


Figure 10. Comparison of elapsed time on exponential random data.

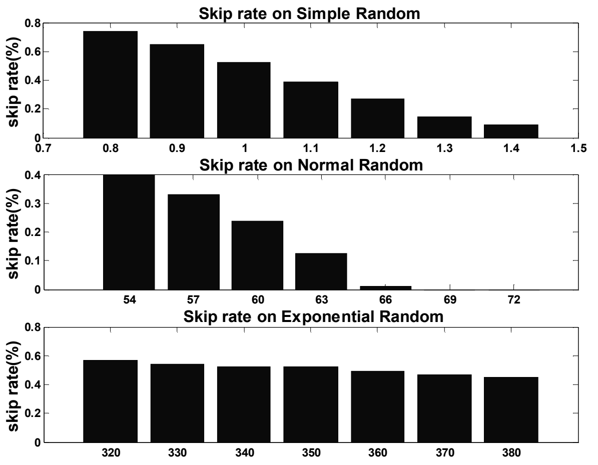


Figure 11. Skip rate on random data.

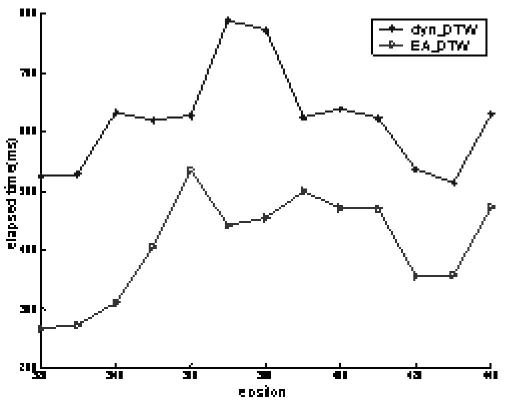


Figure 12. Comparison of elapsed time on real data.

4.3. Evaluation on Real Dataset

The length of the sequences in real datasets is 60, and the exact dynamic time warping distance between the randomly chosen sequences for querying is 413.1. The elapsed time on real dataset is illustrated in Figure 12. The same trend, as expected, is observed from the results, and EA_DTW yielded more evident performance gains in real data. With the same parameter configured, the EA_DTW finished the calculation in less time. The elapsed time of dyn_DTW ranged from 513 to 788, while the elapsed time of EA_DTW ranged from 267 to 499, which is roughly 60% of that in the dyn_DTW case.

Figure 13 shows the skip rate on real dataset. As the result indicates, the skip rate ranges from 55.3% to 70.9%. More than half of the cells calculations are omitted and the pruning power of EA_DTW is satisfying.

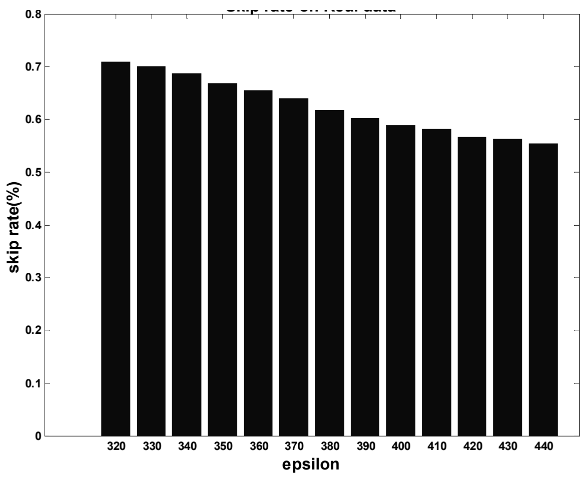


Figure 13. Skip rate on real data.

Note that even when the threshold is bigger than the distance, we have some improvement on the elapsed time, as well as the skip rate. This is due to that we can skip the calculations of base distance to save computations. Considering the size of the matrix (60 * 60), the overall gains on the omission of calculations is tremendous.

5. Related Work

Now that we have provided detailed descriptions of our contributions in this paper, we proceed to make a brief comparison with previous work. While there have been many methods proposed to calculate DTW, we use early abandon to accelerate the exact calculation of DTW. The methods of lower bounding functions in [4], [9], [14], [15] are, in reality, the approximate calculations of DTW, and need a step of refinement. Recently, the work in [13] also used the early abandon (they used the term early stopping) to terminate the calculation, however, we deepened the notation by omitting the calculation on base distance as well as terminating the whole DTW calculation.

