Machine Learning, 58, 231–267, 2005
2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Fast and Exact Warping of Time Series Using Adaptive Segmental Approximations

YUTAO SHOU ytshou@cs.hku.hk
NIKOS MAMOULIS nikos@cs.hku.hk
DAVID W. CHEUNG dcheung@cs.hku.hk
Department of Computer Science, University of Hong Kong, Pokfulam Road, Hong Kong

Editor: Eamonn Keogh

Abstract. Similarity search is a core module of many data analysis tasks, including search by example, classification, and clustering. For time series data, Dynamic Time Warping (DTW) has been proven a very effective similarity measure, since it minimizes the effects of shifting and distortion in time. However, the quadratic cost of DTW computation to the length of the matched sequences makes its direct application on databases of long time series very expensive. We propose a technique that decomposes the sequences into a number of segments and uses cheap approximations thereof to compute fast lower bounds for their warping distances. We present several, progressively tighter bounds, relying on the existence or not of warping constraints. Finally, we develop an index and a multi-step technique that uses the proposed bounds and performs two levels of filtering to efficiently process similarity queries. A thorough experimental study suggests that our method consistently outperforms state-of-the-art methods for DTW similarity search.

Keywords: time series analysis, similarity search, nearest neighbor search

1. Introduction

Time series is an ordered sequence of real-valued elements. Time series data are found in a variety of domains, e.g., product sales, sensor transmissions, telecommunication signals, medical and financial data. There is a need for efficient similarity search in databases of time sequences (Agrawal, Faloutsos, & Swami, 1993; Faloutsos, Ranganathan, & Manolopoulos, 1994; Goldin & Kanellakis, 1995; Moon, Whang, & Han, 2002; Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003), due to its wide use by data analysts. As a typical application, consider an investor who is interested in finding stocks that have similar behavior to a certain query stock. Similarity search is also a core module of classification or clustering algorithms that apply on time series. For instance, nearest-neighbor search based classifiers assign class labels to a new sample according to its nearest neighbor in the samples of known labels. In addition, partitioning clustering algorithms, like k-medoids (Kaufman & Rousseeuw, 1990), assign a sample to the cluster corresponding to the nearest medoid.

Similarity queries are classified into two categories. The first is *whole sequence matching* (Agrawal, Faloutsos, & Swami, 1993; Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003); given a collection $\mathcal{S}$ of $|\mathcal{S}|$ data sequences,

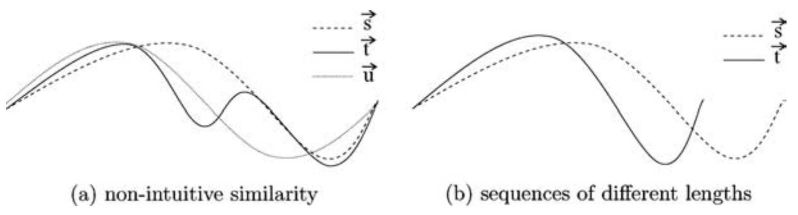


Figure 1. Ineffectiveness of Euclidean distance as a similarity measure.

a query sequence $\vec{q}$, and a threshold ϵ, the goal is to find all sequences $\vec{s} \in \mathcal{S}$, such that $D(\vec{s}, \vec{q}) \leq \epsilon$, where D is a distance (i.e., dissimilarity) function. The second category is *subsequence matching* (Faloutsos, Ranganathan, & Manolopoulos, 1994; Moon, Whang, & Han, 2002); this time we search for all (contiguous) *sub-sequences* $\vec{u} \subseteq \vec{s}$ of any $\vec{s} \in \mathcal{S}$, such that $D(\vec{u}, \vec{q}) \leq \epsilon$. In this paper, we focus on the whole sequence matching problem.[1]

A popular distance measure (Agrawal, Faloutsos, & Swami, 1993; Faloutsos, Ranganathan, & Manolopoulos, 1994; Moon, Whang, & Han, 2002) is the Euclidean distance. Although it can be computed relatively fast and has nice properties (e.g., it satisfies the triangular inequality), it is not an intuitively effective distance measure when there are fluctuations or phase shifts in time. Two time series may be considered dissimilar, although they have similar shape, if their fluctuations do not occur at the same time moments. Consider, for instance, the three series of figure 1(a). Intuitively, $\vec{s}$ is more similar to $\vec{u}$ than it is to $\vec{t}$, since $\vec{s}$ and $\vec{u}$ fluctuate in the same way. However, $\vec{s}$ is closer to $\vec{t}$ than it is to $\vec{u}$, considering Euclidean distance as a measure. Besides, Euclidean distance can only be used to efficiently measure the similarity between two sequences of the same length. For instance, we cannot directly use Euclidean distance to measure similarity between $\vec{s}$ and $\vec{t}$ in figure 1(b), since these two (similar) series have different lengths. We have to reinterpolate the two time series to be of the same length and then use Euclidean distance to measure similarity.

Dynamic time warping (DTW) has been used as a technique to calculate more robust distance for time series data, since it allows elastic shifting of sequence in order to detect similar shapes with different phases. Furthermore, it can be used to measure similarity between sequences of different lengths. Based on these advantages, DTW has been widely used in different kinds of applications such as signature verification (Munich & Perona, 1999) and voice recognition (Rabiner & Juang, 1993). Recent studies (Berndt & Clifford, 1994; Chu, Keogh, and Pazzani, 2002; Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003) have adopted DTW for generic analysis and mining tasks on time series.

However, DTW has its own limitations; it is quite expensive to compute and it does not obey the triangular inequality (Yi, Jagadish, & Faloutsos, 1998), showing resistance to indexing. In order to overcome these limitations, several studies (Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003) have defined computationally cheap lower bounds $Lb(\vec{q}, \vec{s})$ for DTW distance and indices that are used to prune fast sequences that may not be included in the query results. Thus, similarity search is performed in two steps. First, all sequences $\vec{s} \in \mathcal{S}$, such that $Lb(\vec{q}, \vec{s}) \leq \epsilon$ are discovered (filter step). Then, for each $\vec{s} \in \mathcal{S}$ that passes the filter step, the expensive $D_{dtw}(\vec{q}, \vec{s})$ is computed in order to accurately verify the distance predicate (refinement step).

In this paper, we adopt this multi-step processing technique for similarity queries using DTW. We study the application of an approximation scheme that can provide improved bounds for DTW distance. Each data sequence $\vec{s} \in \mathcal{S}$ is decomposed into a small number of segments, using a dimensionality reduction technique (Keogh et al., 2001). We then apply a version of DTW on the segmented approximations of the data and query sequences to compute fast tight lower bounds for DTW distance. Our technique resembles the segmented dynamic time warping approach proposed in Keogh, and Pazzani (1999) and Chu, Keogh, and Pazzani (2002), in that we apply DTW on segmented sequences. However, we use the segmented sequences to derive lower bounds for *exact* DTW distance, guaranteeing no false dismissals. We propose several, progressively tighter bounds depending on the existence or not of warping constraints, to be introduced in Section 2.1. Finally, we develop an index and a multi-step technique that uses the proposed bounds and performs two levels of filtering to efficiently process similarity queries. A thorough experimental study suggests that our method consistently outperforms state-of-the-art methods (Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003) for DTW similarity search.

The rest of paper is organized as follows. Section 2 provides background and related work. Section 3 describes our approach to efficiently solve the similarity search under time warping problem. Section 4 evaluates the proposed techniques with extensive experimentation. Finally, Section 5 concludes the paper.

2. Background and related work

A time series (or time sequence) $\vec{s}$ is an ordered list of elements $\langle (s_1, t_1), \ldots, (s_n, t_n) \rangle$, each consisting of a value s_i and a timestamp t_i. The values in a time sequence are typically real numbers. The timestamps are non-decreasing with respect to the sequence indices, which means that $t_i \leq t_j \Leftrightarrow i \leq j$. In typical applications, timestamps are strictly increasing. Moreover, time sequences (e.g., stock ticks) are usually obtained by sampling values at a certain rate, i.e., $\forall i, t_{i+1} - t_i = \tau$, where τ is a positive constant. For such cases, a time sequence $\vec{s}$ could be expressed by $\vec{s} = \langle s_1, \ldots, s_n \rangle$, without any information loss (i.e., when τ is known). In this paper, in accordance to past research (Agrawal, Faloutsos, & Swami, 1993; Moon, Whang, & Han, 2002; Faloutsos, Ranganathan, & Manolopoulos, 1994; Keogh, 2002a; Kim, Park, and Chu, 2001; Yi, Jagadish, & Faloutsos, 1998; Zhu & Shasha, 2003), we consider such *constant-sampled* time sequences only.

Let $\mathcal{S}$ be a collection of data sequences. Let $\vec{q}$ be a *query* sequence. Let $D(\vec{q}, \vec{s})$ be a distance function that quantizes the dissimilarity between two sequences $\vec{q}$ and $\vec{s}$. We can identify two interesting similarity query types:

- **range similarity search**: given a *distance threshold* ϵ, find all $\vec{s} \in \mathcal{S}$ such that $D(\vec{q}, \vec{s}) \leq \epsilon$
- ***k*-nearest neighbor (*k*-NN) search**: given a $k < |\mathcal{S}|$, find $\mathcal{S}' \subset \mathcal{S}$, such that $|\mathcal{S}'| = k$ and $\forall \vec{u} \in \mathcal{S}', \vec{s} \in \mathcal{S} - \mathcal{S}'$, $D(\vec{q}, \vec{u}) \leq D(\vec{q}, \vec{s})$

Table 1. Distance matrix of $\vec{q}$ and $\vec{s}$.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
q_1	3.76	8.07	1.64	1.08	2.86	0.00	0.06	1.88	1.25
q_2	2.02	5.38	0.58	2.43	4.88	0.31	0.59	3.57	2.69
q_3	6.35	11.70	3.46	0.21	1.23	0.29	0.11	0.62	0.29
q_4	16.89	25.10	11.90	1.28	0.23	4.54	3.69	0.64	1.10
q_5	3.20	7.24	1.28	1.42	3.39	0.04	0.16	2.31	1.61
q_6	3.39	7.51	1.39	1.30	3.20	0.02	0.12	2.16	1.49
q_7	4.75	9.49	2.31	0.64	2.10	0.04	0.00	1.28	0.77
q_8	0.96	3.53	0.10	4.00	7.02	1.00	1.46	5.43	4.33
q_9	0.02	1.08	0.27	8.07	12.18	3.39	4.20	10.05	8.53

Both query types are useful to data analysts who wish to find in $\mathcal{S}$ similar sequences to an interesting case (i.e., query $\vec{q}$). However, the effectiveness of search heavily depends on the distance function D used. In the next paragraphs, we describe *dynamic time warping* (DTW) and provide related work on time series similarity using DTW.

2.1. Dynamic time warping

Given two time sequences $\vec{q}$ of length n and $\vec{s}$ of length m, *dynamic time warping* (DTW) aligns each element q_i of $\vec{q}$ to one or more elements s_j of $\vec{s}$ and vice versa. DTW is performed by applying dynamic programming on an $n \times m$ *distance matrix* (Kruskall & Liberman, 1983; Rabiner & Juang, 1993). Each cell (i, j) of the distance matrix *DM* contains the *local* distance $d(q_i, s_j) = (q_i - s_j)^2$ between elements q_i of $\vec{q}$ and s_j of $\vec{s}$. Table 1 shows the distance matrix for sequences $\vec{q} = \langle -0.06, 0.46, -0.64, -2.23, 0.09, 0.04, -0.30, 0.90, 1.74\rangle$ and $\vec{s} = \langle 1.88, 2.78, 1.22, -1.10, -1.75, -0.10, -0.31, -1.43, -1.18\rangle$, plotted in figure 2.

