Effective Filtering for Nearest-Neighbors Queries in Large Time-Series Databases

Simon Sheu
Computer Science
National Tsing Hua University
sheu@cs.nthu.edu.tw

Jinxiong Shen
Computer Science
National Tsing Hua University
u892545@oz.nthu.edu.tw

Abstract

Time-series data are periodic recordings of time-varying information. Since the data are temporal in nature, finding a similar data sequence in time-series databases to a given query is very costly. The straightforward strategy to examine each possible occurrence by sliding a window over each database sequence will take quadratic computation cost. For large time-series databases, this approach is practically infeasible. To shorten query response time, we propose in this paper a low-cost filtering mechanism to sieve out the most similar candidates from the dissimilar ones in the database. Then, only small portions of database require the true similarity measurement to finalize the query. As a result, our preprocessing approach achieves significant savings in overall query processing. We show our filtering technique incur no false dismissals, and has greater pruning power than the other competing schemes. Empirical results indicate 57% of non-similar data can be filtered out without resorting to the expensive true similarity measurement.

Keywords: Dynamic time warping, L2 distance, indexing, filtering, subsequence matching.

1. Introduction

Time-series data represent temporal tracking of information source over time. Examples include the pitch/beat information from the user's acoustic input [8], the heart rate of patients [1], the strength of gamma rays from celestial sources [2], hourly power demand of a research facility [19], RNA expression levels [3], just to name a few. The trajectories of mobile users on the plane or flying aircrafts in space are conceivable extensions of the original 1-D model [18]. Typically, time-series data consist of a sequence of N measured values from the observed aspect of the information source. These measures are likely affected by anonymous noises so that specific values alone hardly characterize the aspect being monitored. Rather, certain non-stationary features, e.g., abrupt changes, transient events, slowly varying trends, change patterns, are primary knowledge to be discovered. For instance, a classical ARIMA model of Box and Jenkins can be used to predict future values and obtain general insight into the time-dependent behavior [4]. However, the outline of global trend is obtained for the sacrifice of many details within the data. Often, we know the details of change patterns, and are interested to find one or more similar reoccurrences in time-series databases. This type of query is frequently referred to as k-Nearest-Neighbors (k-NN) search in the content-based information retrieval research. As an example, users can sing to retrieve an intended song in the music databases [8], or write down the words to be recognized by computer systems in automatic transcription [18]. Since the input query is shorter than database time sequences in length, this kind of query is also known as subsequence similarity matching [14, 15].

To answer k-NN query, some similarity metrics are required to objectively judge the extent of how similar the query is to the time sequences in databases. Most algorithms depend on Euclidean distance or some variation thereof [5, 13, 20]. Virtually, they slide a window over each database sequence to measure the similarity of the subsequence toward the query input. The k subsequences with the largest similarity measures from all possible sliding are then returned for the query. However, this approach is very sensitive to noise. Particularly, imperfectly taking the query sample likely deteriorates the quality of the results. For instance, mediocre singer barely rehearses the song in mind professionally. There exist small variations in the time axis. Even without the background noise, the pitch/beat information from the user's acoustic input taken in an ad hoc manner is often far away from the corresponding digital recordings. The query input with slightly defective tempo cannot match its desirable associates of standard form. Therefore, recent studies utilize Dynamic Time Warping (DTW) [6], more flexible distance measure for similarity match, to allow elastic shifting of the time axis [5, 8, 11, 12, 18, 20]. This distance metric permits each data point of the query to coincide with one or more consecutive points of the data subsequence in question, and vice verse. Effectively, DTW computes the shortest distance from stretchable alignments, and thus overcomes the problem that the query data may be slightly imperfectly aligned with the database subsequences in the time axis. The idea is similar in spirit to the edit distance measure used in approximate string matching [7, 9, 10, 14, 16, 17], where small gap of mismatched characters can be skipped during matching.

However, the capability of stretchable alignments comes at a cost of increased computational complexity. In spite the DTW distance can be computed by efficient dynamic programming algorithm [6], the overall cost of exhaustive search through the entire database becomes formidable as the database size increases. Especially, DTW does not obey the triangular inequality, rendering preprocessing database sequences to expedite k-NN query difficult. To address this problem, several approximate DTW functions are proposed to prune away the unnecessary true DTW computations [11, 12, 18, 20]. The general idea is to employ an efficient lower bound function of DTW to pre-qualify database subsequences for the query. Then, the true DTW distance of the best subsequence with the shortest approximate distance to the query is calculated. Since the approximate distance is always no greater than the true distance, the subsequences with the approximate distances larger than the currently best true distance can be safely discarded with no need to compute their true distances. Subsequently, the next best subsequences are repeatedly used to update the best true distance so as to keep the size of the pool of the candidates all with the true DTW distances equal to k. Obviously, how close the approximate function is to DTW determines the quality of this general pruning procedure. The tighter, yet low-cost, function will lead to considerable savings on expensive DTW computations. In contrast, a looser function may be computed more quickly. However, the majority of its approximate distances can be much smaller than the best true distance measure. As a result, its pruning capability is very limited.

