

Exact Indexing of Dynamic Time Warping

Eamonn Keogh

University of California - Riverside
Computer Science & Engineering Department
Riverside, CA 92521
USA
eamonn@cs.ucr.edu

Abstract

The problem of indexing time series has attracted much research interest in the database community. Most algorithms used to index time series utilize the Euclidean distance or some variation thereof. However it has been forcefully shown that the Euclidean distance is a very brittle distance measure. Dynamic Time Warping (DTW) is a much more robust distance measure for time series, allowing similar shapes to match even if they are out of phase in the time axis. Because of this flexibility, DTW is widely used in science, medicine, industry and finance. Unfortunately however, DTW does not obey the triangular inequality, and thus has resisted attempts at exact indexing. Instead, many researchers have introduced approximate indexing techniques, or abandoned the idea of indexing and concentrated on speeding up sequential search. In this work we introduce a novel technique for the exact indexing of DTW. We prove that our method guarantees no false dismissals and we demonstrate its vast superiority over all competing approaches in the largest and most comprehensive set of time series indexing experiments ever undertaken.

1. Introduction

The indexing of very large time series databases has attracted the attention of database community in recent years. The vast majority of work in this area has focused on indexing under the Euclidean distance metric [5, 10, 17, 18, 21, 34]. However there is an increasing awareness

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment

**Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002**

that the Euclidean distance is a very brittle distance measure [6, 16, 20]. What is needed is a method that allows an elastic shifting of the time axis, to accommodate sequences which are similar, but out of phase, as shown in Figure 1. Just such a technique, based on dynamic programming, has long been known to the speech processing community [27, 29]. Berndt and Clifford introduced the technique, Dynamic Time Warping (DTW), to the database community [3]. Although they demonstrate the utility of the approach, they acknowledge that its resistance to indexing is a problem and that “...performance on very large databases may be a limitation”. Despite this shortcoming of DTW, it is still widely used in various fields: In bioinformatics, Aach and Church successfully applied DTW to RNA expression data [1]. In chemical engineering, it has been used for the synchronization and monitoring of batch processes in polymerization [13]. DTW has been successfully used to align biometric data, such as gait, signatures and even fingerprints [12]. Many researchers including Caiani et al. [4] have demonstrated the utility of DTW for ECG pattern matching. Finally in robotics, Schmill et al. demonstrated a technique that utilizes DTW to cluster an agent’s sensory outputs [30].

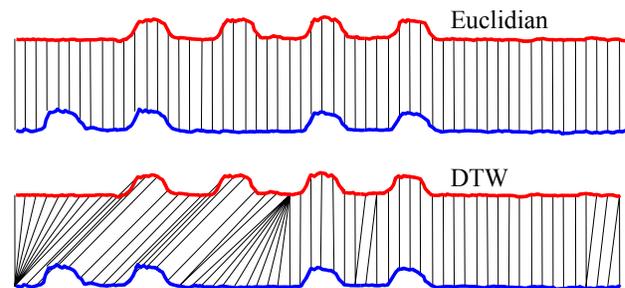


Figure 1: Note that while the two sequences have an overall similar shape, they are not aligned in the time axis. Euclidean distance, which assumes the i^{th} point in one sequence is aligned with the i^{th} point in the other, will produce a pessimistic dissimilarity measure. The nonlinear Dynamic Time Warped alignment allows a more intuitive distance measure to be calculated

More than two-dozen techniques have been introduced to index time series under the Euclidean distance [5, 10, 17, 18, 21, 34] (see [18] for a more comprehensive listing). In addition, several researchers have shown techniques to approximately index DTW [33], or introduced methods to reduce its demanding CPU time [6]. However only two researchers have claimed to have introduced an exact indexing technique for DTW [19, 23]. In the case of [23], the claim of no false dismissals was later retracted [24, 26]. We have carefully implemented only technique to correctly claim the ability to exactly index DTW, and the only other lower bounding approximation of DTW, for detailed comparison with our proposed approach.

In contrast to the approaches above, we will prove the no false dismissal property of our approach and demonstrate its superiority with the most comprehensive set of time series indexing experiments ever undertaken. In particular, in terms of number and diversity of datasets, size of datasets, range of query lengths and indexing parameters, our experiments are one to two orders of magnitude larger than all previous papers combined.

The rest of the paper is organized as follows. In Section 2 we will consider the utility of time series similarity search, review the DTW algorithm, and consider related work. In Section 3 we will introduce a novel lower bounding technique that tightly approximates the true DTW distance. Section 4 introduces a method that allows the exact indexing using our lower bounding function. In Section 5 we conduct an exhaustive empirical comparison of our method with competing techniques. Finally in Section 6 we offer conclusions and suggestions for extensions.

2. Background

Similarity search in time series is useful in its own right as a tool for interactive exploration of very large databases, it is also a subroutine in many data mining applications including rule discovery [7], clustering [6, 8], and classification [9, 16]. The superiority of DTW over Euclidean distance for these tasks has been demonstrated by several authors [1, 6, 13, 20, 30, 33]. However for completeness, we include a simple experiment to illustrate the point.

The most studied time series classification/clustering problem is the Cylinder-Bell-Funnel dataset [16], it is a deceptively simple looking 3-class problem, Figure 2 shows typical examples of each class.



Figure 2: Typical examples from the Cylinder-Bell-Funnel dataset

The problem has been attacked with sophisticated techniques including rule-based learners [16], boosting,

Bayesian techniques, and decision trees. We performed a simple classification experiment on this dataset, using the 1- nearest neighbour algorithm. Our dataset consists of ten instances of each class, and the classifier was evaluated using the “leaving-one-out” strategy. Since we had the luxury of unlimited data, we averaged the results over 1,000 runs. The mean error rate for the Euclidean distance metric on the problem was 0.2734, but for DTW it was only 0.0269, an order of magnitude lower. This “off-the-shelf” result is competitive with the highly tuned, sophisticated techniques enumerated above. The lower error rate came with a cost however; classification with DTW took approximately 230 times longer than with Euclidean distance.

This result reiterates the utility of DTW and motivates the necessity of introducing techniques to index it.

C	A time series of length n $C = c_1, c_2, \dots, c_i, \dots, c_n$.
$[c_i : c_j]$	A subsequence of C , beginning at c_i and ending at c_j
\bar{C}	A piecewise linear approximation of a time series. [17,34]
DTW	The Dynamic Time Warp distance measure
LB_Kim	The lowerbounding function introduced by Kim et al. [19]
LB_Yi	The lowerbounding function introduced by Yi et al. [33]
LB_Keogh	The lower bounding function introduced in this work

Table 1: The basic notation used in this work

2.2 Review of DTW

Suppose we have two time series Q and C , of length n and m respectively, where:

$$Q = q_1, q_2, \dots, q_i, \dots, q_n \quad (1)$$

$$C = c_1, c_2, \dots, c_j, \dots, c_m \quad (2)$$

To align two sequences using DTW, we construct an n -by- m matrix where the $(i^{\text{th}}, j^{\text{th}})$ element of the matrix contains the distance $d(q_i, c_j)$ between the two points q_i and c_j (i.e. $d(q_i, c_j) = (q_i - c_j)^2$). Each matrix element (i, j) corresponds to the alignment between the points q_i and c_j . This is illustrated in Figure 3. A warping path W , is a contiguous (in the sense stated below) set of matrix elements that defines a mapping between Q and C . The k^{th} element of W is defined as $w_k = (i, j)_k$ so we have:

$$W = w_1, w_2, \dots, w_k, \dots, w_K \quad \max(m, n) \leq K < m+n-1 \quad (3)$$

The warping path is typically subject to several constraints.