From the distance matrix, a $n \times m$ *warping* matrix WM is constructed. Let $\overrightarrow{s_{i:j}}$ denote the subsequence $\langle s_i, s_{i+1}, \ldots, s_j\rangle$ of a sequence $\vec{s}$ $(i \leq j)$. The value $D(i, j)$ of each cell

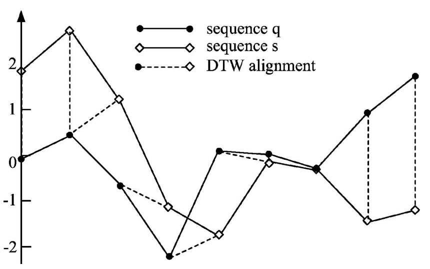


Figure 2. Two sequences $\vec{q}$ and $\vec{s}$ and their DTW alignment.

Table 2. (Constrained) warping matrix of $\vec{q}$ and $\vec{s}$.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
q_1	**3.76**	11.83	13.47	14.55	17.41	17.41	17.47	19.35	20.60
q_2	5.78	**9.14**	**9.72**	12.15	17.03	17.34	17.93	21.04	22.04
q_3	12.13	17.48	12.60	**9.93**	11.16	11.45	11.56	12.18	12.47
q_4	29.02	37.23	24.50	11.21	**10.16**	14.70	15.14	12.20	13.28
q_5	32.22	36.26	25.78	12.63	13.55	**10.20**	10.36	12.67	13.81
q_6	35.61	39.73	27.17	13.93	15.83	**10.22**	10.32	12.48	13.97
q_7	40.36	45.10	29.48	14.57	16.03	10.26	**10.22**	11.50	12.27
q_8	41.32	43.89	29.58	18.57	21.59	11.26	11.68	**15.65**	15.83
q_9	41.34	42.40	29.85	26.64	30.75	14.65	15.46	21.73	**24.18**

(i, j) in *WM* corresponds to the *minimum warping distance* that aligns $\overrightarrow{q_{1:i}}$ with $\overrightarrow{s_{1:j}}$ via a *warping path*. Formally, $D(i, j)$ for each cell (i, j) is defined by:

$$D(i, j) = \min\{D(i-1, j-1), \quad D(i-1, j), D(i, j-1)\} + d(i, j), \tag{1}$$

where $d(i, j)$ is the corresponding value in the distance matrix. The initial conditions $D(0, 0) = 0$, and $D(0, j) = D(i, 0) = \infty$, provide a basis for a dynamic-programming algorithm that computes the warping matrix progressively. The square root of the final warping distance $D(n, m)$ defines $D_{dtw}(\vec{q}, \vec{s})$. For example, Table 2 shows the warping matrix of sequences $\vec{q}$ and $\vec{s}$, from which we can derive $D_{dtw}(\vec{q}, \vec{s}) = \sqrt{D(9, 9)} = 4.9173$. The global optimal alignment between the two sequences is defined by the warping path $W = (w_1, w_2, \ldots, w_K)$ $(\max(m, n) \leq K < m + n - 1)$ in WM, which minimizes the global distance between the two sequences; each w_l, $1 \leq l \leq K$ corresponds to an (i, j) cell of WM and the value of w_l is equal to the corresponding $d(i, j)$ in DM.[2] Thus, $D_{dtw}(\vec{q}, \vec{s}) = \sqrt{\sum_{l=1}^{K} w_l}$. A warping path obeys three constraints. It should be *monotonic*; both i and j either stay the same or increase in *WM*. It should be *continuous*; i and/or j advance by 1 at a time in *WM*. Finally (*boundary condition*) the path starts at the top left and ends at the bottom right of *WM*. In Table 2, the elements in bold form the *optimal warping path* W of the two sequences $\vec{q}$ and $\vec{s}$. Figure 2 connects $\vec{q}$ and $\vec{s}$ according to this path.

Calculating the whole warping matrix and the DTW distance comes at a high $O(n \cdot m)$ computational cost. In order to reduce this cost, a *warping path constraint* could be used to limit how far the warping path may stray from the diagonal. Two popular warping path constraints are the Sakoe-Chiba band and the Itakura Parallelogram (Rabiner & Juang, 1993; Sakoe & Chiba, 1978). In this work, we focus on Sakoe-Chiba band also used by Keogh (2002a) and Zhu & Shasha (2003). In Table 2, all but the faded-out cells correspond to a Sakoe-Chiba band which allows element q_i to map only with elements $s_{i-2}, \ldots, s_{i+2}$ (and vice versa for each s_i). Formally, a *warping width coefficient* w ($w = 0.25$ in this example) allows the i-th element of one sequence to match with elements from the other at any position in $[i - x, i + x]$, where $x = \lfloor w \cdot \max\{|\vec{q}|, |\vec{s}|\} \rfloor$. Such constraints speed up the DTW distance calculation and prevent pathological warping (Keogh, 2002a; Ratanamahatana & Keogh, 2004), where a small section of one sequence maps onto a relatively large section of another.

2.2. *Lower bounds and indices for DTW*

In this section, we briefly summarize previous work related to similarity search in time series databases using DTW. Due to the expensive computation of DTW, most approaches employ the multi-step query processing strategy (Seidl & Kriegel, 1998). A computationally cheap *lower bounding* function $Lb(\vec{q}, \vec{s})$ for DTW distance is defined, such that $Lb(\vec{q}, \vec{s}) \leq D_{dtw}(\vec{q}, \vec{s})$. Let ϵ be the distance threshold for range similarity search or distance of the k-th nearest neighbor found so far for k-NN search. If for a sequence $\vec{s}$, we know that $Lb(\vec{q}, \vec{s}) > \epsilon$, we can immediately prune the sequence from search, since it cannot be in the response set. In order for a lower bounding function $Lb(\vec{q}, \vec{s})$ to be effective, (i) it should be fast to compute and (ii) it should be as tight as possible.

Yi and Faloutsos (1998) proposed a lower bounding function (denoted by Lb_Yi), based on the following observation. Any element from each sequence that is above the minimum of the two maxima of the sequences (denoted by min_max) should contribute to the DTW distance. For example, in figure 3(a), note that s_1 and s_2 are greater than the maximum of $\vec{q}$. According to the continuity property of the warping path, these elements should be mapped to at least one element of $\vec{q}$. Thus, s_1 and s_2 should contribute at least $d(s_1, \max(\vec{q})) + d(s_2, \max(\vec{q}))$ to the (squared) DTW distance. A symmetric observation holds for the elements smaller than the maximum of the two minima (denoted by max_min).[3] In figure 3(a), max_min $=$ $\max\{\min(s_i), \min(q_i)\} = s_5 = -1.75$ and min_max $= \min\{\max(s_i), \max(q_i)\} = q_9 = 1.74$, where $1 \leq i \leq 9$ and $1 \leq j \leq 9$. Thus, $Lb\_Yi(\vec{q}, \vec{s})^2 = d(s_1, \text{min\_max}) + d(s_2, min\_\text{max}) + d(q_4, \text{max\_min}) = 1.3316 \Rightarrow Lb\_Yi(\vec{q}, \vec{s}) = 1.1539$.

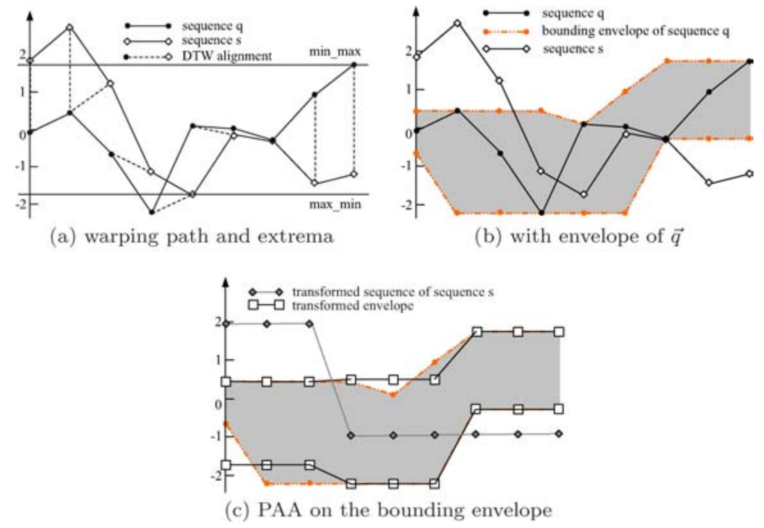


Figure 3. Two sequences $\vec{q}$ and $\vec{s}$ and lower bounding techniques.

Kim, Park, and Chu (2001) proposed a lower bounding function (denoted by Lb_Kim), which is based on a 4-tuple feature vector $\langle F(\vec{s}), L(\vec{s}), G(\vec{s}), S(\vec{s})\rangle$ extracted from each sequence $\vec{s}$. $F(\vec{s})$ and $L(\vec{s})$ are the first and the last elements, whereas $G(\vec{s})$ and $S(\vec{s})$ are the greatest and smallest elements of $\vec{s}$, respectively. For our running example, $\mathit{Feature}(\vec{s}) = \langle 1.88, -1.18, 2.78, -1.75\rangle$ and $\mathit{Feature}(\vec{q}) = \langle -0.06, 1.74, 1.74, -2.23\rangle$. The lower bounding function $\mathit{Lb\_Kim}(\vec{q}, \vec{s})$ is defined by the maximum absolute difference of the corresponding features, or formally by $L_\infty(\mathit{Feature}(\vec{q}), \mathit{Feature}(\vec{s}))$. In our example, $\mathit{Lb\_Kim}(\vec{q}, \vec{s}) = \max\{1.94, 2.92, 1.04, 0.48\} = 2.92$. Note that Lb_Kim is usually looser than Lb_Yi, since it uses just one difference between the vector scalars.

On the other hand, Lb_Kim facilitates the development of an indexing scheme that can be used to efficiently prune sequences $\vec{s}$, for which $\mathit{Lb\_Kim}(\vec{q}, \vec{s}) > \epsilon$ (Kim, Park, and Chu, 2001). Based on $\mathit{Feature}(\vec{s})$, each sequence $\vec{s}$ is mapped to a point in 4-dimensional space and inserted into a multi-dimensional index (e.g. R*-tree (Beckmann et al., 1990)). For a query sequence $\vec{q}$, $\mathit{Feature}(\vec{q})$ is first extracted and then a range query is performed to obtain candidate sequences for which $\mathit{Lb\_Kim}(\vec{q}, \vec{s}) \leq \epsilon$. The range query is defined by extending each dimension of $\mathit{Feature}(\vec{q})$ by ϵ in both directions (i.e., the L_∞ range around $\mathit{Feature}(\vec{q})$). For example, the range for $F(\vec{q})$ is $[F(\vec{q}) - \epsilon, F(\vec{q}) + \epsilon]$. Finally, $D_{dtw}(\vec{q}, \vec{s})$ is computed for each candidate.