In this paper, we focus on the tightness of approximate functions to the true DTW measures. Specifically, we develop a novel lower-bound function to improve the pruning power of all the existing competitors. To expedite the filtering process, we also investigate an upper-bound function to work seamlessly with the above approaches. We formally prove both low-cost approximate distances incur no false dismissals during filtering, and assess their qualities by extensive performance evaluations using numerous time-series data sets. The results indicate our new functions consistently outperform all the competing approaches. The rest of the paper is organized as follows. Section 2 briefly reviews the DTW algorithm and describes the related work. We will introduce the proposed low-cost approximate functions and prove their correctness in Section 3. Section 4 presents the performance study. Finally, we give our concluding remarks and discuss our ongoing work in Section 5.

2. Related Work

Given two time-series Q and C of the same length n, the classical L_2 distance, namely Euclidean distance, is defined as follows:

$$L_2(Q,C)=\sqrt{\sum_{i=1}^{n}(q_i-c_i)^2},\ \text{where } Q=q_1q_2\ldots q_n, C=c_1c_2\ldots c_n. \quad (1)$$

Comparatively, with stretchable alignments, the DTW distance between Q and C can be computed along an elastic warping path $W=w_1w_2\cdots w_m$, $n\le m<2n-1$, as

$$DTW(Q,C)=\min_W\sqrt{\sum_{k=1}^{m}[\![w_k]\!]},\ \text{where } w_k=(q_i,c_j), [\![w_k]\!]\triangleq(q_i-c_j)^2, \quad (2)$$

according to the following constraints on the warping path[1]:

- $w_1=(q_1,c_1), w_m=(q_n,c_n)$.
- $w_k=(q_i,c_j) \ \Rightarrow\ w_{k-1}=(q_{i-1},c_{j-1}),(q_i,c_{j-1})$, or (q_{i-1},c_j).
- $\forall w_k=(q_i,c_j),\ |i-j|\le r, \text{where } r\approx\frac{n}{20}$.[2]

DTW allows each q_i to flexibly coincide with one or more c_j, $i-r\le j\le i+r$, yet considers all possible paths to minimize the pair-wise distance. (If r=0, DTW degenerates to L_2.) DTW(Q,C) can be computed by the dynamic programming using the recursion [6]:

$$DTW^2(q_1\ldots q_i,c_1\ldots c_j)=(q_i-c_j)^2+\min\left\{\begin{array}{l}DTW^2(q_1\ldots q_{i-1},c_1\ldots c_{j-1}),\\ DTW^2(q_1\ldots q_{i-1},c_1\ldots c_j),\\ DTW^2(q_1\ldots q_i,c_1\ldots c_{j-1})\end{array}\right\}. \quad (3)$$

Fig. 1 illustrates one possible minimum-distance warping path (shown in blue) within the allow-

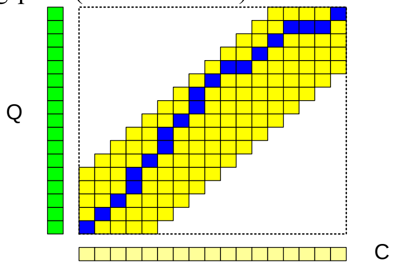

Fig. 1: an example to illustrate the minimum-distance warping path in DTW.

[1] For clarity, we consider Q and C have the same length. In fact, DTW is applicable for the different lengths.
[2] This constraint is known as the Sakoe-Chiba Band to confine W at most r away from the diagonal.

able region (shown in yellow) for n = 17. Some c_j's are repeatedly used to match q_i & q_{i+1} ($c_4 \leftrightarrow q_4,q_5$; $c_6 \leftrightarrow q_7,q_8$; $c_8 \leftrightarrow q_{10},q_{11}$), while two q_i's are continually in alignment with two and three c_j's ($q_{13} \leftrightarrow c_{10},c_{11}$; $q_{16} \leftrightarrow c_{14},c_{15},c_{16}$), respectively.

To answer k-NN query using DTW, the simple solution is to compute DTW(Q, C) for the query sequence Q and each database subsequence C using Eqn (2).The k subsequences with minimum distances are returned for the query. However, such approach is practically infeasible owing to high cost in DTW computation. Yi and Faloutsos proposed a fast approximate function of DTW [20] to avoid many unnecessary true DTW computations. Specifically, their function, denoted as YF by taking the authors' initials, is shown to be a lower bound function of DTW.

$$YF(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - c_{\max})^2, & q_i > c_{\max} \\ 0, & c_{\min} \le q_i \le c_{\max} \\ (q_i - c_{\min})^2, & q_i < c_{\min} \end{cases}},$$
$$where\ c_{\max} = \max_j c_j, c_{\min} = \min_j c_j. \qquad (4)$$

That is, $YF(Q,C) \le DTW(Q,C)$, since

$$YF^2(Q,C) = \sum_{i=1}^{n} \left\{ (q_i - c_{\max})^2, 0, (q_i - c_{\min})^2 \right\}$$
$$\le \sum_{j=1}^{n} (q_i - c_j)^2 \le DTW^2(Q,C), where (q_i, c_j) \in W. \qquad (5)$$

```
Input: query Q.
Output: k most similar database subsequences to Q.
Variable: Last best distance L, approximate distance d,
          true distance D,minimum queue M with at
          most k items, kept sorted in ascending order.
1.  Compute DTW(Q,C_i) for the first k database sub-
    sequences; insert(i, DTW(Q,C_i)) into M.
2.  L = the associated distance of the last item in M.
3.  For each C_i of the rest database subsequences
4.      d = YF(Q,C_i).
5.      If d < L
6.          D = DTW(Q,C_i).
7.          If D < L
8.              Insert (i, D) into M. [discard the k+1th item]
9.              L = the distance of the last (kth) item in M.
10.         Endif
11.     Endif
12. Endfor
```

Fig. 2: the k-NNs algorithm with a lower bound function YF.