- **Boundary conditions:** $w_1 = (1, 1)$ and $w_K = (m, n)$, this requires the warping path to start and finish in diagonally opposite corner cells of the matrix.
- **Continuity:** Given $w_k = (a, b)$ then $w_{k-1} = (a', b')$ where $a - a' \leq 1$ and $b - b' \leq 1$. This restricts the allowable steps in the warping path to adjacent cells (including diagonally adjacent cells).
- **Monotonicity:** Given $w_k = (a, b)$ then $w_{k-1} = (a', b')$ where $a - a' \geq 0$ and $b - b' \geq 0$. This forces the points in W to be monotonically spaced in time.

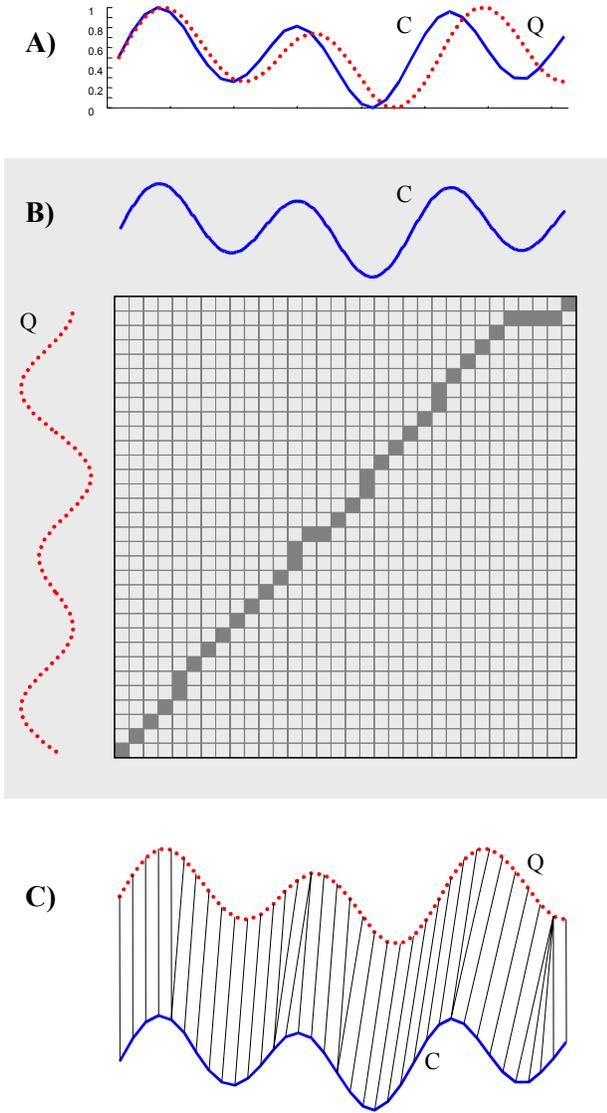


Figure 3: **A)** Two sequences Q and C which are similar, but out of phase. **B)** To align the sequences we construct a warping matrix, and search for the optimal warping path, shown with solid squares. **C)** The resulting alignment

There are exponentially many warping paths that satisfy the above conditions, however we are only interested in the path that minimizes the warping cost:

$$DTW(Q, C) = \min \left\{ \sqrt{\sum_{k=1}^K w_k} \right\} \quad (4)$$

This path can be found using dynamic programming to evaluate the following recurrence which defines the cumulative distance $\gamma(i, j)$ as the distance $d(i, j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements:

$$\gamma(i, j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \} \quad (5)$$

The Euclidean distance between two sequences can be seen as a special case of DTW where the k^{th} element of W

is constrained such that $w_k = (i, j)_k$, $i = j = k$. Note that it is only defined in the special case where the two sequences have the same length. The time and space complexity of DTW is $O(nm)$.

This review of DTW is necessarily brief; we refer the interested reader to [22, 27] for a more detailed treatment.

2.3 Related work

While there has been much work on indexing time series under the Euclidean metric [5, 10, 17, 18, 21, 34], there has been much less progress on indexing under DTW.

Yi, Jagadish, and Faloutsos [33] introduced a technique for approximate indexing of DTW that utilizes their FastMap technique [11]. The idea is to embed the sequences into Euclidean space such that the distances between them are approximately preserved, then classic multi-dimensional index structures can be utilized [14]. In addition, they introduce a lower bounding function (described in more detail in Section 3.2) that can be used to prune some of the inevitable false hits their method will introduce. The method does produce an observed (maximum) speedup of 7.8 over sequential scanning. However this does have some limitations. First, it does allow false dismissals. Second, while the time to build the index is linear in M (the size of the database), it is actually $O(Mn^2)$, which quickly becomes intractable for very large databases and/or long sequences.

Kim et al. introduced an exact algorithm for indexing of time series under DTW [19]. The method extracts four features from the sequences and organizes them in a multi-dimensional index structure. They introduce a lower bounding function (described in more detail in Section 3.2) that is defined on the four features and thus guarantee no false dismissals. Although the work introduced the first technique for exact indexing under DTW, it suffers from several limitations. First the method only allows the extraction of exactly four features, and thus cannot take advantage of multi-dimensional index structures that scale well to higher dimensions. In addition, although four features are extracted, only one of them (determined at query time) is actually used in the lower bounding function, thus the lower bound is very loose, and many false alarms are generated, each of which will require evaluation with the quadratic-time DTW algorithm.

In Park et al. [23] the authors demonstrate a DTW indexing technique which is based on a piecewise linear representation of the data. They “prove” that this method can guarantee no false dismissals. Unfortunately, the no false dismissals claim is incorrect. A candidate sequence in the database can differ from the query sequence by an arbitrarily small epsilon and still not be retrieved [26]. A later version of the paper did carry a disclaimer stating “...it is possible that a subsequence similar to a query in terms of the original time warping distance may not be included in the answer set in our approach” [24]. However this qualification understates the problem.

Having tested the approach with 39,200 experiments on 32 different datasets we found that the approach only returned the true best match to a 1-nearest neighbor query 613 times. This result does not significantly differ from random chance. We therefore exclude this approach from further consideration.

Another attempt at indexing DTW utilizes a suffix tree [25]. While the method is interesting, we do not include it in our empirical comparisons since the index size is one to two orders of magnitude larger than the data itself. Such enormous space overhead is simply untenable for very large databases. In any case, the claimed speed up is rather modest.

Finally there has been some work in which attempts at indexing and/or lower bounding are abandoned and instead efforts are concentrated on fast approximation of the DTW distance using a lower resolution approximation of the data. The idea was introduced by [6] who use a piecewise linear approximation of the data. The method shows significant speedup with few false dismissals. A similar idea was suggested by [5]. Here the authors obtain the lower resolution of the data approximation with wavelets and use their approximate distance measure instead of Yi et al’s lower bounding measure, within the FastMap framework. The method improves the speedup of Yi et al’s work at the expense of introducing more false dismissals.

3. Lower bounding the DTW distance

In this section we explain the importance of lower bounding, and introduce our new lower bounding distance measure.

3.1 The utility of lower bounding measures

Time series similarity search under the Euclidean metric is heavily I/O bound, however similarity search under DTW is also very demanding in terms of CPU time. One way to address this problem is to use a fast lower bounding function to help prune sequences that could not possibly be a best match. Table 2 gives the pseudocode for such an algorithm.

```

Algorithm Lower_Bounding_Sequential_Scan(Q)
1. best_so_far = infinity;
2. for all sequences in database
3.   LB_dist = lower_bound_distance( $C_i$ , Q);
4.   if LB_dist < best_so_far
5.     true_dist = DTW( $C_i$ , Q);
6.     if true_dist < best_so_far
7.       best_so_far = true_dist;
8.       index_of_best_match = i;
9.     endif
10.  endif
11. endfor

```

Table 2: An algorithm that uses a lower bounding distance measure to speed up the sequential scan search for the query Q

There are only two desirable properties of a lower bounding measure:

- It must be fast to compute. Clearly a measure that takes as long to compute as the original measure is of little use. In our case we would like the time complexity to be at most linear in the length of the sequences.
- It must be a relatively tight lower bound. A function can achieve a trivial lower bound by always returning zero as the lower bound estimate. However, in order for the algorithm in Table 2 to be effective, we require a method that more tightly approximates the true DTW distance.