Keogh (2002a) proposed two lower bounding functions for the case where the warping path is constrained (e.g., by a Sakoe-Chiba band). The first lower bound (denoted by Lb_Keogh) is the sum of the squared distances from every part of the data sequence $\vec{s}$, which does not fall in the *bounding envelope* of the query sequence $\vec{q}$, to the nearest orthogonal edge of the bounding envelope. For example, in figure 3b, the shaded area is the bounding envelope of sequence $\vec{q}$, corresponding to the Sakoe-Chiba band shown in Table 2. The bounding envelope captures the temporal range where each element from $\vec{q}$ can be shifted in order to be aligned to an element of $\vec{s}$. Let $\vec{U}$ and $\vec{L}$ denote the upper and lower boundaries of the envelope, respectively. Each U_i is defined by $\max\{q_{i-2}{:}q_{i+2}\}$ and each L_i by $\min\{q_{i-2}{:}q_{i+2}\}$. In our running example, $\mathit{Lb\_Keogh}(\vec{q}, \vec{s})^2 = d(s_1, U_1) + d(s_2, U_2) + d(s_3, U_3) + d(s_7, L_7) + d(s_8, L_8) + d(s_9, L_9) = 10.0278$. Thus, $\mathit{Lb\_Keogh}(\vec{q}, \vec{s}) = 3.1667$.

Due to lack of an efficient indexing scheme that employs Lb_Keogh, Keogh (2002a) also proposed another bound and an indexing scheme, which were later optimized by Zhu & Shasha (2003). First, the data sequence $\vec{s}$ is transformed into $\vec{s}'$ using the Piecewise Aggregate Approximation (PAA) dimensionality reduction technique. PAA approximates a time series by dividing it into M equi-length segments and using the average values of all the elements in each segment as its data reduced representation. For example, for $M = 3$, PAA transforms $\vec{s}$ of figure 3a to $\vec{s}' = \langle 1.96, -0.98, -0.97\rangle$. Then, the upper envelope $\vec{U}$ and lower envelope $\vec{L}$ of the query sequence $\vec{q}$ are also transformed into $\vec{U}'$ and $\vec{L}'$ using PAA. For example, figure 3(c) shows the PAA transformation for the envelope of sequence $\vec{q}$, where $\vec{U}' = \langle 0.46, 0.48\ 1.74\rangle$ and $\vec{L}' = \langle -1.7, -2.23, -0.3\rangle$. The sum of squared distances from every part of $\vec{s}'$, which does not fall in the transformed envelope, to the nearest orthogonal edge of $\vec{U}'$ and $\vec{L}'$ defines Lb_PAA. For example, $\mathit{Lb\_PAA}(\vec{q}, \vec{s})^2 = \frac{n}{M} \cdot (d(s_1', U_1') + d(s_3', L_3')) = 8.0967$. Thus, $\mathit{Lb\_PAA}(\vec{q}, \vec{s}) = 2.8455$.

Keogh (2002a) and Zhu & Shasha (2003) suggested an indexing scheme for the PAA segmentations of all $\vec{s}$ in $\mathcal{S}$ that supports similarity search in the presence of warping constraints. A multi-dimensional index, i.e., R*-tree, organizes the transformed sequences. For a given query $\vec{q}$, the warping envelope boundaries $\vec{U}$ and $\vec{L}$ are converted to $\vec{U}'$ and $\vec{L}'$, as already explained, and the index is used to retrieve sequences whose approximations are no further than ϵ from the envelope approximation. A limitation of this scheme (and also of Lb_Keogh and Lb_PAA bounds) is that it works efficiently only when all the data and query sequences are of the same length. In addition, it cannot be applied when there is no warping path constraint. Although a recent study (Ratanamahatana & Keogh, 2004) has shown that tight warping constraints usually provide better classification results compared to unconstrained warping, there may still be cases where unconstrained warping is useful. In the following section, we propose a methodology that can be applied for sequences and queries of varying lengths and in the presence or not of warping constraints.

3. Proposed methodology

In this section, we propose a methodology for efficient warping of time series. First, we describe a technique that approximates each sequence by a short sequence of M segments. Then, we provide a lower bound that can be efficiently computed from the segmented sequences by applying a small scale version of the DTW dynamic programming algorithm described in Section 2.1. Next, we show how to employ the min_max and max_min extrema of Lb_Yi to tighten the lower bound. We also describe how the bound can be further tightened in the presence of warping constraints. Finally, we extend Lb_Kim to a tighter global sequence bound and describe an indexing scheme and a multi-step process that efficiently processes similarity queries, using the proposed bounds.

3.1. Sequence segmentation

Our technique is based on the segmentation of each sequence into an Adaptive Piecewise Constant Approximation (APCA) (Keogh et al., 2001; Vlachos et al., 2003). APCA approximates a time sequence by a set of constant value segments of varying lengths such that their individual reconstruction errors are minimal. APCA is a better approximation than PAA, since it can approximate sequence parts of low variance with few segments and parts of high variance with many segments.

Obtaining an optimal APCA for a sequence of length n costs O(n^2), using dynamic programming. In this work, we use an approximate technique, based on Keogh et al. (2001), which computes the APCA in O($n \log n$) time. However, the original algorithm merges the pair of segments that lead to the least rise of reconstruction error, whereas we merge the pair of segments that results in the least increase in the area of the *minimum bounding rectangle* in the time/value dimensions, which contains all elements that belong to the merged segment. Details are omitted for the sake of readability.

We represent the APCA of a sequence $\vec{s} = \langle s_1, s_2, \ldots, s_m \rangle$ by a sequence $\overrightarrow{s^{\text{seg}}} = \langle s_1^{\text{seg}}, s_2^{\text{seg}}, \ldots, s_M^{\text{seg}} \rangle$. Each *segment* s_i^{seg} is a triplet $\langle s_i^{\text{seg}}.low, s_i^{\text{seg}}.up, s_i^{\text{seg}}.cnt \rangle$. $s_i^{\text{seg}}.low$ and $s_i^{\text{seg}}.up$ is the minimum and maximum value among all the elements of $\vec{s}$ contained in s_i^{seg}, respectively; $s_i^{\text{seg}}.cnt$ is the number of elements contained in the segment. e.g., the segment shown in figure 4(a) is represented by ⟨4, 8, 6⟩.

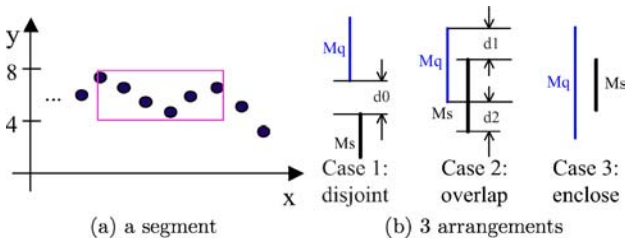


Figure 4. Segments and arrangements thereof.

APCA was used in Keogh et al. (2001) to index time series for Euclidean distance based similarity retrieval. In the next paragraphs, we show how we can use APCA segmentations to derive lower bounds for DTW.

3.2. *Segmented DTW*

Given two sequences $\vec{q}$ of length n and $\vec{s}$ of length m and a *compression ratio* c ($1 \leq c \leq \min\{m, n\}$), we can obtain two APCA segmented sequences $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$, where $N = |\overrightarrow{q^{\text{seg}}}| = \lfloor n/c \rfloor$ and $M = |\overrightarrow{s^{\text{seg}}}| = \lfloor m/c \rfloor$. By examining these segmental approximations of the two sequences, we can derive a *segmental* lower bound $Lb\_seg(\vec{q}, \vec{s})$ for $D_{dtw}(\vec{q}, \vec{s})$. For this, we employ a *Segmented Dynamic Time Warping* (SDTW) algorithm, which is a modification of the DTW algorithm, described in Section 2.1.[4] The intuition behind this approach is to use the compressed information contained in each segment to derive fast a lower bound Lb_seg of the true DTW distance.

To compute $Lb\_seg(\vec{q}, \vec{s})$, we first construct an $N \times M$ matrix, whose (i, j) element contains a *segmental distance* $d^{\text{seg}}(i, j)$ between segments q_i^{seg} and s_j^{seg}, defined as follows:

$$d^{\text{seg}}(i, j) = \begin{cases} d_1^{\text{seg}}(q_i^{\text{seg}}, s_j^{\text{seg}}) & \text{if } i = 1 \text{ and } j = 1 \\ & \text{or } i = N \text{ and } j = M \\ d_2^{\text{seg}}(q_i^{\text{seg}}, s_j^{\text{seg}}) & \text{otherwise} \end{cases} \tag{2}$$

Before we discuss how to derive d_1^{seg} and d_2^{seg}, let us examine the possible arrangements for a pair of segments q_i^{seg} and s_j^{seg}. Let $M_q = \langle q_i^{\text{seg}}.low, q_i^{\text{seg}}.up\rangle$ and $M_s = \langle s_j^{\text{seg}}.low, s_j^{\text{seg}}.up\rangle$ denote the ranges where these two segments can fluctuate. Figure 4(b) shows the three possible arrangements for these two ranges, assuming that $q_i^{\text{seg}}.up \geq s_j^{\text{seg}}.up$ (the other case is symmetric); (1) M_q and M_s are disjoint, (2) M_q and M_s overlap, and (3) M_q encloses M_s. We can now define d_1^{seg} and d_2^{seg} as follows:

$$d_1^{\text{seg}}(q_i^{\text{seg}}, s_j^{\text{seg}}) = \begin{cases} \min\{q_i^{\text{seg}}.cnt, s_j^{\text{seg}}.cnt\} \cdot d0 & \text{case 1} \\ \min\{d1, d2\} & \text{case 2} \\ 0 & \text{case 3} \end{cases} \tag{3}$$

$$d_2^{\text{seg}}(q_i^{\text{seg}}, s_j^{\text{seg}}) = \begin{cases} d0 & \text{case 1} \\ 0 & \text{case 2 and case 3} \end{cases} \tag{4}$$

In Eqs. (3) and (4), $d0 = d(q_i^{\text{seg}}.low, s_j^{\text{seg}}.up)$, $d1 = d(q_i^{\text{seg}}.up, s_j^{\text{seg}}.up)$, and $d2 = d(q_i^{\text{seg}}.low, s_j^{\text{seg}}.low)$. d_1^{seg} is the minimum warping distance of two subsequences contained in segments q_i^{seg} and s_j^{seg}, where all the elements in one segment must be mapped to the elements in the other segment and the mappings satisfy the monotonicity and continuity constraints defined in Section 2.1. Thus, d_1^{seg} lower bounds the DTW distance of two subsequences contained in segments q_i^{seg} and s_j^{seg}, where all the elements in one segment must be mapped to the elements in the other segment. This holds for the pairs of first and last segments of $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$. d_2^{seg} captures $\min(d(x, y))$, where $q_x \in q_i^{\text{seg}}$ $s_y \in s_j^{\text{seg}}$, i.e., it lower bounds the DTW distance of two subsequences in segments q_i^{seg} and s_j^{seg}, where at least one element in one segment is mapped to at least one element in the other segment and the mappings satisfy the monotonicity and continuity constraints. This holds for any other pair of segments of $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$.