With YF, the algorithm shown in Fig. 2 can be used to search for the k most similar database subsequences. The k currently best candidates are maintained in the queue M, which keeps the items inside sorted on the DTW distances in ascending order. For each subsequence in consideration, YF(Q,C) is first computed to be compared with L, the DTW distance of the kth best candidate. If greater than L, there is no need to compute DTW(Q,C) in Step 6. Only when C has an approximate distance smaller than L, it then could be one of the k-NNs. Step 7 verifies if this is true. If so, the currently kth best candidate is disqualified and removed from M in Step 8. The new kth best candidate is used as the pivot for ongoing filtering. As we can see in these iterations, the quality of the approximate function, such as YF(Q,C), in fact determines the pruning power of the algorithm in Fig. 2. If the function is not very tight, it is more likely that the algorithm executes Step 7 through 10, degenerating to the above simple solution using only DTW.

On the other hand, Kim, Park and Chu proposed another lower bound functions, denoted as KPC, using 4-tuple feature vector [12]. These four components are first and last elements of the subsequence, and the maximum & minimum values. The maximum absolute difference of corresponding features is used as KPC(Q,C). Mathematically,

$$KPC(Q,C) = \max \left\{ \begin{array}{l} |q_1 - c_1|, |q_n - c_n|, \\ |q_{\max} - c_{\max}|, |q_{\min} - c_{\min}| \end{array} \right\}. \qquad (6)$$

Besides the maximum and minimum, this function also considers the first constraint on the warping path: $w_1 = (q_1, c_1), w_m = (q_n, c_n)$. It can be applied similarly in the aforementioned algorithm simply by replacing YF(Q,C) by KPC(Q,C). We note that this function is symmetric: KPC(Q, C) = KPC(C, Q), while YF is not. If $c_{\max}$ and $c_{\min}$ in Eqn. (4) are extremely large and small, YF(Q,C) attends to yield very small value in our investigations. Therefore, we slightly modify YF function to make it symmetric, and denote the new function as YF2(Q,C). Likewise, $YF(Q,C) \le YF2(Q,C) \le DTW(Q,C)$.

$$YF2(Q,C) = \max\{YF(Q,C), YF(C,Q)\}. \qquad (7)$$

Most recently, Keogh [11] proposed a new lower bound function, denoted as KE, by utilizing the last path constraint: $\forall w_k = (q_i, c_j)$, $|i - j| \le r, where\ r = \frac{n}{20}$. This function is used to develop the index structure and further extended for 2D time-series based on the original DTW in Eqn. (2) [18]. The extension by Vlachos, et. al., is the first to introduce the upper bound of DTW in pruning. For clarity, we use VHGK[3] to denote this upper bound function. Both functions can be modeled: assume $U_i = \max_{i-r \le k \le i+r} c_k, L_i = \min_{i-r \le k \le i+r} c_k$.

$$KE(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2, & q_i > U_i \\ 0, & L_i \le q_i \le U_i \\ (q_i - L_i)^2, & q_i < L_i \end{cases}}, \qquad (8)$$

$$VHGK(Q,C) = \sqrt{\sum_{i=1}^{n} \max\left\{ (q_i - U_i)^2, (q_i - L_i)^2 \right\}}, \qquad (9)$$

Similar to YF, $KE(Q,C) \le DTW(Q,C)$. In particular, KE improves YF by using several

[3] The original definition is used for the internal nodes in the index tree. We formulate the distance for pairs of time-series data by following the same spirit.

piece-wise maximum and minimum values, instead of only c_{max} and c_{min}, to approach DTW. It is easy to show that $YF(Q,C) \le KE(Q,C)$. KE is a tighter approximation than YF. VHGK estimates the worst-case distance measure, which can be used to confine the true DTW: $DTW(Q,C) \le VHGK(Q,C)$. Both lower and upper bound functions can jointly work together to avoid expensive DTW computations. Since $KE \le DTW \le VHGK$, we can substitute DTW by VHGK in k-NNs algorithm to eliminate the subsequences whose lower bounds are exceeding the VHGK of the k$^{\underline{th}}$ candidate kept in M. Then, resort to the subsequent stage to finalize the query using true DTW distances. We will show the modified algorithm in Figure 5 and discuss the details of the procedure shortly after in Section 3 after introducing the proposed approximate functions of DTW.

3. Bounding the DTW function

In this section, we introduce the proposed lower bound and upper bound functions for DTW. We discuss our design philosophy and show the rationales behind their formulations. Subsequently, we will show the algorithm to utilize both approximate functions to efficiently eliminate the non-similar database subsequences without first computing their true DTW distances to the query.