While lower bounding functions for string edit, graph edit and tree edit distance have been studied extensively [22], there has been far less work on DTW, which is very similar in spirit to its discrete cousins. Below we will consider the existing DTW lower bounding techniques.

3.2 Existing lower bounding measures

To the best of our knowledge there are only two existing lower bounding functions available for DTW (Not including [23] which incorrectly claims to be lower bounding, or [25] which has a time complexity equal to the full algorithm). While referring the interested reader to the original papers for detailed explanations, below we give a visual intuition and brief explanation of each.

The lower bounding function introduced by Kim et al. [19] (hereafter known as LB_Kim), works by extracting a 4-tuple feature vector from each sequence. The features are the first and last elements of the sequence, together with the maximum and minimum values. The maximum absolute difference of corresponding features is reported as the lower bound. Figure 4 illustrates the idea.

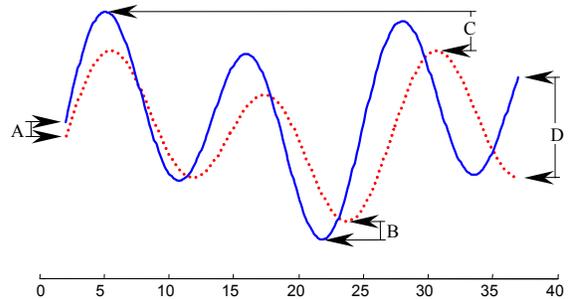


Figure 4: A visual intuition of the lower bounding measure introduced by Kim et al. The squared difference between the two sequence’s first (A), last (D), minimum (B) and maximum points (C) is returned as the lower bound

The lower bounding function introduced by Yi et al. [33]. (hereafter referred to as LB_Yi) takes advantage of the observation that all the points in one sequence that are larger (smaller) than the maximum (minimum) of the other sequence must contribute at least the squared

difference of their value and the maximum (minimum) value of the other sequence to the final DTW distance. Figure 5 illustrates the idea.

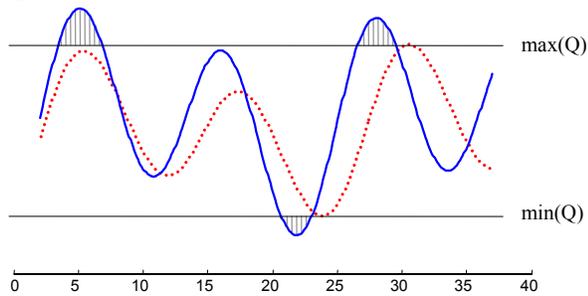


Figure 5: A visual intuition of the lower bounding measure introduced by Yi et al. The sum of the squared length of gray lines represent the minimum the corresponding points contribution to the overall DTW distance, and thus can be returned as the lower bounding measure

3.3 Proposed lower bounding measure

Before introducing our lower bounding technique we must review an additional detail of the DTW algorithm that we deliberately omitted until now.

3.3.1 Global constraints on time warping

In addition to the constraints on the warping path enumerated in Section 2.2, virtually all practitioners using DTW also constraint the warping path in a global sense by limiting how far it may stray from the diagonal [3]. The subset of matrix that the warping path is allowed to visit is called the warping window. Figure 6 illustrates two of the most frequently used global constraints, the Sakoe-Chiba Band and the Itakura Parallelogram [27, 29].

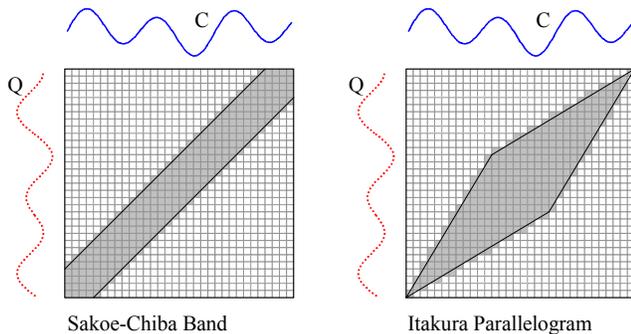


Figure 6: Global constraints limit the scope of the warping path, restricting them to the gray areas. The two most common constraints in the literature are the Sakoe-Chiba Band and the Itakura Parallelogram

There are several reasons for using global constraints, one of which is that they slightly speed up the DTW distance calculation. However the most important reason is to prevent pathological warpings, where a relatively small section of one sequence maps onto a relatively large section of another. The importance of global constraints was documented by the originators of the DTW

algorithm, who were exclusively interested in aligning speech patterns [29]. However, has been empirically confirmed in other settings, including finance, medicine, biometrics, chemistry, astronomy, robotics and industry.

As a motivating example consider the two sequences in Figure 1 which were used to illustrate DTW. The smooth peaks in each correspond to increase in demand for electrical power during weekdays. In the topmost sequence there is no peak on Monday because it was a national holiday, the same is true for Wednesday in the bottom sequence. In this domain, we may well decide that that it makes sense to allow warpings of up to one day, i.e. Monday may warp to Tuesday and Tuesday may warp to Wednesday etc, but more drastic warpings (i.e. Monday to Friday) should not be allowed. This constraint can easily be enforced by using a Sakoe-Chiba Band with a width equal to $n/7$.

3.3.2 Proposed lower bounding measure

We can view a global constraint as constraining the indices of the warping path $w_k = (i, j)_k$ such that $j-r \leq i \leq j+r$ where r is a term defining the *reach*, or allowed range of warping, for a given point in a sequence. In the case of the Sakoe-Chiba Band r is independent of i , for the Itakura Parallelogram r is a function of i .

We will use the term r to define two new sequences, U and L :

$$U_i = \max(q_{i-r} : q_{i+r}) \quad (6)$$

$$L_i = \min(q_{i-r} : q_{i+r}) \quad (7)$$

U and L stand for *Upper* and *Lower* respectively, we can see why if we plot them together with the original sequence Q as in Figure 7. They form a bounding envelope that encloses Q from above and below. Note that although the Sakoe-Chiba Band is of constant width, the corresponding envelope generally is not of uniform thickness. In particular, the envelope is wider when the underlying query sequence is changing rapidly, and narrower when the query sequence plateaus.

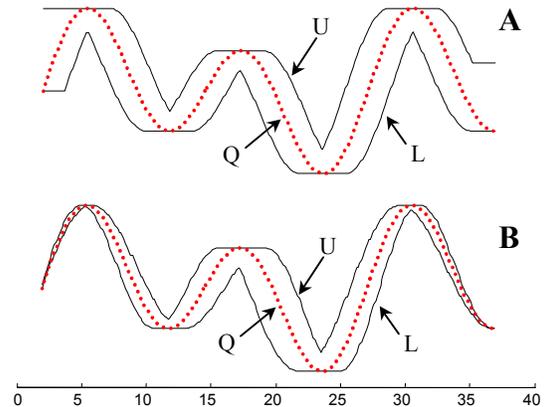


Figure 7: An illustration of the sequences U and L , created for sequence Q (shown dotted). **A** was created using the Sakoe-Chiba Band and **B** using the Itakura Parallelogram

An obvious but important property of U and L is the following:

$$\forall_i \quad U_i \geq q_i \geq L_i \quad (8)$$

Having defined U and L , we now use them to define a lower bounding measure for DTW.

$$LB_Keogh(Q,C) = \sqrt{\sum_{i=1}^n \begin{cases} (c_i - U_i)^2 & \text{if } c_i > U_i \\ (c_i - L_i)^2 & \text{if } c_i < L_i \\ 0 & \text{otherwise} \end{cases}} \quad (9)$$

This function can be readily visualized as the Euclidean distance between the any part of the candidate matching sequence not falling within the envelope and the nearest (orthogonal) corresponding section of the envelope. Figure 8 illustrates the idea.