We can apply the same DTW algorithm described in Section 2.1 to derive the optimal warping path between the two approximated (segmented) sequences by replacing $d(i, j)$ in Eq. (1) by $d^{\text{seg}}(i, j)$. For example, figure 5 shows the segmentations $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$ of sequences $\vec{q}$ and $\vec{s}$ of figure 3(a). Table 3 and table (a) in figure 6 show the segmental distance matrix and segmental warping matrix of $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$, respectively. We can now give our first DTW lower bounding function for sequences $\vec{q}$ and $\vec{s}$:

$$Lb\_seg1(\vec{q}, \vec{s}) = \sqrt{D(N, M)}, \tag{5}$$

where $D(N, M)$ is obtained by using SDTW described above.

Note that $d_1^{\text{seg}}(q_i^{\text{seg}}, s_j^{\text{seg}})$ is only used for calculating elements (1,1) and (N,M) of the segmental distance matrix. Note also that the first (and last) elements of two sequences $\vec{q}$ and

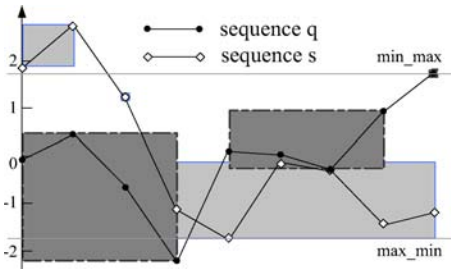


Figure 5. Segmented sequences of $\vec{q}$ and $\vec{s}$.

	$sseg_1$	$sseg_2$	$sseg_3$
$qseg_1$	4.0328	4.6104	4.6104
$qseg_2$	4.9932	4.1352	4.1352
$qseg_3$	5.0128	4.4056	7.5208

(a) using Equation 3

	$sseg_1$	$sseg_2$	$sseg_3$
$qseg_1$	5.78	6.3576	6.3576
$qseg_2$	6.7404	5.8824	5.8824
$qseg_3$	6.76	6.1528	14.4088

(b) using Equation 6

Figure 6. Segmental warping matrix ($N = M = 3$).

Table 3. Segmental distance matrix ($N = M = 3$).

	$sseg_1$ $\langle 1.88, 2.78, 2\rangle$	$sseg_2$ $\langle 1.22, 1.22, 1\rangle$	$sseg_3$ $\langle -1.75, -0.1, 6\rangle$
$qseg_1$ $\langle -2.23, 0.46, 4\rangle$	$2{*}d(0.46,1.88)$ $=4.0328$	$d(1.22,0.46)$ $=0.5776$	0
$qseg_2$ $\langle -0.3, 0.9, 4\rangle$	$d(1.88,0.9)$ $=0.9604$	$d(1.22,0.9)$ $=0.1024$	0
$qseg_3$ $\langle 1.74,1.74,1\rangle$	$d(1.88,1.74)$ $=0.0196$	$d(1.74,1.22)$ $=0.2704$	$d(1.74,-0.1)$ $=3.3856$

$\vec{s}$ must be mapped together for obtaining their DTW distance, i.e., cells (1, 1) and (n, m) are always in the warping path. Following this observation, we can tighten the distance bound of Eq. (3) for $(q_1^{\text{seg}}, s_1^{\text{seg}})$ as follows:

$$d_1^{\text{seg}}\big(q_1^{\text{seg}}, s_1^{\text{seg}}\big) = \begin{cases} (\min\left\{q_i^{\text{seg}}.cnt, s_j^{\text{seg}}.cnt\right\} - 1) \cdot d0 + d(q_1, s_1) & \text{case 1} \\ \max\{\min\{d1, d2\}, d(q_1, s_1)\} & \text{case 2} \\ d(q_1, s_1) & \text{case 3} \end{cases} \tag{6}$$

$d_1^{\text{seg}}(q_N^{\text{seg}}, s_M^{\text{seg}})$ can be derived by replacing $d(q_1, s_1)$ by $d(q_n, s_m)$ in Eq. (6). In this improvement, we explicitly consider the definite mappings $s_1 \leftrightarrow q_1$ and $s_n \leftrightarrow q_m$ in the DTW distance. By using Eq. (6) instead of Eq. (3), $D(N, M)$ becomes $(7.5208+(d(1.88, -0.06)-d(1.88, 0.46)) + (d(1.74, -1.18) - d(1.74, -0.1))) = 14.4088$ and $Lb\_seg1(\vec{q}, \vec{s}) = \sqrt{D(N, M)} = 3.7959$. The corresponding segmental warping matrix is shown in Table (b) of figure 6.

The cost of computing $D(N, M)$ is $O(N \cdot M)$ and much lower compared to the $O(n \cdot m)$ cost of $D_{dtw}(\vec{q}, \vec{s})$, since $N \ll n$ and $M \ll m$. The cost reduction factor is $O(c^2)$, where c is the compression ratio of APCA. Larger values of c result in a cheaper lower bound, and smaller values of c in a tighter one. Lemma 1 proves the correctness of Lb_seg1.

Lemma 1. *For any two time sequences $\vec{q}$ of length n and $\vec{s}$ of length m, approximated by $\overrightarrow{q^{\text{seg}}}$ of length N and $\overrightarrow{s^{\text{seg}}}$ of length M, respectively, the following inequality holds: $Lb\_seg1(\vec{q},\vec{s}) \leq D_{dtw}(\vec{q},\vec{s})$.*

Proof: Let WM be the warping matrix for $\vec{q}$ and $\vec{s}$. Let $W = w_1, w_2, \ldots, w_K$, $\max\{m, n\} \leq K < m+n-1$ be the optimal warping path in WM. Let WMS be the segmental warping matrix for $\overrightarrow{q^{\text{seg}}}$ and $\overrightarrow{s^{\text{seg}}}$. Let $P = p_1, p_2, \ldots, p_{K'}$, $\max\{M, N\} \leq K' < M + N - 1$ be the corresponding warping path of W in WMS. Due to the continuity constraint, all $w_i \in W$ in a specific $p_j \in P$ form a contiguous sub-path of W. Therefore, we can denote each p_i by a sub-path $w_{l_i} \ldots w_{u_i}$, where w_{l_i} is the first cell in W contained in p_i and w_{u_i} is the last one. Finally let $R = r_1, r_2, \ldots, r_{K''}$, $\max\{M, N\} \leq K'' < M + N - 1$ be the optimal warping path in WMS using the SDTW algorithm.

We wish to prove that $\sqrt{\sum_{i=1}^{K''} r_i} \leq \sqrt{\sum_{i=1}^{K} w_i}$. First of all, we can easily prove that $p_i \leq \sum_{j=l_i}^{u_i} w_j$, for any p_i $(1 \leq i \leq K')$, based on the correctness of d_1^{seg} and d_2^{seg} (Eqs. (3), (4), and (6)). Thus:

$$\sum_{i=1}^{K'} p_i \leq \sum_{i=1}^{K'} \sum_{j=l_i}^{u_i} w_j = \sum_{i=1}^{K} w_i \tag{7}$$

We also know that $\sum_{i=1}^{K''} r_i \leq \sum_{i=1}^{K'} p_i$, because R is the optimal path in the segmental warping matrix. By transitivity, due to Eq. (7), we have $\sum_{i=1}^{K''} r_i \leq \sum_{i=1}^{K} w_i$, or else $\sqrt{\sum_{i=1}^{K''} r_i} \leq \sqrt{\sum_{i=1}^{K} w_i}$. □

3.2.1. An improved bound using extrema. The lower bound Lb_seg1 is not very tight due to the approximate nature of the segments. Specifically, d_2^{seg} considers only the minimum distances between the two segments which is too small to be useful in practice. In order to derive a tighter bound, without trading much computational time, we make use of the property used by Lb_Yi; the sum of squared differences from all elements larger than min_max (or smaller than max_min), must contribute to the squared DTW distance. The difference from Lb_Yi is that we use the segmented information instead of the actual sequences. From two segmented (query and data) sequences q^{seg} and s^{seg}, we can compute the min_max and max_min values and characterize their segments by one of the following three types:

- SegTypeA: the whole segment is outside range [max_min, min_max]
- SegTypeB: part of the segment is outside range [max_min, min_max]
- SegTypeC: the whole segment is inside range [max_min, min_max]

Figure 7 shows two segmented sequences, their max_min and min_max values and the types of their segments. Assume a data segment s_i^{seg} is of type SegTypeA, where $s_i^{\text{seg}}.low >$ min_max (it is located above the min_max line). In this case, $s_i^{\text{seg}}.cnt - 1$ of its elements have at least distance $d(\text{min\_max}, s_i^{\text{seg}}.low)$ and exactly one of its elements has at least distance $d(\text{min\_max}, s_i^{\text{seg}}.up)$ from any segment of $\vec{q}$. If the segment is of type SegTypeB, then at least one of its elements has distance at least $d(\text{min\_max}, s_i^{\text{seg}}.up)$ from any segment

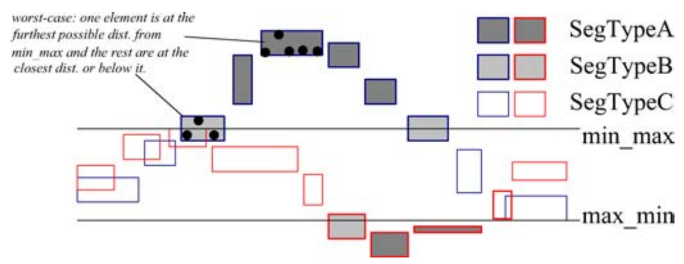

worst-case: one element is at the furthest possible dist. from min_max and the rest are at the closest dist. or below it.
SegTypeA
SegTypeB
SegTypeC
min_max
max_min

Figure 7. Three segment types.

```
Algorithm getSegDiff($\overrightarrow{r^{seg}}$,min_max, max_min)
1.  $SegDiffs := 0$;
2.  if $r_i^{seg}.low$ >min_max then /* SegTypeA */
3.      $segDiffs := (r_i^{seg}.cnt - 1) * d(\text{min\_max}, r_i^{seg}.low) + d(\text{min\_max}, r_i^{seg}.up)$;
4.      $r_i^{seg}.low$:=min_max; $r_i^{seg}.up$:=min_max;
5.  else if $r_i^{seg}.up$ >min_max then /* SegTypeB */
6.      $segDiffs := d(\text{min\_max}, r_i^{seg}.up)$;
7.      $r_i^{seg}.up$:=min_max;
8.  else if $r_i^{seg}.up$ <max_min then /* SegTypeA */
9.      $segDiffs := (r_i^{seg}.cnt - 1) * d(\text{max\_min}, r_i^{seg}.up) + d(\text{max\_min}, r_i^{seg}.low)$;
10.     $r_i^{seg}.up$:=max_min; $r_i^{seg}.low$:=max_min;
11. else if $r_i^{seg}.low$ <max_min then /* SegTypeB */
12.     $segDiffs := d(\text{max\_min}, r_i^{seg}.low)$;
13.     $r_i^{seg}.low$:=max_min;
14. return $SegDiffs$;
```

```
Algorithm getDiff($\overrightarrow{s^{seg}}$,$\overrightarrow{q^{seg}}$, $M$, s_min, s_max, q_min, q_max)
/* s_max(s_min) and q_max(q_min) are two maxima(minima) of sequences $\vec{s}$ and $\vec{q}$ respectively */
1.  $Diff := 0$;
2.  min_max := min(s_max,q_max);
3.  max_min := max(s_min,q_min);
4.  for each segment $s_i^{seg}$ in $\overrightarrow{s^{seg}}$ do /* process seq. $\overrightarrow{s^{seg}}$ */
5.      $Diff$ += getSegDiff($\overrightarrow{s^{seg}}$, min_max, max_min);
6.  if min($\vec{s}$) >max($\vec{q}$) then /* $\vec{s}$ is completely above $\vec{q}$ */
7.      max_min:=min_max;
8.  if min($\vec{q}$) >max($\vec{s}$) then /* $\vec{q}$ is completely above $\vec{s}$ */
9.      min_max:=max_min;
10. for each segment $q_j^{seg}$ in $\overrightarrow{q^{seg}}$ do /* process seq. $\overrightarrow{q^{seg}}$ */
11.     $Diff$ += getSegDiff($\overrightarrow{q^{seg}}$, min_max, max_min);
12. return $Diff$;
```

Figure 8. Lb_seg2 preprocessing.

of $\vec{q}$. Based on the symmetric cases for below max_min and for data segments s_j^{seg}, the **getDiff** algorithm of figure 8 accumulates these distances for segments of types SegTypeA and SegTypeB.