3.1 Lower Bound function of DTW

As shown in the Eqn. (2), DTW is a minimum accumulated distance along all possible warping paths subject to several warping constraints. YF considers the maximum and minimum values of one sequence to reduce its distance measure to the other sequence. This approach is rather pessimistic. Only two extreme values are used to safely achieve lower bounding. Particularly, it takes no advantage of the third warping constraint, namely, $\forall w_k = (q_i, c_j)$, $|i-j| \le r, where\ r \approx \frac{n}{20}$. In reality, each c_j, even if extremely large or small, can only match with at most 2r q_i's. This fact motivates KE to apply the sliding window of width 2r for local maxima U_i's and minima L_i's so as to better lower bound DTW. On the other hand, KPC considers the first and last elements of the sequence additionally. When there are large differences on each boundary element of the sequences in comparison, this lower bound can be very close to DTW. In a sense, KPC benefits from exploiting the first warping constraint, $w_1=(q_1,c_1)$, $w_m=(q_n,c_n)$. From the insights of these lower bound functions, we devise a novel alternative that likely yields better

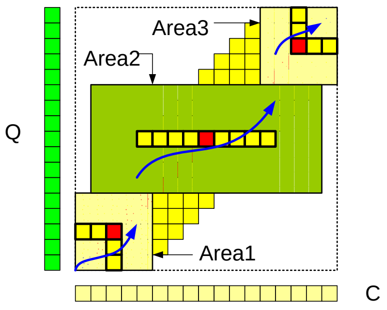

Fig. 3: the proposed lower bound function is constructed by a series of fences.

estimation. Fig. 3 illustrates our idea. Conceptually, we partition the permissible warping region into three areas: $Area_1$, $Area_2$, and $Area_3$. Each warping path will traverse through these areas. In $Area_2$, we follow the KE approach to construct (U_i, L_i) for each c_i of the database subsequence C. Any possible warping path, shown in blue curve, needs to walk through each horizontal bar, shown in bold rectangle, due to the 2nd constraint: $w_k = (q_i, c_j) \Rightarrow \quad w_{k-1} = (q_{i-1}, c_{j-1})$, (q_i, c_{j-1}) or (q_{i-1}, c_j). That is, at least one of (q_i, c_j), $|i-j| \le r$, belongs to the warping path of DTW. Therefore, the minimum distance of q_i to these c_j's, is included to safely lower bound DTW. The KE approach is good in this area.

However, KE does not work well in both $Area_1$ and $Area_3$, each an r×r square. Due to the boundary limitation, the length of a horizontal bar for each c_i gradually reduces to r as i approaches 1 or n. Particularly, any legitimate warping path will start from (q_1,c_1) and end at (q_n,c_n). By (U_1,L_1), aggregated from c_1 to c_r, will only give a far less accurate distance approximation. Indeed, using c_1 directly surely leads to a tighter lower bound since the distance between c_1 and q_1 is part of DTW. The same argument holds for c_n. Therefore, our approach is to employ an L-shape fence anchored along the diagonal in $Area_1$ and $Area_3$. Fig. 3 illustrates one fence in each area. Similar to horizontal bars, these fences will each intercept at least one point $w_k = (q_i, c_j)$ of the warping path for DTW. As shown in the figure, any path (presented in blue curve) will pass through any given fence. We note that our approach can bound DTW tighter than KE. As an example, both L-shape fences shown in bold consist of five w_k points, compared to seven points used for (U_3, L_3) or (U_{n-2}, L_{n-2}) in KE. Larger distance measures are expectable when a fence or bar contains fewer points in construction. This is because more points tend to increase the maximum U_i and decrease the minimum L_i, and thus reduce the estimated distance greatly, as indicated in Eqn. (8).

Our approach utilizes the first warping

constraint as the KPC approach. We formally define the proposed lower bound function, denoted as LB:

$$LB(Q,C)=\sqrt{\sum_{i=r+1}^{n-r}\begin{Bmatrix}(q_i-U_i)^2, & q_i>U_i\\ 0, & L_i\le q_i\le U_i\\ (q_i-L_i)^2, & q_i<L_i\end{Bmatrix}+\sum_{\substack{i=1\sim r,\\ n-r+1\sim n}}\min\left\{\begin{Bmatrix}(q_i-F_i)^2, & q_i>F_i\\ 0, & R_i\le q_i\le F_i\\ (q_i-R_i)^2, & q_i<R_i\end{Bmatrix},\begin{Bmatrix}(c_i-G_i)^2, & c_i>G_i\\ 0, & S_i\le c_i\le G_i\\ (c_i-S_i)^2, & c_i<S_i\end{Bmatrix}\right\}},$$

$$where\begin{cases}U_i=\max\limits_{i-r\le k\le i+r}c_k, L_i=\min\limits_{i-r\le k\le i+r}c_k.\\ F_i=\max\limits_{1\le k\le i}c_k, R_i=\min\limits_{1\le k\le i}c_k,\\ G_i=\max\limits_{1\le k\le i}q_k, S_i=\min\limits_{1\le k\le i}q_k.\end{cases}\qquad(10)$$