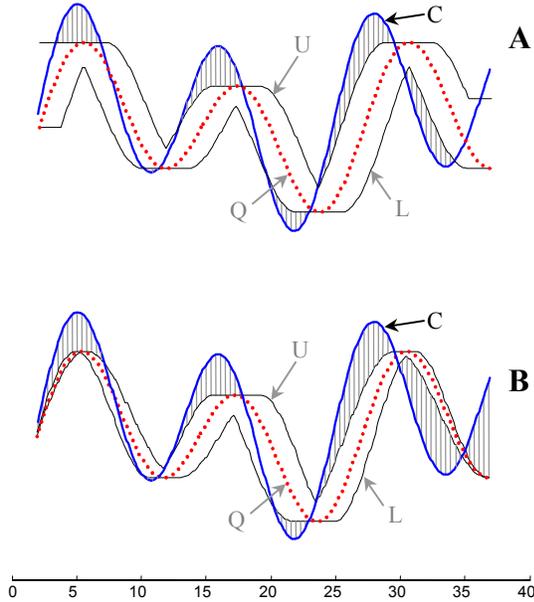


Figure 8: An illustration of the lower bounding function $LB_Keogh(Q,C)$. The original sequence Q (shown dotted), is enclosed in the bounding envelope of U and L . The squared sum of the distances from every part of the candidate sequence C not falling within the bounding envelope, to the nearest orthogonal edge of the bounding envelope is returned as the lower bound. Bounding envelope **A** was created using the Sakoe-Chiba Band and bounding envelope **B** using the Itakura Parallelogram

Since the tightness of the bounds is proportional to the number and length of the gray hatch lines, we can see that in this example at least, that the Itakura Parallelogram provides a tighter bound than the Sakoe-Chiba Band, and both appear tighter than LB_Kim or LB_Yi in Figures 4 and 5 respectively.

We will now prove the claim of lower bounding.

Proposition 1: For any two sequences Q and C of the same length n , for any global constraint on the warping path of the form $j-r \leq i \leq j+r$, the following inequality holds: $LB_Keogh(Q,C) \leq DTW(Q,C)$

Proof: We wish to prove

$$\sqrt{\sum_{i=1}^n \begin{cases} (c_i - U_i)^2 & \text{if } c_i > U_i \\ (c_i - L_i)^2 & \text{if } c_i < L_i \\ 0 & \text{otherwise} \end{cases}} \leq \sqrt{\sum_{k=1}^K w_k}$$

Since the terms under radicals are positive, we can square both sides:

$$\sum_{i=1}^n \begin{cases} (c_i - U_i)^2 & \text{if } c_i > U_i \\ (c_i - L_i)^2 & \text{if } c_i < L_i \\ 0 & \text{otherwise} \end{cases} \leq \sum_{k=1}^K w_k$$

From Eq. 3, we know that $n \leq K$, so our strategy will be to show that every term in the left summation can be matched with some greater or equal term in the right summation.

There are three cases to consider, for the moment we will just consider cases when $c_i > U_i$. We want to show:

$$\begin{aligned} (c_i - U_i)^2 &\leq w_k \\ (c_i - U_i)^2 &\leq (c_i - q_j)^2 && \text{By definition (cf Section 2.2).} \\ (c_i - U_i) &\leq (c_i - q_j) && \text{Since } c_i > U_i, \text{ we can take square roots} \\ -U_i &\leq -q_j && \text{Add } -c_i \text{ to both sides.} \\ q_j &\leq U_i && \text{Add } U_i + q_j \text{ to both sides.} \\ q_j &\leq \max(q_{i-r}, q_{i+r}) && \text{By definition, Eq. 6.} \end{aligned}$$

Since we have $n = m$, then $j-r \leq i \leq j+r$, $\Rightarrow i-r \leq j \leq i+r$, so we can rewrite the right hand side as

$$q_j \leq \max(q_{i-r}, q_{(i+1)-r}, q_j, \dots, q_{i+r})$$

If we remove all terms except q_j from the RHS we are left with $q_j \leq \max(q_j)$ Which is obviously true.

The case $c_i < L_i$ yields to a similar argument. The final case is simple to show, since clearly

$$0 \leq (c_i - q_j)^2 \quad \text{Because } (c_i - q_j)^2 \text{ must be nonnegative}$$

Thus we have shown that each term on the left side is matched with an equal or larger term on the right side, our inequality holds, so $LB_Keogh(Q,C) \leq DTW(Q,C)$. ■

In the next section we will show how LB_Keogh can be indexed.

4. Indexing DTW

Virtually all approaches to indexing time series under the Euclidean distance that guarantee no false dismissals use the GEMINI framework of Faloutsos et al. [5, 10, 17, 18, 21, 34]. Using the GEMINI framework all one has to do is to choose a high level representation of the data and define a lower bounding measure on it. Many such representations have been suggested, including Fourier Transforms [11], Wavelets [5], Singular Value Decomposition [21], Adaptive Piecewise Constant Approximation [18] and a simple technique independently introduced by two authors called Piecewise Constant Approximation (PAA) [17, 34]. This technique is attractive because it is simple, intuitive and competitive with the other, more complex approaches. In this section we will show that PAA can be adapted to allow indexing under DTW. We begin with a brief review of PAA.

4.1 Piecewise Constant Approximation

We have previously denoted a time series as $C = c_1, \dots, c_n$. We assume each sequence in our database is n units long. Let N be the dimensionality of the space we wish to index ($1 \leq N \leq n$). For convenience, we assume that N is a factor of n . This is not a requirement of our approach, however it does simplify notation.

A time series C of length n can be represented in N dimensional space by a vector $\bar{C} = \bar{c}_1, \dots, \bar{c}_N$. The i^{th} element of \bar{C} is calculated by the following equation:

$$\bar{c}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} c_j \quad (10)$$

Simply stated, to reduce the time series from n dimensions to N dimensions, the data is divided into N equal sized ‘‘frames’’. The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation. The complicated subscripting in Eq. 10 just insures that the original sequence is divided into the correct number and size of frames. The representation can best be visualized as an attempt to model the original time series with a linear combination of box basis functions as shown in Figure 9.

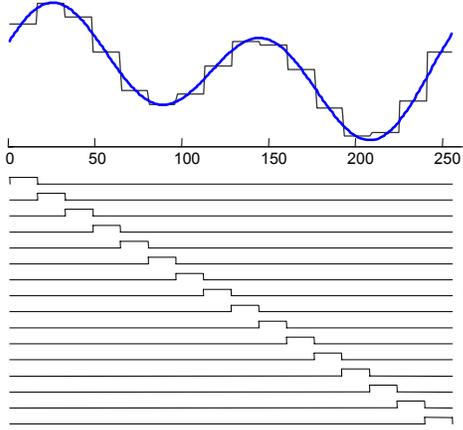


Figure 9: The PAA representation can be readily visualized as an attempt to model a sequence with a linear combination of box basis functions. In this case, a sequence of length 256 is reduced to 16 dimensions

Given two original sequences Q and C , we can transform them into \bar{Q} and \bar{C} using Eq. 10, and approximate their Euclidean distance by:

$$DR(\bar{Q}, \bar{C}) \equiv \sqrt{\frac{n}{N} \sum_{i=1}^N (\bar{q}_i - \bar{c}_i)^2} \quad (11)$$

A proof that $DR(\bar{Q}, \bar{C})$ lower bounds the true Euclidean distance is in [17] (A different proof appears in[34]).

4.2 Modifying PAA to index time warped queries

In Section 3 we introduced the lowering bounding function LB_Keogh , however calculating this function

requires n values. Since n may be in the order of hundreds to thousands, and multi-dimensional index structures begin to degrade rapidly somewhere above 16 dimension [15, 31], we need a way to create a lower, N dimension version of the function, where N is a number that can be reasonably handled by a multi-dimensional index structure [14]. We also need this lower dimension version of the function to lower bound LB_Keogh (and therefore, by transitivity, DTW).