After accumulating the distances in the *Diff* variable, we run the SDTW algorithm to get a $D(N, M)$ (as for Lb_seg1). In order to avoid overestimation of the lower bound, the **getSegDiff** algorithm, in addition to computing the distances, *projects* the parts of type SegTypeA and SegTypeB segments that are above the min_max line (below the max_min line) onto the corresponding line (pseudocode lines 4, 7, 10, and 13). A subtle point to note is that if one sequence is completely above the other (see algorithm **getDiff** lines 6–9), the segments are projected onto the same min_max or max_min line in order to avoid erroneous calculation during SDTW. The Lb_seg2 lower bound can now be defined by the following equation:

$$Lb\_seg2(\vec{q}, \vec{s}) = \sqrt{Diff + D(N, M)} \tag{8}$$

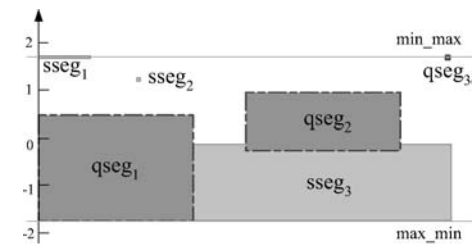


Figure 9. Segmented $\vec{q}$ and $\vec{s}$ after preprocessing.

Let us now see how Lb_seg2 can be computed for our running example. Figure 9 shows the segmented sequences of sequences $\vec{s}$ and $\vec{q}$ after applying the **getDiff** algorithm. Observe that the first segment of $\overrightarrow{s^{\text{seg}}}$ is projected on the min_max line since it is totally above it. Also the part of $\overrightarrow{q^{\text{seg}}}$ below the max_min line is truncated. The accumulated *Diff* distance is $(2-1)d(1.88, \text{min\_max}) + d(2.78, \text{min\_max}) + d(\text{max\_min}, -2.23) = 1.3316$.

Table 4 shows the converted segmented sequences after **getDiff** has been applied (first row and column) and the new distances between the segments. Table 5 shows the corresponding segmental warping matrix. One thing to note is that the adjustments also affect the first and last values of the segments $\vec{q}$ and $\vec{s}$, which are considered in the calculation of $D(N, M)$ (i.e., see Eq. (6)). For example, s_1 is adjusted to 1.74. Thus, in our example, $Lb\_seg2(\vec{q}, \vec{s}) = \sqrt{Diff + D(N, M)} = 3.8521$. Although $D(N, M)$ for $Lb\_seg2(\vec{q}, \vec{s})$ becomes smaller than

Table 4. Segmental distance matrix after conversion.

	$ssseg_1$ ⟨1.74, 1.74, 2⟩	$ssseg_2$ ⟨1.22, 1.22, 1⟩	$ssseg_3$ ⟨−1.75, −0.1, 6⟩
$qqseg_1$ ⟨−1.75, 0.46, 4⟩	(2−1)**d*(0.46,1.74) +*d*(1.74, −0.06) =4.8784	*d*(1.22,0.46) =0.5776	0
$qqseg_2$ ⟨−0.3, 0.9, 4⟩	*d*(1.74,0.9) =0.7056	*d*(1.22,0.9) =0.1024	0
$qqseg_3$ ⟨ 1.74,1.74,1⟩	0	*d*(1.74,1.22) =0.2704	*d*(1.74,−1.18) =8.5264

Table 5. Segment warping matrix after conversion.

	$ssseg_1$	$ssseg_2$	$ssseg_3$
$qqseg_1$	4.8784	5.456	5.456
$qqseg_2$	5.584	4.9808	4.9808
$qqseg_3$	5.584	5.2512	13.5072

Lb_seg$1(\vec{q}, \vec{s})$, *Diff* captures global warping distances contributed to DTW distance and finally *Lb_seg*$2(\vec{q}, \vec{s})$ turns out to be tighter than *Lb_seg*$1(\vec{q}, \vec{s})$. Besides capturing global warping distances contributed to DTW, like Lb_Yi (described in Section 2.2), SDTW also takes the three constraints of warping path (as stated in Section 2.1) into consideration.

Lemma 2 states the correctness of Lb_seg2. Before we give its case-based proof, we prove the following proposition, used in several parts of the proof of Lemma 2.

Proposition 1. *For any* $x, y, z \in \mathbb{R}$, *if* $x \leq y \leq z$, *then* $d(x, y) + d(y, z) \leq d(x, z)$, *where* $d(a, b) = (a - b)^2$.

Proof:

$$\begin{aligned} d(x, y) + d(y, z) - d(x, z) &= (x - y)^2 + (y - z)^2 - (x - z)^2 \\ &= (x^2 + y^2 - 2xy + y^2 + z^2 - 2yz) - (x^2 + z^2 - 2xz) \\ &= 2(y^2 - xy - zy + xz) \\ &= 2(y - z)(y - x) \leq 0 \Rightarrow d(x, y) + d(y, z) \leq d(x, z) \end{aligned}$$

□

Lemma 2. *For any two time sequences* $\vec{q}$ *of length* n *and* $\vec{s}$ *of length* m, *approximated by* $\overrightarrow{q^{\text{seg}}}$ *of length* N *and* $\overrightarrow{s^{\text{seg}}}$ *of length* M, *respectively, the following inequality holds*: $Lb\_seg2(\vec{q}, \vec{s}) \leq D_{dtw}(\vec{q}, \vec{s})$.

Proof: W.l.o.g., assume that $\max(\vec{q}) \geq \max(\vec{s})$. Let K be the length of the optimal warping path of $\vec{q}$ and $\vec{s}$. We can replace $\vec{s}$ and $\vec{q}$ by two K-length sequences $\vec{q'}$ and $\vec{s'}$, respectively, such that for each $1 \leq i \leq K$, q'_i and s'_i correspond to the elements of $\vec{q}$ and $\vec{s}$, respectively, that are mapped at the warping element w_i. Thus, $D_{dtw}(\vec{q}, \vec{s})^2 = \sum_{i=1}^{K} d(q'_i, s'_i)$.

Let $\overrightarrow{s''}$ be the sequence that is derived from $\vec{s'}$ such that $s''_i = \text{min\_max}$, if $s'_i > \text{min\_max}$, $s''_i = \text{max\_min}$, if $s'_i < \text{max\_min}$, and $s''_i = s'_i$, otherwise $(1 \leq i \leq K)$. $\overrightarrow{q''}$ is similarly derived from $\vec{q'}$, if $\vec{q}$ and $\vec{s}$ overlaps or $\vec{q}$ encloses $\vec{s}$. If $\vec{q}$ and $\vec{s}$ are disjoint, $q''_i = q'_i$ $(1 \leq i \leq K)$.

If we consider the relative positions of the minima and maxima of the two sequences, we can distinguish three possible arrangements thereof, which are shown in figure 4(b) (considering whole sequences instead of segments). For each of the three cases, we will first prove that $D_{dtw}(\vec{q}, \vec{s})^2 \geq Diff + \sum_{i=1}^{K} d(q''_i, s''_i)$, where *Diff* is obtained by using the algorithm of figure 8.

Case 1: $\vec{q}$ and $\vec{s}$ are disjoint (i.e., $\min(\vec{q}) > \max(\vec{s})$)

$$\begin{aligned} D_{dtw}(\vec{q}, \vec{s})^2 = \sum_{i=1}^{K} d(q'_i, s'_i) &\geq \sum_{i=1}^{K} (d(\text{max\_min}, s'_i) + d(q'_i, \text{max\_min})) \\ &\quad \text{(due to proposition 1)} \\ &= \sum_{i=1}^{K} d(\text{max\_min}, s'_i) + \sum_{i=1}^{K} d(q''_i, s''_i) \geq Diff + \sum_{i=1}^{K} d(q''_i, s''_i) \end{aligned}$$

Case 2: $\vec{q}$ and $\vec{s}$ overlap (i.e., $\min(\vec{q}) \leq \max(\vec{s}) \wedge \min(\vec{q}) \geq \min(\vec{s})$)

$$\begin{aligned}
D_{dtw}(\vec{q}, \vec{s})^2 &= \sum_{i=1}^{K} d(q_i', s_i') \\
&\geq \sum_{q_i' > \text{min\_max} \wedge s_i' < \text{max\_min}} (d(q_i', \text{min\_max}) + d(q_i'', s_i'') + d(\text{max\_min}, s_i')) \\
&\quad + \sum_{q_i' > \text{min\_max} \wedge s_i' \geq max\_min} (d(q_i', \text{min\_max}) + d(q_i'', s_i'')) \\
&\quad + \sum_{q_i' \leq \text{min\_max} \wedge s_i' \geq max\_min} (d(q_i'', s_i'')) \\
&\quad + \sum_{q_i' \leq \text{min\_max} \wedge s_i' < max\_min} (d(q_i'', s_i'') + d(\text{max\_min}, s_i'))(\text{due to proposition 1}) \\
&= \sum_{q_i' > \text{min\_max}} d(q_i', \text{min\_max}) + \sum_{s_j' < \text{max\_min}} d(s_j', \text{max\_min}) + \sum_{i=1}^{K} d(q_i'', s_i'') \\
&\geq \textit{Diff} + \sum_{i=1}^{K} d(q_i'', s_i'')
\end{aligned}$$

Case 3: if $\vec{q}$ encloses $\vec{s}$ (i.e., $\min(\vec{q}) < \min(\vec{s})$)

$$\begin{aligned}
D_{dtw}(\vec{q}, \vec{s})^2 &= \sum_{i=1}^{K} d(q_i', s_i') \\
&\geq \sum_{q_i' > \text{min\_max}} (d(q_i', \text{min\_max}) + d(q_i'', s_i'')) \\
&\quad + \sum_{q_j' < \text{max\_min}} (d(q_j', \text{max\_min}) + d(q_j'', s_j'')) \ \ (\text{due to proposition 1}) \\
&= \sum_{q_i' > \text{min\_max}} d(q_i', \text{min\_max}) + \sum_{q_j' < \text{max\_min}} d(q_j', \text{max\_min}) + \sum_{i=1}^{K} d(q_i'', s_i'') \\
&\geq \textit{Diff} + \sum_{i=1}^{K} d(q_i'', s_i'')
\end{aligned}$$

Having shown that $D_{dtw}(\vec{q}, \vec{s})^2 \geq \textit{Diff} + \sum_{i=1}^{K} d(q_i'', s_i'')$ in all cases, we will now prove that $\sum_{i=1}^{K} d(q_i'', s_i'') \geq D(N, M)$, where $D(N, M)$ is computed using the mapped sequences on the max_min (and/or min_max) axes. Let $\overrightarrow{ss}$ be a sequence derived from $\vec{s}$, such that $ss_i = \text{min\_max}$, if $s_i > \text{min\_max}$, $ss_i = \text{max\_min}$, if $s_i < \text{max\_min}$ $(1 \leq i \leq m)$, and $ss_i = s_i$, otherwise. $\overrightarrow{qq}$ is similarly derived from $\vec{q}$, if $\vec{q}$ and $\vec{s}$ overlaps or $\vec{q}$ encloses $\vec{s}$. If $\vec{q}$ and $\vec{s}$ are disjoint, $qq_i = q_i$ $(1 \leq i \leq n)$.