Area$_2$ reiterates KE's definition in Eqn. (8) using a series of horizontal bars. In Area$_1$ and Area$_3$, the minimum distance is accumulated on each L-shape fence. When the argument i is 1 or n in the above formula, $F_i = R_i = c_i, G_i = S_i = q_i$. The squared difference $(q_i - c_i)^2$ is included exactly as DTW. The lower bounding property of LB is established by the proof on the inequality, $LB(Q,C) \le DTW(Q,C)$:

Proof: Given the minimum-distance warping path $W = w_1w_2\cdots w_m, n \le m < 2n-1$, of DTW, we abbreviate W into X from the joints at which W intersects a total of n horizontal bars or L-shape fences. If there are more than one joints for each bar or fence, we retain only the first one in X. Clearly, X has n terms: $X = x_1x_2\cdots x_n$. In particular, $DTW = \sqrt{\sum_{k=1}^{m}[\![w_k]\!]} \ge \sqrt{\sum_{k=1}^{n}[\![x_k]\!]}$. For each $x_k = (q_i, c_j)$ on the bar or fence, LB would wish to include the squared difference $[\![x_k]\!] = (q_i - c_j)^2$ ideally. However, such joint point is hard to predict in advance. Conservatively, LB comprehends the amount that is no greater than $[\![x_k]\!]$. For a horizontal bar, if $q_i > U_i = \max\limits_{i-r\le k\le i+r} c_k$, LB has $(q_i - U_i)^2$ that is surely no greater than $[\![x_k]\!]$; likewise, the similar argument applies for the case that $q_i < L_i$. Besides, when $L_i \le q_i \le U_i$, LB includes only zero that is no larger than $[\![x_k]\!]$ because $[\![x_k]\!] \ge 0$. For an L-shape fence, LB virtually divides it into a horizontal bar and a vertical bar, and embodies the minimum from two separate conservative estimations as before. Therefore, the amount that is summed into LB inside its square root is at most as large as $[\![x_k]\!]$. Since each term included into LB is no greater than $[\![x_k]\!]$ on bars or fences, LB(Q,C) is surely no greater than $\sqrt{\sum_{k=1}^{n}[\![x_k]\!]}$, which is in turn $\le \sqrt{\sum_{k=1}^{m}[\![w_k]\!]} = DTW$. We conclude the inequality $LB(Q,C) \le DTW(Q,C)$ holds. Q.E.D.▪

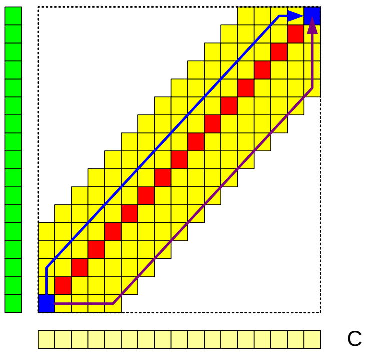

Fig. 4: the proposed upper bound function is built by a series of diagonal-like lines.

3.2 Upper Bound function of DTW

Our low-cost upper bound for DTW is a simple extension of the classical Euclidean distance. We compute distance measure along a set of simple diagonal warping paths and use the minimum as the function output. Fig. 4 shows this idea. As indicated, the diagonal-like warping path in blue progresses first vertically up for two points, then proceeds straightly parallel to the diagonal line, and finally moves horizontally to the end point. The path in purple consists of four horizontal movements, diagonal movements, and four vertical movements. Both paths are legal for DTW. In fact, DTW is computed from all possible warping paths. Appreciably, the measure from either path upper bounds the true DTW, and can be computed very quickly

We incorporate a parameter s into our upper bound function, denoted as UB, to specify the size of the set in consideration. For s = 0, the set contains only the diagonal path. Thus, UB is exactly as the Euclidean measure. For s = 1, UB considers additionally the two warping paths that proceed one point off the diagonal. The value of s is limited by the third warping constraint: $\forall w_k = (q_i, c_j),\ \ |i-j| \le r, where\ r \approx \frac{n}{20}$. That is, $0 \le s \le r$. Formally, $UB(Q,C,s) = \min\limits_{-s\le t\le s} \sqrt{P(Q,C,t)}$,

$$\begin{aligned}P(Q,C,t) &= \Sigma_{i=1}^{t}(q_1-c_i)^2 + \Sigma_{i=t+1}^{n-1}(q_{i-t+1}-c_i)^2 + \\ &\quad \Sigma_{i=n-t+1}^{n}(q_i-c_n)^2,\ \ t\ge 0 \quad \text{or} \\ &=\Sigma_{i=1}^{-t}(q_i-c_1)^2+\Sigma_{i=-t+1}^{n-1}(q_i-c_{i+t+1})^2+ \\ &\quad \Sigma_{i=n+t+1}^{n}(q_n-c_i)^2,\ \ t\le 0.\end{aligned}\qquad(11)$$

It is trivial to show $DTW(Q,C) \le UB(Q,C,s)$, and

$UB(Q,C,r) \le UB(Q,C,r-1) \le \cdots \le UB(Q,C,0) = L_2(Q,C)$. We note that VHGK in Eqn. (9) also an upper bound function of DTW. However, its quality is not as good as our UB. The following proof establishes the relation, $DTW(Q,C) \le UB(Q,C,s) \le L_2(Q,C) \le VHGK(Q,C)$ by showing that VHGK is no less than the L_2 distance.