We begin by creating special piecewise constant approximations of U and L , which we will denote \hat{U} and \hat{L} . Although they are piecewise constant approximations, the definitions of \hat{U} and \hat{L} differ from those we have seen in Eq. 10, in particular we have

$$\hat{U}_i = \max\left(U_{\frac{n}{N}(i-1)+1}, \dots, U_{\frac{n}{N}i}\right) \quad (11)$$

$$\hat{L}_i = \min\left(L_{\frac{n}{N}(i-1)+1}, \dots, L_{\frac{n}{N}i}\right) \quad (12)$$

We can visualize \hat{U} and \hat{L} as the piecewise constant functions which bound, without intersecting, U and L respectively. Figure 10 illustrates this intuition.

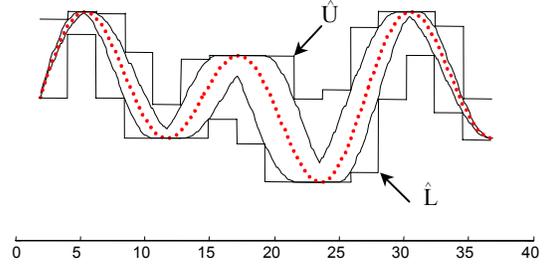


Figure 10: We can readily visualize \hat{U} and \hat{L} as the piecewise constant functions which bound, without intersecting, U and L respectively

We are now able to define the low dimension, lower bounding function, which we denote LB_PAA . Given a candidate sequence C , transformed to \bar{C} by Eq. 10, and a query sequence Q , with its companion PAA functions \hat{U} and \hat{L} , the following function lower bounds LB_Keogh

$$LB_PAA(Q, \bar{C}) = \sqrt{\sum_{i=1}^N \frac{n}{N} \begin{cases} (\bar{c}_i - \hat{U}_i)^2 & \text{if } \bar{c}_i > \hat{U}_i \\ (\bar{c}_i - \hat{L}_i)^2 & \text{if } \bar{c}_i < \hat{L}_i \\ 0 & \text{otherwise} \end{cases}} \quad (13)$$

The proof that $LB_PAA(Q, \bar{C}) \leq LB_Keogh(Q, C)$ is a straightforward but long extension of Proposition 1, we omit it for brevity.

The final step necessary to allow indexing is to define a $MINDIST(Q, R)$ function that returns a lower bounding measure of the distance between a query Q , and R , were R is a Minimum Bounding Rectangle (MBR).

Suppose our index structure contains a leaf node U . Let $R = (L, H)$ be the MBR associated with U where $L = \{l_1, l_2, \dots, l_N\}$ and $H = \{h_1, h_2, \dots, h_N\}$ are the lower and higher endpoints of the major diagonal of R . By

definition, R is the smallest rectangle that spatially contains each PAA point $\bar{C} = \bar{c}_1, \dots, \bar{c}_N$ stored in U . Given the above, $MINDIST(Q, R)$ is defined as:

$$MINDIST(Q, R) = \sqrt{\sum_{i=1}^N \frac{n}{N} \begin{cases} (l_i - \hat{U}_i)^2 & \text{if } l_i > \hat{U}_i \\ (h_i - \hat{L}_i)^2 & \text{if } h_i < \hat{L}_i \\ 0 & \text{otherwise} \end{cases}} \quad (14)$$

This function is visualized in Figure 11.

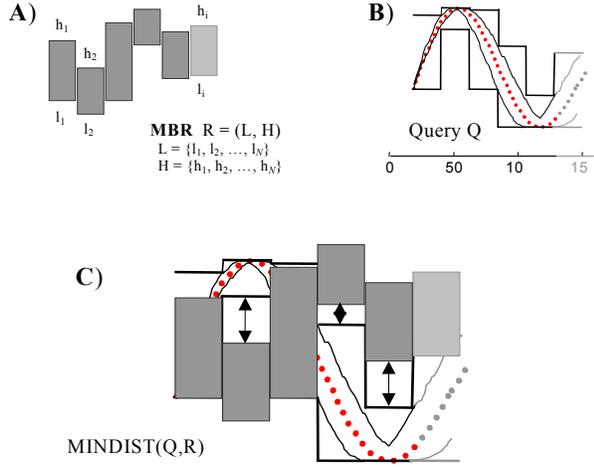


Figure 11: A) A representation of a Minimum Bounding rectangle (MBR). B) A subsection of the query shown in Figure 10, with its attendant functions \hat{U} and \hat{C} . C) An illustration of the MINDIST function. The lengths of the arrow lines, squared, scaled by n/N , summed and square rooted, are returned as the minimum distance between Q and any sequence contained within R

Having defined LB_PAA and $MINDIST(Q, R)$ we are now ready to introduce the K -Nearest Neighbor search algorithm. The basic algorithm is shown in Table 3, it is an optimization on the GEMINI K -NN algorithm [11] as suggested by [31], and is a modification of the algorithm used for indexing time series under the Euclidean metric in [18].

A query $KNNSearch(Q, K)$ with query sequence Q and desired number of neighbors K retrieves a set C of K time series such that for any two sequences $C \in C, E \notin C, DTW(Q, C) \leq DTW(Q, E)$. Like the classic K -NN algorithm [28], the algorithm in Table 3 uses a priority queue $queue$ to visit nodes/objects in the index in the increasing order of their distances from Q in the indexed (i.e. PAA) space. The distance of an object (i.e. PAA point) C from Q is defined by $LB_PAA(Q, \bar{C})$ (cf. Section 4.2, Eq. 13) while the distance of a node U from Q is defined by the minimum distance $MINDIST(Q, R)$ of the minimum bounding rectangle (MBR) R associated with U from Q .

We begin by pushing the root node of the index into the $queue$ (Line 1). The algorithm navigates the index by popping out the item from the top of $queue$ at each step

(Line 8). If the popped item is an PAA point C , we go to disk to retrieve the original time series C , and we compute its exact distance $DWT(Q, C)$ from the query then insert it into a temporary list $temp$ (Lines 9-11). If, on the other hand, the popped item is a node of the index structure, we compute the distance of each of its children from Q and push them into $queue$ (Lines 12-17).

Algorithm $KNNSearch(Q, K)$	
Variable	$queue$: MinPriorityQueue;
Variable	$list$: $temp$;
1.	$queue.push(\text{root_node_of_index}, 0)$;
2.	while not $queue.IsEmpty()$ do
3.	$top = queue.Top()$;
4.	for each time series C in $temp$ such that $DTW(Q, C) \leq top.dist$
5.	Remove C from $temp$;
6.	Add C to $result$;
7.	if $ result = K$ return $result$;
8.	$queue.Pop()$;
9.	if top is an PAA point C
10.	Retrieve full sequence C from database;
11.	$temp.insert(C, DTW(Q, C))$;
12.	else if top is a leaf node
13.	for each data item C in top
14.	$queue.push(C, LB_PAA(Q, \bar{C}))$;
15.	else // top is a non-leaf node
16.	for each child node U in top
17.	$queue.push(U, MINDIST(Q, R))$ // R is MBR associated with U .

Table 3: K -NN algorithm to compute the exact K nearest neighbors of a query time series Q using a multidimensional index structure

We only move a sequence C from $temp$ to $result$ when we are sure that it is one of the K nearest neighbors of Q . That is to say, there exists no object $E \notin result$ such that $DTW(Q, E) < DTW(Q, C)$ and $|result| < K$. This second condition is guaranteed by the exit condition in Line 7. The first condition can be guaranteed as follows. Let I be the set of PAA points retrieved thus far using the index (i.e. $I = temp \cup result$). If we can guarantee that $\forall C \in I, \forall E \notin I, LB_PAA(Q, \bar{C}) \leq DTW(Q, E)$, then the condition “ $DTW(Q, C) \leq top.dist$ ” in Line 4 will ensure that there exists no unexplored sequence E such that $DTW(Q, E) < DTW(Q, C)$.