Let $\overrightarrow{ss^{\text{seg}}}$ and $\overrightarrow{qq^{\text{seg}}}$ be the segmental representations derived from $\vec{ss}$, $\vec{qq}$, $\overrightarrow{s^{\text{seg}}}$ and $\overrightarrow{q^{\text{seg}}}$, where $\overrightarrow{s^{\text{seg}}}$ and $\overrightarrow{q^{\text{seg}}}$ are the segmental representations of $\vec{s}$ and $\vec{q}$ respectively. To derive $\overrightarrow{ss^{\text{seg}}}$, we set $ss_i^{\text{seg}}.cnt = s_i^{\text{seg}}.cnt$, but $ss_i^{\text{seg}}.low$ and $ss_i^{\text{seg}}.up$ are obtained by using sequence elements contained in the i-th segment of $\vec{ss}$. Similarly, we get $\overrightarrow{qq^{\text{seg}}}$. $\overrightarrow{ss^{\text{seg}}}$ and $\overrightarrow{qq^{\text{seg}}}$ correspond to the segmental representations after the algorithm of figure 8 has been applied. Recall that $D(N, M)$ is obtained by applying SDTW using $\overrightarrow{qq^{\text{seg}}}$ and $\overrightarrow{ss^{\text{seg}}}$. Since $\sum_{i=1}^{K} d(q_i'', s_i'')$ is the distance corresponding to the optimal warping path, we know $\sum_{i=1}^{K} d(q_i'', s_i'') \geq D(N, M)$ from Lemma 1. Thus,

$Lb\_seg2(\vec{q}, \vec{s})^2 = Diff + D(N, M) \leq Diff + \sum_{i=1}^{K} d(q_i'', s_i'') \leq \sum_{i=1}^{K} d(q_i', s_i') = D_{dtw}(\vec{q}, \vec{s})^2$ □

Note that Lb_seg1 and Lb_seg2 are independent of the existence of warping constraints. Next, we show how we can take advantage of such constraints to derive a tighter lower bound, which can be computed efficiently.

3.2.2. A bound for warping constraints. In the presence of warping constraints, we improve our SDTW technique in two ways. First, we restrict the cells of the segmental warping matrix to only those which contain at least one cell of the constrained warping matrix. Figure 10(a) shows an example of a warping matrix constrained by a Sakoe-Chiba band, indicated by the heavy-shaded cells around the diagonal. The heavy border defines the segmental warping matrix of $\overrightarrow{s^{\text{seg}}}$ and $\overrightarrow{q^{\text{seg}}}$. The corresponding constrained segmental warping matrix is shown in figure 10b. When applying SDTW, only shaded cells are considered for the optimal segmental warping path. The second improvement is based on the following observation. A specific segment s_i^{seg} can only map with some segments from $\overrightarrow{q^{\text{seg}}}$, according to the *constrained* segmental warping matrix, described above. Thus, instead of using the global min_max, max_min extrema for s_i^{seg} in the **getDiff** algorithm of figure 8, we consider only the extrema based on the segments with which each s_i^{seg} and q_j^{seg} can map. For example, in figure 11 assume that segment s_8^{seg} can only map with segments q_6^{seg}, q_7^{seg}, q_8^{seg} and q_9^{seg} based on the segmental constrained warping matrix. up and low are the extrema for this

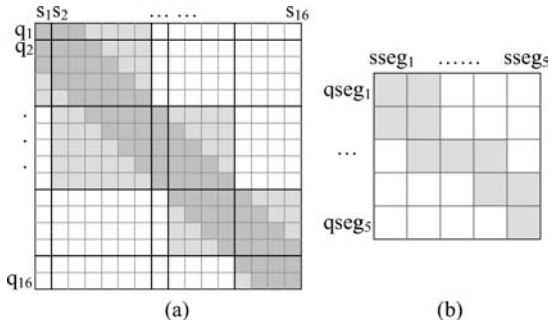


Figure 10. Constrained segmented matrix.

group of segments, as shown in the figure. Since s_8^{seg} is completely above up, the *segDiffs* quantity for it becomes $(s_8^{seg}.cnt - 1) \times d(up, s_8^{seg}.low) + d(up, s_8^{seg}.up)$, as opposed to the looser $d(\text{min\_max}, s_8^{seg}.up)$ computed by the unconstrained **getDiff** algorithm. After the computation of *segDiffs*, s_8^{seg} is converted to $\langle up, up, s_8^{seg}.cnt\rangle$ in order to avoid overestimation of SDTW, like in the algorithm of figure 8. Note here that this improvement is for $\overrightarrow{s^{seg}}$ only. This means we make no change in the loop of process sequence $\overrightarrow{q^{seg}}$ to avoid overestimation.

The improved Lb_seg3 bound is then defined by Eq. (9), similarly to Lb_seg2, but now *Diff'* is the accumulated distance in the constrained version of **getDiff** and $D'(N, M)$ is the optimal SDTW distance in the constrained segmental warping matrix.

$$Lb\_seg3(\vec{q}, \vec{s}) = \sqrt{Diff' + D'(N, M)} \tag{9}$$

For our running example of figure 3, if we use the Sakoe-Chiba band shown in figure 3(c) and the segmentation of figure 5, the converted segmented sequences and the corresponding distance and warping matrices are shown in figure 12 and Tables 6 and 7, respectively. In this case, $Diff' = (2-1) \times d(0.46, 1.88) + d(2.78, 0.46) + d(1.22, 0.9) + d(-1.75, -2.23) = 7.7316$ and $Lb\_seg3(\vec{q}, \vec{s}) = \sqrt{Diff' + D'(N, M)} = 4.0655$.

Compared with $Lb\_seg2(\vec{q}, \vec{s})$, *Diff'* of $Lb\_seg3(\vec{q}, \vec{s})$ is much larger than *Diff* for $Lb\_seg2(\vec{q}, \vec{s})$ by taking advantage of using the warping envelope constraint, although $D'(N, M)$ becomes smaller than $D(N, M)$ shown in Table 5.

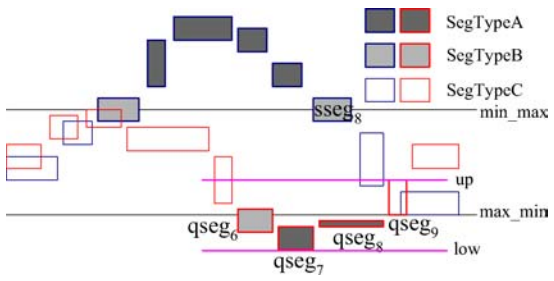


Figure 11. Segment distances in constrained SDTW.

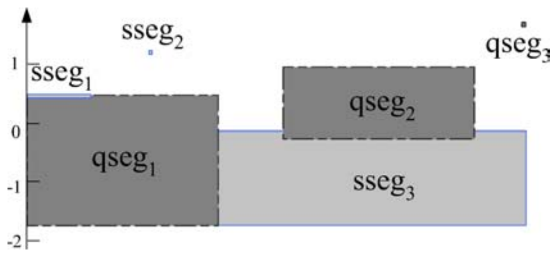


Figure 12. Segmented $\vec{q}$ and $\vec{s}$ after constrained preprocessing.

Table 6. Segmental distance matrix for Lb_seg3.

	$ssseg_1$ $\langle 0.46, 0.46, 2\rangle$	$ssseg_2$ $\langle 0.9, 0.9, 1\rangle$	$ssseg_3$ $\langle -1.75, -0.1, 6\rangle$
$qqseg_1$ $\langle -1.75, 0.46, 4\rangle$	$d(0.46, -0.06)$ $=0.2704$	$d(0.9, 0.46)$ $=0.1936$	0
$qqseg_2$ $\langle -0.3, 0.9, 4\rangle$	/	0	0
$qqseg_3$ $\langle 1.74, 1.74, 1\rangle$	/	/	$d(1.74, -1.18)$ $=8.5264$

Table 7. Segmental warping matrix for Lb_seg3.

	$ssseg_1$	$ssseg_2$	$ssseg_3$
$ssseg_1$	0.2704	0.464	0.464
$ssseg_2$	/	0.2704	0.2704
$ssseg_3$	/	/	8.7968

3.3. An improvement over Lb_Kim

In this section, we propose an improvement of the Lb_Kim bound, discussed in Section 2.2. Recall that we can approximate a sequence $\vec{s}$ by a 4-tuple $Feature(\vec{s}) = \langle F(\vec{s}), L(\vec{s}), G(\vec{s}), S(\vec{s})\rangle$, storing the first, last, greatest, and smallest elements of $\vec{s}$. Kim, Park, and Chu (2001) have shown that $Lb\_Kim(\vec{q}, \vec{s}) = L_\infty(Feature(\vec{q}), Feature(\vec{s}))$ is a lower bound for $D_{dtw}(\vec{q}, \vec{s})$.

We say that sequence $\vec{s}$ is *oscillating* iff its maximum is larger than its first and last elements and its minimum is smaller than its first and last elements. Otherwise we say that it is *not oscillating*. Formally, $\vec{s}$ is oscillating iff $G(\vec{s}) > \max\{F(\vec{s}), L(\vec{s})\} \wedge S(\vec{s}) < \min\{F(\vec{s}), L(\vec{s}\})$. For example, in figure 3(a), $\vec{s}$ is oscillating, but $\vec{q}$ is not ($L(\vec{q}) = G(\vec{q})$). We can prove the following lemma:

Lemma 3. *Let $\vec{q}$ and $\vec{s}$ be two time sequences. If both $\vec{q}$ and $\vec{s}$ are oscillating then*

$$L_2(Feature(\vec{q}), Feature(\vec{s})) \leq D_{dtw}(\vec{q}, \vec{s}), \tag{10}$$

where $L_2(Feature(\vec{q}), Feature(\vec{s}))$ denotes the Euclidean distance between the two global feature vectors.