Proof: Recall the definition of VHGK

$VHGK(Q,C) = \sqrt{\sum_{i=1}^{n} \max\left\{(q_i - U_i)^2, (q_i - L_i)^2\right\}}, U_i = \max_{i-r \le k \le i+r} c_k, L_i = \min_{i-r \le k \le i+r} c_k$.

Since $\max\left\{(q_i - U_i)^2, (q_i - L_i)^2\right\} \ge (q_i - c_i)^2$ for either $c_i = U_i, L_i, \text{or } L_i < c_i < U_i$, it is easy to obtain $L_2(Q,C) \le VHGK(Q,C)$. Q.E.D.▪

3.3 Using both approximate functions of DTW

Like lower bound functions, an upper bound can be used to further save the expensive computation of the true DTW. We provide an algorithm using LB and UB in Figure 5. Any lower bound functions can be used by simply replacing LB therein, while an upper bound function, such as VHGK, can substitute UB to work right away. Instead of computing DTW for the first k database subsequences as in Fig. 2, this modified algorithm starts with their lower and upper estimates in Step 1. Insert them with their associative distances into M and N, followed by iteratively checking over the rest of database subsequences in Step 4 through 14. In each iteration, the lower estimate is first computed. If such value is greater than L, the currently k<u>th</u> best estimates, this subsequence can be safely disregarded. Otherwise, it can be part of the answer. We insert it into the candidate queue N, and go on to check if its upper estimate can help reducing L. If so, Step 10 and 11 update L. The original subsequence defining L is removed as well. Step 16 through 20 are optional. Its presence supports further filtering the items in N using the best L that is obtained from previous iterations. Afterwards, we need the true DTW computations on each item kept in N to finalize the query.

```
Input:query Q, the number of nearest-neighbors k.
Output:k most similar subsequences in the database to Q.
Variable:Last best distance L, lower estimate d, upper
         estimate D,minimum queue M with at most k
         items, kept sorted in ascending order,candidate
         queue N holding the possible result from filter-
         ing.
1.  Compute UB(Q,$C_i$) and LB(Q,$C_i$) for the first k
    database subsequences; insert (i, UB(Q,$C_i$)) into M
    and insert (i, LB(Q,$C_i$)) into N.
2.  L = the associated distance of the last item in M.
3.  // First Pass
4.  For each $C_i$ of the rest database subsequences
5.      d = LB(Q,$C_i$).
6.      If d < L [counters]
7.          Insert (i,d) into N.
8.          D = UB(Q,$C_i$).
9.          If D < L
10.             Insert (i, D) into M. [discard its k+1th item]
11.             L = the distance of the last (kth) item in M.
12.         Endif
13.     Endif
14. Endfor
15. // Second Pass
16. For each (i,d) in N
17.     If d of (i,d)>L
18.         Remove (i,d) from N
19.     Endif
20. Endfor
21. // Post-processing stage: finalize the query
22. Empty M.
23. If the size of N is larger than k
24.     For each (i,d) in N
25.         Compute DTW(Q,$C_i$); insert (i, DTW(Q,$C_i$))
              into M. [discard the replaced item]
26.     Endfor
27. Endif
```

Fig. 5: the k-NNs algo. with lower/upper bound functions.

We note that the effect of filtering processes can be measured by the size of N prior to Step 22. The larger the size, the more costly the overhead in DTW computation. If |N| is exactly equal to k, then the items in N are the answer. Otherwise, we use M to keep the k best candidates, and return the items in M for the query. The quality of lower and upper estimate functions is the key to reduce the size of N during filtering. We will present the results from extensive investigations over the diverse time-series data of great variety in the next section.

4. Performance Study

To assess the quality of approximate functions, we have performed experiments using a total of 32 datasets, which were used originally in the study of KE [11].These sets range over stationary/non-stationary, and noisy/smooth, time-series data in different fields [1-3, 19] so as to justify the general applicability of the approximate functions. Since the focus of this paper is on the quality of the approximate functions, we only report the results from 1-NN queries using the algorithms in Fig. 2 & 5 for brevity. We set n=256 & r=12 in this study in order for the interested readers to compare our results with the ones shown in [11]. To avoid ambiguity, LB_Ours and UB_Ours(s) are used to present

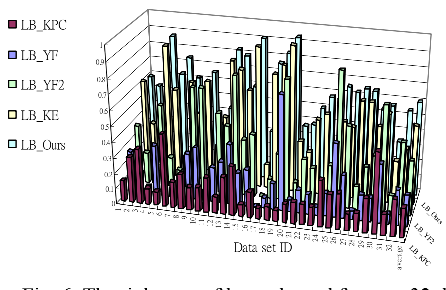

Fig. 6: The tightness of lower bound fn. over 32 data sets.