By inserting the time series in $temp$ (i.e. previously seen objects) into $result$ in increasing order of their distances $DTW(Q, C)$ (by keeping $temp$ sorted by $DTW(Q, C)$), we ensure that there exists no explored object E such that $DTW(Q, E) < DTW(Q, C)$.

The definitions of LB_Keogh , LB_PAA and $MINDIST$ proposed in this work are also needed for answering range queries using a multidimensional index structure. We can use a classic R -tree-style recursive search algorithm. Since both $MINDIST(Q, R)$ and $LB_PAA(Q, \bar{C})$ lower bound $DTW(Q, C)$, the algorithm shown in Table 4 is correct [11].

```

Algorithm RangeSearch(Q,  $\epsilon$ , T)
1. if T is a non-leaf node
2.   for each child U of T
3.     if MINDIST(Q,R)  $\leq \epsilon$    RangeSearch(Q,  $\epsilon$ , U);
                                   // R is MBR of U
4.   else                               // T is a leaf node
5.     for each PAA point C in T
6.       if LB_PAA(Q, C)  $\leq \epsilon$ 
7.         Retrieve full sequence C from database;
8.       if DTW(Q,C)  $\leq \epsilon$  Add C to result;

```

Table 4: Range search algorithm to retrieve all the time series within a range of ϵ from query time series Q. The function is invoked as RangeSearch(Q, ϵ , root_node_of_index)

5. Experimental Evaluation

In this section we test our proposed approach with a comprehensive set of experiments.

5.1 Experimental Philosophy

Previous experience in reimplementing and testing more than a dozen different Euclidean time series indexing techniques [18], suggests that many published results do not generalize to real world datasets and conditions. We therefore conducted the experiments in this paper with the explicit goal of conducting the most comprehensive and detailed set of time series indexing experiments ever attempted. In particular we have taken the following steps to insure the most meaningful and generalizable results.

- Instead of testing on just one or two datasets as is typical [2, 3, 5, 7, 8, 10, 12, 13, 16, 19, 23, 24, 25, 30 34], we tested all algorithms on 32 datasets. These datasets cover the complete spectrum of stationary/ non-stationary, noisy/ smooth, cyclical/ non-cyclical, symmetric/ asymmetric, etc. The data also represents the many areas in which DTW is used, including finance, medicine, biometrics, chemistry, astronomy, robotics, networking and industry.
- We designed our experiments to be completely reproducible. We saved every random number, every setting and all data, and have made them available on a free CD Rom.

- To ensure true randomness where required, we use random numbers created by a quantum mechanical process [32].
- Although we also present results of an implemented system, we present comprehensive results that are completely of implementation (ie. page size, cache size, etc). This is to guard against implementation bias [18] and to allow and encourage independent replication of our results.

For simplicity and brevity we only show results for nearest neighbor queries, however we obtained very similar results for range queries. Because of the sheer volume of experiments conducted, in this section we will present graphics to summarize our findings and we will reproduce the actual numbers in Appendix A.

For all experiments we used the Sakoe-Chiba Band with a width of 10% of n , since this appears to be the most commonly used constraint in the literature [27, 29].

5.2 Comparison of lower bounding functions

We begin our experiments with a comparison of the tightness of the lower bounds for the three functions, LB_Yi, LB_Kim and LB_Keogh. We define T as the ratio of the estimated distance between two sequences over the true distance between the same two sequences.

$$T = \frac{\text{Lower Bound Estimate of Dynamic Time Warp Distance}}{\text{True Dynamic Time Warp Distance}} \quad (15)$$

T is in the range [0,1], with the larger the better. To estimate T for each of the 32 datasets we did the following: We randomly extracted 50 sequences of length 256. We compared each sequence to the 49 others, using the true DTW distance, and the three lower bounding functions. For each dataset we report T as average ratio from the 1,225 (50*49/2) comparisons made.

Figure 12 shows the results of the experiments. On 24 out of 32 datasets LB_Yi produces tighter bounds than LB_Kim, and its average value is approximately 1.38 times larger. The most obvious result from the experiment however is the dominance of LB_Keogh. It wins on every dataset and its average value is approximately 3.11 times larger than its nearest rival. Since the efficiency of indexing has a (much) greater than linear dependence on the tightness of the lower bounding function, these results augur well for our approach.

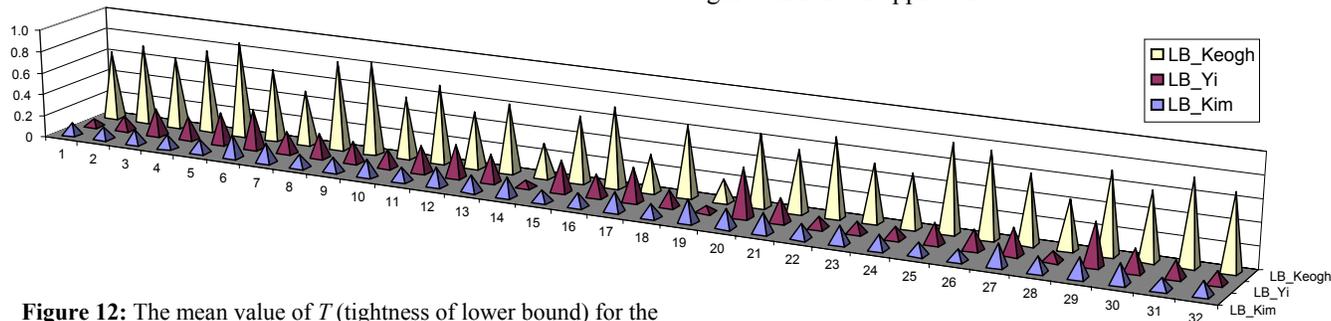


Figure 12: The mean value of T (tightness of lower bound) for the three lower bounding functions under consideration, for 32 datasets from finance, medicine, biometrics, chemistry, physics, astronomy, robotics, networking and industry. Appendix A contains a key to the datasets

We choose to report results from a query length of 256, since this is about the mid range of queries reported in the literature [5, 6, 33]. However we also experimented with queries in the range of 32 to 1024. This range was chosen to include the longest and shortest reported in the literature [5, 23, 33]. All techniques perform better for short queries, however while both LB_Kim and LB_Yi degrade rapidly for longer queries, LB_Keogh stays almost constant for longer queries. This effect was observed on all datasets, for brevity we just present results for the random walk dataset in Figure 13.

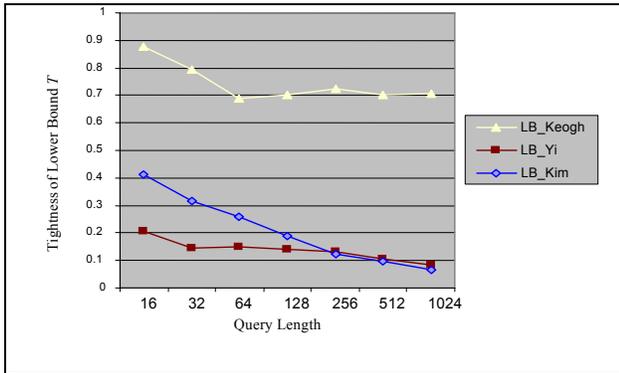


Figure 13: The effect of query length on the tightness of lower bounds for the three techniques under consideration

5.3 Comparison of pruning power

To compare the pruning power of the three techniques under consideration, we measure P , the fraction of the database that does *not* require full computation of DTW while still allowing use to guarantee that we have found the nearest match to a 1-NN query.

$$P = \frac{\text{Number of objects that do not require full DTW}}{\text{Number of objects in database}} \quad (16)$$

To calculate P we do the following. From each of the 32 datasets we randomly extract 50 sequences of length 256. For each of the 50 sequences we do the following, we separate out the sequence from the other 49 sequences. We then find the nearest match to our withheld sequence among the remaining 49 sequences using the sequential scan algorithm of Table 2. We measure the number of times we can use the linear-time lower bounding functions to prune away the quadratic-time computation of the full DTW algorithm. For fairness we visit the 49 sequences in the same order for each approach. The value

P reported is averaged over all 50 runs.