6. Conclusions and Future Work

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 experiment results show, EA_DTW, compared with the dynamic programming method of DTW calculation, works with less calculation time, and the skip rate is better when the threshold is below the exact distance.

For future research, we plan to apply the method to the segmented-wised DTW calculation [9], where sequences are first segmented before the calculation of DTW, the difference with this work is that if we skip one cell in the cumulative distance matrix, we may skip more cells from same segment, since the values in one segment are more the same to each other.

Acknowledgements

An initial version of the work has been published on International Conference on Intelligent Systems and Knowledge Engineering (ISKE2007), Cengdu, China. We would like to thank comments of the conference reviewers as well as the anonymous reviewers of IAJIT, which greatly improve the quality of the paper. We also grateful to Professor Eamonn Keogh for his valuable insights. The work was supported by the project of secure and intelligent data integration platform in China with no. 2005-538.

References

[1] Berndt D. and Clifford J., “Using Dynamic Time Warping to Find Patterns in Time Series,” *AAAI Workshop on Knowledge Discovery in Databases*, USA, 1994.

[2] Faloutsos C., Ranganthan M., and Manolopoulos Y., “Fast Subsequence Matching in Time-Series Databases,” *in Proceedings of the ACM SIGMOD Conference*, USA, pp. 419-429, 1994.

[3] Keogh E., “Exact Indexing of Dynamic Time Warping,” *in Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, China, pp. 406-417, 2002.

[4] Keogh E., Li W., and Xi X., “LB_Keogh Supports Exact Indexing of Shapes Under Rotation Invariance with Arbitrary Representations and Distance Measures,” *in Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, Korea, 2006.

[5] Keogh E. and Kasetty S., “On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration,” *in the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, Canada, pp. 102-111, 2002.

[6] Keogh E. and Pazzani M., “A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases,” *in Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Japan, pp. 122-133, 2000.

[7] Kim S. and Jeong B., “Performance Bottleneck in Time-Series Subsequence Matching,” *in ACM*

Symposium on Applied Computing," Greece, pp. 469-473, 2005.

[8] Kim S., Park S., and Chu W., "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," *in Proceedings of 17th International Conference on Data Engineering (ICDE),* USA, pp. 607-614, 2001.

[9] Li W., Keogh E., Herle H., and Mafra A., "Atomic WedgieAtomic Wedgie: Efficient Query Filtering for Streaming Time Series," *in Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, USA, pp. 490-497, 2005.

[10] Mi Z. and Man W., "A Segment-Wise Time Warping Method for Time Scaling Searching," *Computer Journal Information Science*, vol. 90, no. 3, pp. 217-253, 2004.

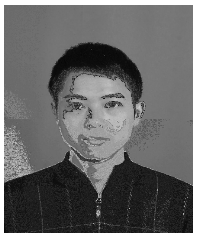
[11] Shou Y., Mamoulis N., and Cheung W., "Fast and Exact Warping of Time Series Using Adaptive Segmental Approximations," *Computer Journal Machine Learning*, vol. 58, no. 2, pp. 231-267, 2005.

[12] Xi X., Keogh E., and Shelton C., "Fast Time Series Classification Using Numerosity Reduction," *in Proceedings of the 23rd International Conference on Machine Learning*, USA, pp. 1033-1040, 2006.

[13] Yasushi S., Masatoshi Y., and Faloutsos C., "FTW: Fast Similarity Search Under the Time Warping," *in Proceedings of 24th ACM Symposium on Principles of Database Systems (PODS)*, USA, pp. 326-337, 2005.

[14] Yi B., Jagadish H., and Faloutsos C., "Efficient Retrieval of Similar Time Sequences Under Time Warping," *in Proceedings of 14th International Conference on Data Engineering (ICDE),* US, pp. 23-27, 1998.

[15] Zhu Y. and Shasha D., "Warping Indexes with Envelope Transforms for Query by Humming," *in Proceedings of ACM SIGMOD International Conference on Management of Data*, USA, pp. 181-192, 2003.



Li Junkui received his BSc degree in computer science and technology from Wuhan University of Technology, China in 2003. His research interests include various database technologies, data mining, and data stream processing.

Wang Yuanzhen is now a professor in College of Computer Science and Technology, Huazhong University of Science and Technology, China. Her research interests include modern database theory and implementation techniques, data mining, and middleware techniques.