Proof: Assume that $\max(\vec{q}) \geq \max(\vec{s})$. If this assumption does not hold, the roles of $\vec{s}$ and $\vec{q}$ are exchanged. Let K be the length of the optimal warping path of $\vec{q}$ and $\vec{s}$. We can replace $\vec{s}$ and $\vec{q}$ by two K-length sequences $\vec{q'}$ and $\vec{s'}$, respectively, such that for each $1 \leq i \leq K$, q'_i and s'_i correspond to the elements of $\vec{q}$ and $\vec{s}$, respectively, that are mapped at the warping element w_i. Thus, $D_{dtw}(\vec{q}, \vec{s})^2 = \sum_{i=1}^{K} d(q'_i, s'_i)$. We can rewrite the optimal warping distance as:

$$\begin{aligned}
D_{dtw}(\vec{q},\vec{s})^2 &= \sum_{i=1}^{K} d(q'_i, s'_i)\\
&= d(q'_1, s'_1) + \sum_{i=2}^{K-1} d(q'_i, s'_i) + d(q'_K, s'_K)\\
&= d(F(\vec{q}), F(\vec{s})) + \sum_{i=2}^{K-1} d(q'_i, s'_i) + d(L(\vec{q}), L(\vec{s}))
\end{aligned}$$

Thus it suffices to prove that $\sum_{i=2}^{K-1} d(q'_i, s'_i) \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))$. Since both $\vec{q}$ and $\vec{s}$ are oscillating, S($\vec{s}$), G($\vec{s}$), S($\vec{q}$), and G($\vec{q}$) are not the first and last elements in each sequence. Thus, they should contribute to the $\sum_{i=2}^{K-1} d(q'_i, s'_i)$ distance. Based on three possible arrangements of $\vec{s}$ and $\vec{q}$, we show that for each case the above inequality holds.

Case 1: $\vec{s}$ and $\vec{q}$ are disjoint ($S(\vec{q}) > G(\vec{s})$)

Let us first assume that $G(\vec{q}) \not\leftrightarrow S(\vec{s})$, i.e., the greatest element of $\vec{q}$ is not mapped to the smallest of $\vec{s}$ in the optimal warping path. In this case, $\sum_{i=2}^{K-1} d(q'_i, s'_i)$ will contain a component corresponding to a mapping of $G(\vec{q})$ to some element s_j in $\vec{s}$ and a component corresponding to a mapping of $S(\vec{s})$ to some element q_l in $\vec{q}$. Since $G(\vec{s}) \geq s_j$, and $\vec{q}$ is above $\vec{s}$, we know that $d(G(\vec{q}), s_j) \geq d(G(\vec{q}), G(\vec{s}))$. Similarly, $d(S(\vec{s}), q_l) \geq d(S(\vec{s}), S(\vec{q}))$. Thus:

$$\sum_{i=2}^{K-1} d(q'_i, s'_i) \geq d(G(\vec{q}), s_j) + d(G(\vec{s}), q_l) \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s})).$$

Now, let us consider the special case, where $G(\vec{q}) \leftrightarrow S(\vec{s})$. We know that apart from this mapping, $S(\vec{q})$ should also map to an element from $\vec{s}$. The minimum distance contributed by $S(\vec{q})$ is when it maps to $G(\vec{s})$. Thus, $\sum_{i=2}^{K-1} d(q'_i, s'_i) \geq d(G(\vec{q}), S(\vec{s})) + d(S(\vec{q}), G(\vec{s}))$. We can now show that

$$\begin{aligned}
&(d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))) - (d(G(\vec{q}), S(\vec{s})) + d(S(\vec{q}), G(\vec{s})))\\
&= (G(\vec{q})^2 + G(\vec{s})^2 - 2G(\vec{q})G(\vec{s}) + S(\vec{q})^2 + S(\vec{s})^2 - 2S(\vec{q})S(\vec{s})) - (G(\vec{q})^2\\
&\quad + S(\vec{s})^2 - 2G(\vec{q})S(\vec{s}) + S(\vec{q})^2 + G(\vec{s})^2 - 2S(\vec{q})G(\vec{s}))\\
&= 2(G(\vec{q})S(\vec{s}) + S(\vec{q})G(\vec{s}) - G(\vec{q})G(\vec{s}) - S(\vec{q})S(\vec{s}))\\
&= 2(G(\vec{q}) - S(\vec{q}))(S(\vec{s}) - G(\vec{s})) \leq 0\\
&\Rightarrow (d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))) \leq (d(G(\vec{q}), S(\vec{s})) + d(S(\vec{q}), G(\vec{s})))
\end{aligned}$$

Thus, $\sum_{i=2}^{K-1} d(q'_i, s'_i) \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))$ also in this special case.

Case 2: $\vec{s}$ and $\vec{q}$ overlaps ($S(\vec{q}) \leq G(\vec{s}) \wedge S(\vec{q}) \geq S(\vec{s})$)

Because all elements in each sequence must be mapped to at least one element in another sequence, $G(\vec{q})$ and $S(\vec{s})$ must be mapped. The mapping which gives the smallest mapping distance is $G(\vec{q}) \leftrightarrow G(\vec{s})$, $S(\vec{q}) \leftrightarrow S(\vec{s})$, because for $G(\vec{q})$ the closest element in sequence $\vec{s}$ is $G(\vec{s})$ and for $S(\vec{s})$ the closest element in sequence $\vec{q}$ is $S(\vec{q})$. Due to $S(\vec{q}) \leq G(\vec{s})$, the mapping distance of $G(\vec{q}) \leftrightarrow G(\vec{s})$, $S(\vec{q}) \leftrightarrow S(\vec{s})$ is not greater than the mapping distance of $G(\vec{q}) \leftrightarrow S(\vec{s})$.

Thus, $\sum_{i=2}^{K-1} d(q_i', s_i') \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))$, when $\vec{s}$ and $\vec{q}$ overlap.

Case 3: $\vec{q}$ encloses $\vec{s}$ $(S(\vec{q}) < S(\vec{s}))$

Because all elements in each sequence must be mapped to at least one element in another sequence, $G(\vec{q})$ and $S(\vec{q})$ must be mapped. The mapping which gives the smallest mapping distance is $G(\vec{q}) \leftrightarrow G(\vec{s})$, $S(\vec{q}) \leftrightarrow S(\vec{s})$, because for $G(\vec{q})$ the closest element in sequence $\vec{s}$ is $G(\vec{s})$ and for $S(\vec{q})$ the closest element in sequence $\vec{s}$ is $S(\vec{s})$.

Thus, $\sum_{i=2}^{K-1} d(q_i', s_i') \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))$, when $\vec{q}$ encloses $\vec{s}$.

In all cases, $\sum_{i=2}^{K-1} d(q_i', s_i') \geq d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s}))$, thus we get

$$\begin{aligned} D_{dtw}(\vec{q}, \vec{s})^2 &= d(F(\vec{q}), F(\vec{s})) + \sum_{i=2}^{K-1} d(q_i', s_i') + d(L(\vec{q}), L(\vec{s})) \\ &\geq d(F(\vec{q}), F(\vec{s})) + d(G(\vec{q}), G(\vec{s})) + d(S(\vec{q}), S(\vec{s})) + d(L(\vec{q}), L(\vec{s})) \\ &\Rightarrow L_2(\mathit{Feature}(\vec{q}), \mathit{Feature}(\vec{s})) \leq D_{dtw}(\vec{q}, \vec{s}). \end{aligned}$$

□

The rationale behind Lemma 3 is that every one of the four features in the vectors should contribute to the DTW distance if both sequences are oscillating. If one of sequences is not oscillating then we can still improve Lb_Kim, since we know that the first two and last two elements should always match together:

$$\mathit{Lb\_glob}_{no\_osc}(\vec{q}, \vec{s}) = \sqrt{\max\{d_F + d_L, d_G, d_S\}}, \tag{11}$$

where d_X is an abbreviation for $d(X(\vec{q}), X(\vec{s}))$ (e.g., $d_F = d(F(\vec{q}), F(\vec{s}))$. Summarizing, based on whether both sequences are oscillating (case A) or not (case B) we can define a global lower bound Lb_glob based on the feature vectors of the sequences as follows:

$$\mathit{Lb\_glob}(\vec{q}, \vec{s}) = \begin{cases} L_2(\mathit{Feature}(\vec{q}), \mathit{Feature}(\vec{s})) & \text{case A} \\ \mathit{Lb\_glob}_{no\_osc}(\vec{q}, \vec{s}) & \text{case B} \end{cases} \tag{12}$$

3.4. Indexing scheme and similarity search

Our indexing scheme organizes the global feature vectors $\mathit{Feature}(\vec{s})$ of all $\vec{s} \in \mathcal{S}$ in a 4-dimensional R^*-tree. Note that the same scheme is used in Kim, Park, and Chu (2001), however, (i) we employ the tighter bound Lb_glob to guide search and (ii) we use segmented

```
Algorithm RSimSearch(query $\vec{q}$, $\mathcal{S}$, $\epsilon$)
1.  compute $Feature(\vec{q})$ and $\overrightarrow{q^{seg}}$;
2.  perform an $L_\infty$ $\epsilon$ range query to get all $\vec{s}$
        such that $L_\infty(Feature(\vec{q}), Feature(\vec{s})) \leq \epsilon$;
3.  put $s.id$ of each result $\vec{s}$ to $\mathcal{C}$ if:
4.      (i) $Lb\_glob(\vec{q}, \vec{s}) \leq \epsilon$ and
5.      (ii) $Lb\_seg(\vec{q}, \vec{s}) \leq \epsilon$;
6.  for each $s.id \in \mathcal{C}$
7.      access exact representation of $\vec{s}$;
8.      if $D_{dtw}(\vec{q}, \vec{s}) \leq \epsilon$ then report $\vec{s}$;
```

Figure 13. Range similarity search algorithm.

representations of the sequences to apply an additional filter using Lb_seg (i.e., one of Lb_seg1, Lb_seg2, or Lb_seg3) before computing the exact $D_{dtw}(\vec{q}, \vec{s})$. Thus, each entry in a leaf node of the R*-tree corresponds to a $\vec{s} \in \mathcal{S}$ and stores (i) *Feature*($\vec{s}$), (ii) a bit $osc(\vec{s})$ indicating whether $\vec{s}$ is oscillating, (iii) the segmented approximation $\overrightarrow{s^{seg}}$ of $\vec{s}$, and (iv) a pointer $s.id$ to the exact representation of $\vec{s}$. Each entry in a directory node of the R*-tree contains a 4-dimensional minimum bounding box, enclosing the entries in the node pointed by it.

Figure 13 describes a multi-step process for evaluating range similarity queries using this indexing scheme. First, *Feature*($\vec{q}$) and $\overrightarrow{q^{seg}}$ are constructed for the query vector $\vec{q}$ (line 1). Then, an L_∞ range query is applied first and the results are passed through the Lb_glob filter (line 4) and an Lb_seg filter (line 5) that applies on the segmented representations of the sequences (i.e., Lb_seg1, Lb_seg2, or Lb_seg3). If the lower bound is smaller than ϵ, the identifier of $\vec{s}$ is added to a candidate set $\mathcal{C}$. Finally, the exact representations of all candidates in $\vec{s} \in \mathcal{C}$ are accessed to verify whether $D_{dtw}(\vec{q}, \vec{s}) \leq \epsilon$ (line 8).[5] In summary, the sequences in $\mathcal{S}$ are passed through two filters (i) Lb_glob, with the help of the index and feature vectors and (ii) Lb_seg, with the help of the segmented representations, before the expensive DTW distance is computed.