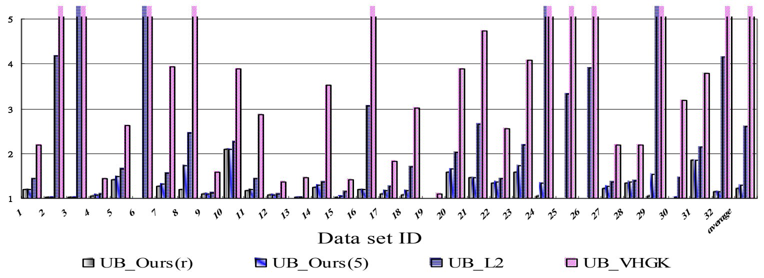

Fig. 7: The tightness of upper bound functions over 32 data sets.

	LB_YF	LB_YF2	LB_KPC	LB_KE	LB_Ours	UB_VHGK	UB_L2	UB_Ours(5)	UB_Ours(r)
Max	0.74	0.86	0.52	0.93	0.94	64.38	9.69	2.10	2.09
Min	0.00	0.00	0.07	0.06	0.32	1.09	1.01	1.00	1.00
Avg	0.20	0.35	0.19	0.51	0.61	7.70	2.61	1.28	1.21

Fig. 8: The summary information on the tightness measure of approximate functions.

the proposed lower bound and upper bound functions, respectively. LB_YF, LB_YF2, LB_KPC, and LB_KE stand for the other lower bound functions, while the upper bound functions VHGK and Euclidean are denoted as UB_VHGK and UB_L2, respectively. For fair judgment, 50 subsequences of length 256 are randomly extracted from each of 32 data sets for each run of experiment. The same set of testing data is applied for each approximate function: one subsequence serves as the query against the 49 others. The average from the total of $C_2^{50} = 1225$ comparisons is used as the performance index. For simplicity, we choose s = 5 and s = r (=12) to portrait the effect of the argument setting in UB_Ours(s). Notice that UB_Ours(0) is the same as UB_L2.

4.1 The tightness of the approximate functions

We measure the tightness of the approximate functions as the ratio of the estimated distance over the true DTW distance, as in Eqn. (12). The closer this ratio
is to 1, the better.

$$Tightness : T = \frac{\text{Estimated distance}}{\text{True DTW distance}}. \qquad (12)$$

Fig. 6 & 7 show the results of the experiments, which are summarized in Fig. 8. Among the lower bound functions, LB_KPC performs the worst (T=0.19) on the average, while LB_YF and its improved version, LB_YF2, may yield the distance estimates[4] that are very close to zero. They cannot consistently deliver satisfactory results. LB_KE may be a good candidate in predicting DTW distances such that its tightness measure can favorably reach as high as 0.93. However, it is not stable enough for the serious time-series database applications. Its worst-case measure is indeed inferior to LB_KPC. In contrast, the proposed lower bound function addresses its shortcomings, and is the only option available to offer excellent performance guarantee, as highlighted in Fig. 8. The key for such achievement is to employ smaller fences in both boundary regions of legitimate warping region. On the other hand, UB_VHGK may closely upper bound the true DTW distance by the best T=1.09. However, its performance is not very reliable since most of its distance estimations are very high. In fact, the classical Euclidean distance outperforms UB_VHGK in every experiment. This empirical evidence is consistent with the theoretical inference established earlier in Section 3.2. Just as expected, our upper bound function is a great DTW-distance predictor. With s = 5, it can estimate the true DTW distance with high accuracy, mostly within 28% overestimate range. This result also indicates UB_Ours(5) is good enough for most of upper-bound estimates in spite a higher value of s is more favorable. We will present its effect on savings of the expensive DTW computations in the following subsection.

4.2 Pruning power of the approximate functions

Fig. 2 & 5 present the algorithms to filter out non-similar database subsequences without the expensive computation of the true DTW distances using the lower bound function and upper bound function. To justify the effectiveness of such preprocessing, we employ the following metric to measure the pruning power:

$$Pruning\ power : P = \frac{\text{Nonsimilar subsequence disqualified w/o DTW}}{\text{Total number of DB Subsequences}}. \qquad (13)$$

We note that $0 \le P \le 1$, a larger P measure signifies more pruning power. Fig. 9 summarizes the p-measure from extensive experiments on 32 data sets. The first group shown on the left-hand side presents the pruning power of lower bound functions by the algorithm in Fig. 2. LB_YF is better than LB_KPC, and can be improved further as LB_YF2 simply by additionally exchanging two input sequences. LB_KE performs very well. However, for some severe test cases, its p-measure may be as low as zero: essentially no effect of filtering. LB_Ours preserves the general pruning power, and has been the best performer ever since.

[4] The actual values are 0.001675 and 0.004937, respectively.

	LB_YF	LB_YF2	LB_KPC	LB_KE	LB_Ours	LB_KE + UB_VHGK		LB_KE + UB_L2		LB_Ours + UB_Ours(5)	
						1st Pass	2nd Pass	1st Pass	2nd Pass	1st Pass	2nd Pass
Max	0.80	0.85	0.66	0.89	0.90	0.85	0.93	0.87	0.96	0.88	0.96
Min	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.01	0.03
Avg	0.23	0.34	0.20	0.47	0.56	0.15	0.17	0.33	0.39	0.47	0.57

Fig. 9: The summary information on the pruning power of approximate functions.