Note the value of P for any depends only on the data and is completely independent of any implementation choices, including spatial access method, buffer size, computer language or hardware platform. A similar idea for evaluating indexing schemes appears in [15].

The results are summarized in Figure 14. On 25 out of 32 datasets LB_Yi is more efficient at pruning than LB_Kim, on average it was able to prune 1.53 times as many items. Once again however, the most obvious result is the dominance of LB_Keogh. It wins on every dataset and was able to prune 3.95 times as many items as LB_Yi and 6.06 times as many items as LB_Kim.

Note that while these results are powerful implementation independent predictors of indexing performance, they may actually be pessimistic. There are two related reasons why. First, the sequential scan algorithm of Table 2 is inefficient, as it visits the items and calculates the DTW measures (where necessary) in a predefined order. A more efficient implementation would sort, and then visit the sequences, in ascending order of the lower bounding distance. This of course, is essentially what spatial indexing does.

The second reason why the results may be pessimistic predictors of indexing performance is the relatively small size of the datasets. We should expect the fraction of pruned sequences to increase on larger datasets. The reason is because the larger the dataset, the greater the chance there is of a good match being found, and a good match allows us to extract the maximum benefit from the pruning conditional $LB_dist < best_so_far$ in line 4 of the algorithm. To demonstrate this effect we ran the same experiment above on increasing larger subsets of the random walk dataset. The results are shown in Figure 15.

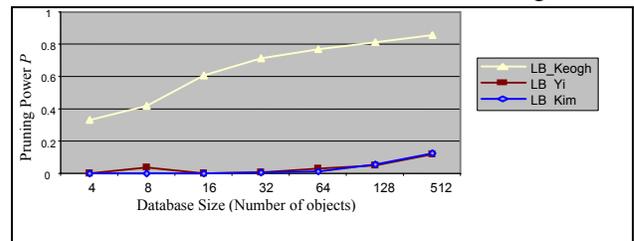


Figure 15: The effect of database size on pruning power. Note that as the size of the database increases we are able to prune a larger fraction of the data

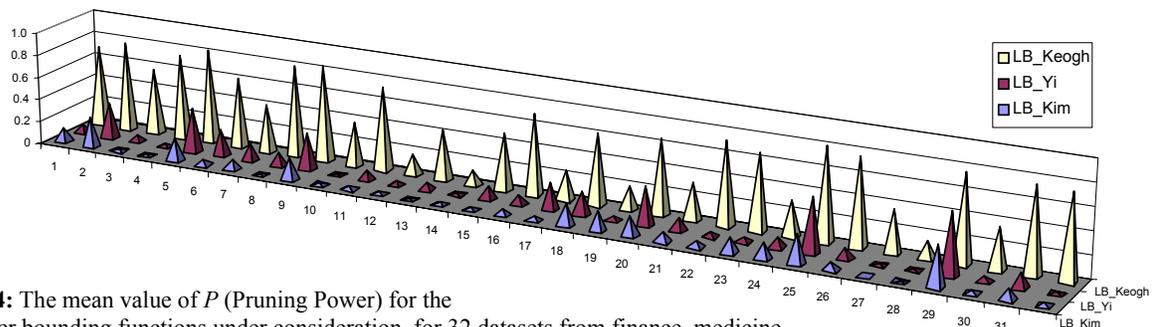


Figure 14: The mean value of P (Pruning Power) for the three lower bounding functions under consideration, for 32 datasets from finance, medicine, biometrics, chemistry, physics, astronomy, robotics, networking and industry. Appendix A contains a key to the datasets

5.4 Experiments on an implemented system

The 32 datasets used in the previous experiments illustrate the dominance of the proposed approach on a wide variety of datasets. However, most are not large enough by themselves to warrant the title “Very Large Data Base”. We therefore pooled all 32 datasets into a single dataset that we call Mixed Bag (MB). In addition to this ultra heterogeneous data, we created a very large database of Random Walk data (RW II), since this is the most studied dataset for indexing comparisons [5, 6, 17, 24, 25, 34] and is, by contrast with the above, a very homogeneous dataset. Details of these datasets appear in Appendix A.

We performed experiments on AMD Athlon 1.4 GHZ processor, with 512 MB of physical memory and 57.2 GB of secondary storage. The spatial access method used was the R-Tree [14].

To evaluate the performance of the proposed technique we used the normalized CPU cost.

Definition: *The Normalized CPU cost:* The ratio of average CPU time to execute a query using the index to the average CPU time required to perform a linear (sequential) scan. The normalized cost of linear scan is 1.0.

Beating linear scan is nontrivial because it can take advantage of sequential disk access, whereas any indexing technique must make random disk accesses. It is generally understood that random access is about ten times slower than sequential access [15, 28, 31]. For fairness, we allowed linear scan to utilize the lower bounding function LB_Keogh.

Because there is no known exact indexing method for LB_Yi, we could not include it in this experiment. We originally included LB_Kim in the experiments, but found that it never beat linear scan, we therefore decided to exclude it from graphic presentation.

We tested over a range of query lengths and dimensionalities, but show just one typical result for brevity. Figure 15 shows the normalized CPU cost of linear scan and LB_Keogh, for queries of length 256, with a 16 dimensional index, for increasingly large databases.

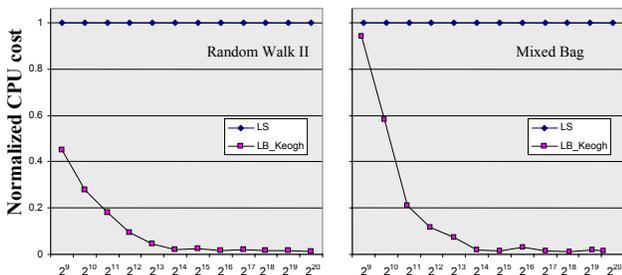


Figure 15: The normalized CPU cost of linear scan and LB_Keogh, for queries of length 256, with a 16 dimensional index, for increasingly large databases. Note that the X-axis is in logarithmic scale, and denotes the number of items in the database

6. Discussion and Conclusions

In one of the most referenced papers on time series similarity ever published [2], the authors explicitly state, “*Dynamic time warping...cannot be speeded up by indexing*”. This sentiment has since been echoed in several dozen other papers [6, 33]. How then have we achieved the seemingly impossible? Firstly, we have only considered the case where the two sequences are of the same length. This is not really a limitation because the user can always re-interpolate the query to any desired length in $O(n)$ time. Secondly, we can only index sequences if we assume the warping path is constrained. Once again we feel that this is not really a restriction since virtually ever practitioner we are aware of reiterates the absolute necessity of using constraints [3, 6, 27, 29, 30].

Our approach is particularly attractive since as a special case (r is set to zero) it degenerates to Euclidean indexing using PAA, an approach that has been shown by two independent groups of researchers to be state of the art in terms of efficiency and flexibility [17, 34].

There are several directions in which we intend to extend this work. For example, we note that some algorithms for matching 2 and 3 dimensional shapes are very close analogues of the DTW algorithm, and thus may benefit from a similar lower bounding function.

Acknowledgements: Thanks to Kaushik Chakrabarti, Dennis DeCoste, Sharad Mehrotra, Michalis Vlachos and the VLDB reviewers for their useful comments.

References

- [1] Aach, J. and Church, G. (2001). Aligning gene expression time series with time warping algorithms. *Bioinformatics*. Volume 17, pp 495-508.
- [2] Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In *Proc. 21st Int. Conf. on Very Large Databases*, pp. 490-501.
- [3] Berndt, D. & Clifford, J. (1994) Using dynamic time warping to find patterns in time series. *AAAI-94 Workshop on Knowledge Discovery in Databases*. pp 229-248.
- [4] Caiani, E.G., Porta, A., Baselli, G., Turiel, M., Muzzupappa, S., Pieruzzi, F., Crema, C., Malliani, A. & Cerutti, S. (1998) Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. *IEEE Computers in Cardiology*.
- [5] Chan, K.P., Fu, A & Yu, C.(2002). Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, to appear.
- [6] Chu, S., Keogh, E., Hart, D., Pazzani, M (2002). Iterative deepening dynamic time warping for time series. In *Proc 2nd SIAM International Conference on Data Mining*.
- [7] Das, G., Lin, K., Mannila, H., Renganathan, G. & Smyth, P. (1998). Rule discovery form time series. *Proc. of the 4th International Conference of Knowledge Discovery and Data Mining*. pp 16-22, AAAI Press.

[8] Debregeas, A. & Hebrail, G. (1998). Interactive interpretation of Kohonen maps applied to curves. *Proc. of the 4th International Conference of Knowledge Discovery and Data Mining*. pp 179-183.

[9] Diez, J. J. R. & Gonzalez, C. A. (2000). Applying boosting to similarity literals for time series Classification. *Multiple Classifier Systems, 1st Inter' Workshop*. pp 210-219.

[10] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Conf.*, Minneapolis.

[11] Faloutsos, C., Lin, K. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Conf* pp 163-174.

[12] Gavrilu, D. M. & Davis, L. S. (1995). Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *International Workshop on Automatic Face- and Gesture-Recognition*.

[13] Gollmer, K., & Posten, C. (1995) Detection of distorted pattern using dynamic time warping algorithm and application for supervision of bioprocesses. *On-Line Fault Detection and Supervision in Chemical Process Industries*.

[14] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings ACM SIGMOD Conference*. pp 47-57.

[15] Hellerstein, J. M., Papadimitriou, C. H., & Koutsoupias, E. (1997). Towards an analysis of indexing schemes. *16th ACM Symposium on Principles of Database Systems*.

[16] Kadous, M. W. (1999) Learning comprehensible descriptions of multivariate time series. In *Proc. of the 16th International Machine Learning Conference*.

[17] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra (2000) Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*. pp 263-286.

[18] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra (2001) Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc of ACM SIGMOD Conference on Management of Data*, May. pp 151-162.

[19] Kim, S., Park, S., & Chu, W. (2001). An Index-based approach for similarity search supporting time warping in large sequence databases. In *Proc 17th International Conference on Data Engineering*, pp 607-614.

[20] Kollios, G., Vlachos, M. & Gunopulos, G. (2002). Discovering similar multidimensional trajectories. In *Proc 18th International Conference on Data Engineering*.

[21] Korn, F., Jagadish, H & Faloutsos. C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proceedings of SIGMOD '97*. pp 289-300.

[22] Kruskal, J. B. & Liberman, M. (1983). The symmetric time warping algorithm: From continuous to discrete. In *Time Warps, String Edits and Macromolecules*. Addison-Wesley.

[23] Park, S., Lee, D., & Chu, W. (1999). Fast retrieval of similar subsequences in long sequence databases. In *3rd IEEE Knowledge and Data Engineering Exchange Workshop*.

[24] Park, S., Kim, S, & Chu, W. (2001). Segment-based approach for subsequence searches in sequence databases. In *Proceedings of the 16th ACM Symposium on Applied Computing*, pp. 248-252, Las Vegas, NV, USA.

[25] Park, S., Chu, W., Yoon, J & Hsu, C. (2000). Efficient searches for similar subsequences of different lengths in sequence databases. In *Proc. 16th IEEE Int'l Conf. on Data Engineering*, pp. 23-32.

[26] Park, S. (1999). Personal communication.

[27] Rabiner, L. & Juang, B. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, N.J, Prentice Hall.

[28] Roussopoulos, N., Kelley, S. & Vincent, F. (1995). Nearest neighbor queries. *SIGMOD Conference*. pp 71-79.

[29] Sakoe, H. & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-26.

[30] Schmill, M., Oates, T. & Cohen, P. (1999). Learned models for continuous planning. In *7th International Workshop on Artificial Intelligence and Statistics*.

[31] Seidl, T. & Kriegel, H. (1998). Optimal multi-step k-nearest neighbor search. *SIGMOD Conference*. pp 154-165.

[32] Walker, J. (2001). HotBits: Genuine random numbers generated by radioactive decay. www.fourmilab.ch/hotbits/

[33] Yi, B, K. Jagadish, H & Faloutsos (1998). Efficient retrieval of similar time sequences under time warping. In *ICDE 98*, pp 23-27.

[34] Yi, B, K., & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary L_p norms. *Proceedings of the 26th Intl Conference on Very Large Databases*. pp 385-394 .

Appendix A

ID	Name	Size	T (Tightness of Lower Bound)			P (Pruning Power)		
			LB_Kim	LB_Yi	LB_Keogh	LB_Kim	LB_Yi	LB_Keogh
1	Sunspot	2,899	0.11	0.06	0.63	0.14	0.07	0.73
2	Power	35,040	0.12	0.13	0.73	0.27	0.32	0.80
3	ERP data	198,400	0.13	0.24	0.65	0.01	0.07	0.59
4	Spot Exrates	2,567	0.12	0.21	0.75	0.01	0.03	0.77
5	Shuttle	6,000	0.12	0.29	0.87	0.20	0.39	0.85
6	Water	6,573	0.22	0.36	0.66	0.05	0.24	0.64
7	Chaotic	1,800	0.18	0.19	0.50	0.09	0.16	0.43
8	Steamgen	38,400	0.11	0.22	0.81	0.00	0.11	0.82
9	Ocean	4,096	0.13	0.19	0.84	0.20	0.34	0.87
10	Tide	8,746	0.16	0.16	0.56	0.02	0.01	0.39
11	CSTR	22,500	0.13	0.25	0.71	0.04	0.09	0.75
12	Winding	17,500	0.17	0.29	0.51	0.03	0.04	0.19
13	Dryer2	5,202	0.15	0.25	0.62	0.01	0.07	0.46
14	Robot Arm	2,048	0.18	0.06	0.30	0.03	0.01	0.13
15	Ph Data	6,003	0.11	0.29	0.60	0.03	0.12	0.51
16	Power Plant	2,400	0.13	0.20	0.72	0.05	0.08	0.72
17	Evaporator	37,830	0.18	0.31	0.34	0.04	0.25	0.26
18	Ballbeam	2,000	0.12	0.15	0.65	0.20	0.21	0.63
19	Tongue	700	0.20	0.06	0.21	0.17	0.04	0.21
20	Fetal ECG	22,500	0.17	0.45	0.66	0.17	0.35	0.67
21	Balloon	4,002	0.18	0.22	0.55	0.09	0.12	0.33
22	Stand' & Poor	17,610	0.13	0.10	0.71	0.06	0.04	0.75
23	Speech	1,020	0.16	0.11	0.53	0.14	0.04	0.69
24	Soil Temp	2,304	0.14	0.11	0.48	0.13	0.08	0.32
25	Wool	2,790	0.11	0.19	0.79	0.26	0.48	0.83
26	Infrasound	8,192	0.10	0.18	0.76	0.07	0.09	0.78
27	Network	18,000	0.14	0.18	0.55	0.00	0.01	0.38
28	EEG	11,264	0.14	0.08	0.44	0.01	0.01	0.16
29	Koski EEG	144,002	0.18	0.39	0.73	0.36	0.54	0.78
30	Buoy Sensor	55,964	0.17	0.20	0.61	0.03	0.06	0.38
31	Burst	9,382	0.10	0.15	0.77	0.09	0.12	0.76
32	Random Walk I	65,536	0.13	0.11	0.68	0.03	0.02	0.75
	Mean Value		0.144	0.199	0.622	0.094	0.144	0.572
MB	Mixed Bag	763,270						
RW	Random Walk II	1,048,576						

Table 5: The raw numbers obtained for the experiments discussed in Sections 5.2 and 5.3. These numbers may be visualized in Figures 12 and 14 respectively