The second group on the right-hand side of Fig. 9 results from the algorithm in Fig. 5 by replacing the true DTW computations with the upper bound functions. Comparatively, this algorithm can disqualify dissimilar subsequences much more rapidly owing to using only low-cost approximate functions. As shown, UB_VHGK hardly assumes the role of upper functions. Its performance is even worse than the simple Euclidean function. However, UB_L2 still cannot offer satisfactory filtering performance. In the worst case, UB_L2 paired by LB_KE yet fails to meet the challenge set by the most intractable test cases. With vast variations, these data tend to have very small lower bound measure and very high upper bound value. Powered by LB_Ours, UB_Ours(5) addresses this problem very well. On the first pass, 47% of non-similar data objects on average are filtered out with no true DTW computations, which is compatible with using LB_KE alone with the true DTW in the first group. The second pass further refines the candidate set, and gives additional 10% improvement (0.57-0.47= 0.1 = 10%). The proposed lower and upper bound functions significantly outperform all the competing functions, and achieve the best filtering result.

5. Concluding Remarks

Nearest-neighbors queries in large time-series databases are popular, but very expensive to execute. The classical Euclidean distance-based index structures, despite their high efficiency, require the perfect input for the query. Dynamic Time Warping disrupts this barrier by allowing the query input with slightly elastic shifting of the time axis. However, the stretchable alignments bear high cost in computation. In particular, the triangular inequality cannot apply for efficient index construction. The current trend is to employ low-cost approximate functions to filter out most non-similar objects before the expensive true distance comparisons. We proposed two approximate functions in this paper to improve the capability of all the existing competitors. We show both functions can work closely to deliver satisfactory preprocessing results. Only relying on these low-cost approximate functions, more than half of database subsequences can be safely sieved out without any dismissal. This achievement shall foster the excellent quality of the index structure we are currently developing based on the approximate functions presented in the paper.

Acknowledgements

We are devoutly thankful for the supports of the National Science Council, Project No. NSC 91-2213-E-007-040, and the MOE Program for Promoting Academic Excellent of Universities, Grant No. 89-E-FA04-1-4, Taiwan, R.O.C. In particular, we appreciate very much the help of Eamonn Keogh for providing the original test data, and the comments of Roger Jang about music retrieval by DTW.

References

[1] "Example of a Life-Threatening Physiologic Event," http://reylab.bidmc.harvard.edu/DynaDx/case-study/seizure/menu.html, 2003.

[2] "Periodicity in a Gamma Ray Buster," http://xweb.nrl.navy.mil/timeseries/multi.diskette, 2003.

[3] J. Aach, http://arep.med.harvard.edu/timewarp/supplement.htm, "Aligning gene expression time series with time warping algorithms," June 11, 2001.

[4] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 2nd ed: Holden-Day, 1976.

[5] F. K.-P. Chan, A. W.-c. Fu, and C. Yu, "Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping," *IEEE Trans. on Knowledge and Data Engineering*, 15(3):686-705, May/June, 2003.

[6] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*: Prentice Hall PTR, 2001.

[7] E. Hunt, "The Suffix Sequoia Index for Approximate String Matching," Department of Computing Science, University of Glosgow, Glasgow, UK, TR 2003-135, March, 2003.

[8] J.-S. R. Jang and H.-R. Lee, "Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input," in Proc. of ACM MM, Canada, Sept., 2001.

[9] T. Kahveci and A. Singh, "An Efficient Index Structures for String Databases," in Proc. of VLDB, Italy, pp. 351-360, Sept., 2001.

[10] W. J. Kent, "BLATThe BLAST-Like Alignment Tool," *Genome Research*, 12(4):656-664, April, 2002.

[11] E. Keogh, "Exact Indexing of Dynamic Time Warping," in Proc. of VLDB Conference, Hong Kong, China, Aug. 20-23, 2002.

[12] S.-W. Kim, S. Park, and W. W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," in Proc. of IEEE Data Engineering, Germany, pp. 607-614, April, 2001.

[13] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences," in Proc. of ACM SIGMOD, Arizona, pp. 289-300, May, 1997.

[14] Y.-S. Moon, K.-Y. Whang, and W.-S. Han, "GeneralMatch: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows," in Proc. of ACM SIGMOD, pp. 382-393, June 3-6, 2002.

[15] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," in Proc. of IEEE Data Engineering, pp. 263-272, April 2-6, 2001.

[16] Z. Ning, A. J. Cox, and J. C. Mullikin, "SSAHA: a fast search method for large DNA databases," *Genome Research*, 11(10):1725-1729, Oct., 2001.

[17] O. Ozturk and H. Ferhatosmanoglu, "Effective Indexing and Filtering for Similarity Search in Large Biosequence Databases," in Proc. of IEEE Sym. on BioInformatics and BioEngineering, pp. 359-366, March, 2003.

[18] M. Vlachos, M. Hadjieleftheriou, D. G. y, and E. Keogh, "Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures," in Proc. of ACM SIGKDD, Aug. , 2003.

[19] J. J. v. Wijk and E. R. v. Selow, "Cluster and Calendar based Visualization of Time Series Data," in Proc. of IEEE Symposium on Information Visualization, San Francisco, Oct. 25-26, 1999.

[20] B.-K. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary Lp Norms," in Proc. of VLDB, Sept., 2000.