Figure 14 shows a pseudo-code for the neighbor search algorithm, which is processed by employing the same filter and refinement steps in combination with the incremental nearest neighbor search algorithm of (Hjaltason & Samet, 1999), as suggested by the multi-step paradigm of Seidl & Kriegel (1998). A priority queue is used to organize R*-tree node entries and sequences $\vec{s}$ based on their lower bounding distance from $\vec{q}$ computed so far. Initially, the R*-tree root entries are enqueued. If the next dequeued heap entry e is a directory R*-tree entry, the corresponding node is loaded and its entries are enqueued. If e is a leaf entry corresponding to $\vec{s}$, $Lb\_glob(\vec{q}, \vec{s})$ is computed and re-enqueued. If it is some $\vec{s}$, for which $Lb\_glob(\vec{q}, \vec{s})$ has already been computed, $Lb\_seg(\vec{q}, \vec{s})$ is computed and $\vec{s}$ is re-enqueued. If the next heap entry is a $\vec{s}$ for which $Lb\_seg(\vec{q}, \vec{s})$ has been computed, then $D_{dtw}(\vec{q}, \vec{s})$ is computed and $\vec{s}$ is re-enqueued. Finally, if $D_{dtw}(\vec{q}, \vec{s})$ has been computed for the dequeued entry $\vec{s}$, we know that this is the next nearest neighbor. The process continues until k neighbors have been dequeued.

```
Algorithm NNSimSearch(query $\vec{q}$, $\mathcal{S}$)
1.  compute $Feature(\vec{q})$ and $\overrightarrow{q^{seg}}$;
2.  initialize a priority queue $Q$;
3.  enqueue all root entries of the R*–tree to $Q$ organized by
        the $L_\infty$ distance of their MBRs from $Feature(\vec{q})$;
4.  while (not empty $Q$)
5.      $e$:=$dequeue(Q)$;
6.      if $e$ is a directory R*–tree entry then
7.          load corresponding R*–tree node and enqueue its entries in $Q$;
8.      else if $e$ is a leaf R*–tree entry seen for the first time then
9.          compute $Lb\_glob(\vec{q}, \vec{s})$ for the corresponding $\vec{s}$;
10.         enqueue $e$ with its Lb_glob distance to $Q$;
11.     else if $e$ is a leaf R*–tree entry seen for the second time then
12.         compute $Lb\_seg(\vec{q}, \vec{s})$ for the corresponding $\vec{s}$;
13.         enqueue $e$ with its Lb_seg distance to $Q$;
14.     else if $e$ is a leaf R*–tree entry seen for the third time then
15.         access the corresponding $\vec{s}$ and compute $D_{dtw}(\vec{q}, \vec{s})$;
16.         enqueue $\vec{s}$ with its DTW distance to $Q$;
17.     else /* $e$ is a sequence whose DTW has been computed
18.         output $\vec{s}$ as next nearest neighbor;
```

Figure 14. Nearest neighbor search algorithm.

3.4.1. Improving indexing with two R*-trees. The indexing scheme described above can be further improved by using the Lb_glob filters at an earlier stage. To achieve this, we build two separate $R*$-trees, *S_osc* and *S_non_osc*. The $R*$-tree *S_osc* indexes all sequences which are oscillating, while the $R*$-tree *S_non_osc* indexes non-oscillating data sequences.

Figure 15 describes the process for evaluating range similarity queries using this improved indexing scheme. First, *Feature*($\vec{q}$) and $\overrightarrow{q^{seg}}$ are constructed for the query vector $\vec{q}$. Then, we perform the range query on *S_osc* (lines 2–7). The algorithm distinguishes two cases. If $\vec{q}$ is oscillating, then it suffices to perform an ϵ-range query on the index, using Euclidean distance, in order to obtain all $\vec{s} \in \mathcal{S}$ which pass the Lb_glob filter (line 3). On the segmented representation of each of those sequences, Lb_seg is then applied. If the lower bound is smaller than ϵ, the identifier of $\vec{s}$ is added to a candidate set $\mathcal{C}$. In the case where $\vec{q}$ is not oscillating, an L_∞ range query is applied first and then the results are passed through the Lb_glob and Lb_seg filters, before they are added to $\mathcal{C}$. After processing the data index by *S_osc*, the range query is also performed on the *S_non_osc*, where the procedure is the same as described in algorithm *RSimSearch*. Finally, the exact representations of all candidates in $\vec{s} \in \mathcal{C}$ are accessed to verify whether $D_{dtw}(\vec{q}, \vec{s}) \leq \epsilon$.

The major difference between two indexing schemes is that some data sequences can be filtered earlier (lines 2–4) if $\vec{q}$ is oscillating, when using two indices instead of one. By indexing oscillating data with a separate $R*$-tree, the application of the Lb_glob filter is combined with the $R*$-tree search, which is now more efficient, since L_2 restricts the search space more compared to L_∞.

Nearest neighbor search can also be improved by using two indices. The two trees are accessed concurrently, but the entries from *S_osc* are organized based on their L_2 distance

```
Algorithm RSimSearch2(query $\vec{q}$, $\mathcal{S}$, $\epsilon$)
1.    compute $Feature(\vec{q})$ and $\overrightarrow{q^{seg}}$;
      /* perform range query on $S\_osc$ */
2.    if $\vec{q}$ is oscillating then
3.        perform an $L_2$ $\epsilon$ range query to get all $\vec{s}$
                such that $L_2(Feature(\vec{q}), Feature(\vec{s})) \leq \epsilon$;
4.        put $s.id$ of each result $\vec{s}$ to $\mathcal{C}$ if $Lb\_seg(\vec{q}, \vec{s}) \leq \epsilon$;
5.    else /* $\vec{q}$ is NOT oscillating */
6.        perform an $L_\infty$ $\epsilon$ range query to get all $\vec{s}$
                such that $L_\infty(Feature(\vec{q}), Feature(\vec{s})) \leq \epsilon$;
7         put $s.id$ of each result $\vec{s}$ to $\mathcal{C}$ if:
                (i) $Lb\_glob(\vec{q}, \vec{s}) \leq \epsilon$ and
                (ii) $Lb\_seg(\vec{q}, \vec{s}) \leq \epsilon$;
      /* perform range query on $S\_non\_osc$ */
8.    perform an $L_\infty$ $\epsilon$ range query to get all $\vec{s}$
          such that $L_\infty(Feature((\vec{q})), Feature((\vec{s}))) \leq \epsilon$;
9.    put $s.id$ of each result $\vec{s}$ to $\mathcal{C}$ if:
          (i) $Lb\_glob(\vec{q}, \vec{s}) \leq \epsilon$ and
          (ii) $Lb\_seg(\vec{q}, \vec{s}) \leq \epsilon$;
      /* retrieve full sequences from database */
10.   for each $s.id \in \mathcal{C}$
11.       access exact representation of $\vec{s}$;
12.       if $D_{dtw}(\vec{q}, \vec{s}) \leq \epsilon$ then report $\vec{s}$;
```

Figure 15. Range similarity search using two indices.

from *Feature*($\vec{q}$), whereas the entries from *S_non_osc* are organized using L_∞, assuming that $\vec{q}$ is oscillating. When a leaf entry from *S_osc* is dequeued, Lb_glob needs not be tested (Lb_seg is immediately computed), whereas leaf entries from *S_non_osc* must pass the Lb_glob filter and be re-inserted. If $\vec{q}$ is not oscillating, entries from both trees are treated in the same way and the Lb_glob filter is always used, as in the algorithm of figure 14.

4. Experimental evaluation

We evaluate the performance of the proposed methodology by using a wide range of real datasets obtained from the UCR Time Series Data Mining Archive (Keogh, 2002b). The datasets, listed in Table 8, cover a variety of disciplines including finance, medicine, chemistry, astronomy, etc. The third column of Table 8 shows the length of each sequence in the dataset and the number of sequences in it.

In order to demonstrate the generality of our methodology, we conducted two sets of experiments based on whether there is an envelope constraint or not. First, we compared the tightness (i.e., pruning effectiveness) of the proposed bounds Lb_seg1, Lb_seg2, Lb_seg3, and Lb_glob against Lb_Yi (Yi, Jagadish, & Faloutsos, 1998), Lb_Kim (Kim, Park, and Chu, 2001), Lb_Keogh (Keogh, 2002a), and Lb_PAA (Keogh, 2002a; Zhu & Shasha, 2003). In addition, we compared the search performance of our methodology to that of previous methods (Yi, Jagadish, & Faloutsos, 1998; Kim, Park, and Chu, 2001; Keogh, 2002a; Zhu

Table 8. Tightness without envelope constraints.

ID	Name	length*no	Lb_Kim	Lb_glob	Lb_Yi	Lb_seg1	Lb_seg2
1	Sunspot	2899*1	0.20	0.25	0.28	0.18	**0.34**
2	Power	35040*1	0.18	0.21	0.23	0.36	**0.44**
3	Spot Exrates	2567*12	0.16	0.23	**0.71**	0.43	0.62
4	Shuttle	1000*6	0.16	0.22	0.58	0.63	**0.72**
5	Water	2191*3	0.47	0.48	0.65	0.70	**0.74**
6	Chaotic	1800*1	0.19	0.23	0.24	0.26	**0.36**
7	Steamgen	9600*4	0.14	0.19	**0.64**	0.37	0.56
8	Ocean	4096*1	0.16	0.20	0.33	0.52	**0.56**
9	Tide	8746*1	0.23	0.29	0.33	0.22	**0.40**
10	CSTR	7500*3	0.15	0.22	0.64	0.44	**0.65**
11	Winding	2500*7	0.21	0.26	0.22	0.21	**0.37**
12	Dryer2	867*6	0.16	0.25	**0.63**	0.31	0.56
13	Robot Arm	1024*2	0.2	0.25	0.2	0.12	**0.34**
14	Ph Data	2001*3	0.14	0.18	**0.53**	0.28	0.49
15	Power Plant	2400*1	0.14	0.20	0.29	0.44	**0.53**
16	Evaporator	6305*6	0.18	0.23	0.2	0.11	**0.27**
17	Ballbeam	1000*2	0.16	0.21	0.57	0.40	**0.60**
18	Fetal ECG	2500*9	0.45	0.59	0.49	0.76	**0.79**
19	Balloon	2001*2	0.24	0.29	0.23	0.57	**0.62**
20	Stand'& Poor	17610*1	0.13	0.24	0.81	0.47	**0.82**
21	Speech	1020*1	0.25	0.29	0.15	0.12	**0.34**
22	Soil Temp	2306*1	0.16	0.23	0.26	0.23	**0.31**
23	Wool	310*9	0.14	0.23	**0.81**	0.45	0.74
24	Infrasound	8192*1	0.17	0.22	**0.62**	0.38	0.53
25	EEG	512*22	0.18	0.23	0.15	0.15	**0.29**
26	Koski EEG	144002*1	0.25	0.28	0.31	0.41	**0.56**
27	Buoy Sensor	13991*4	0.18	0.23	**0.45**	0.29	0.44
28	Burst	9382*1	0.16	0.21	**0.54**	0.40	**0.54**
29	Random Walk	65536*1	0.18	0.23	0.27	0.37	**0.45**

& Shasha, 2003), which utilize the aforementioned bounds and appropriate indices. The implementation language was C++ and the experiments were performed on a Pentium III 700 MHz workstation, running Unix.

4.1. Tightness of the lower bounds

In the first set of experiments, we compared the *tightness* of the proposed lower bounds Lb_seg1, Lb_seg2, Lb_seg3, and Lb_glob against Lb_Yi (Yi, Jagadish, & Faloutsos, 1998), Lb_Kim (Kim, Park, and Chu, 2001), Lb_Keogh (Keogh, 2002a), and Lb_PAA (Zhu & Shasha, 2003). Let $\vec{q}$ and $\vec{s}$ be two sequences. The tightness T of a lower bound $Lb(\vec{q}, \vec{s})$

of $D_{dtw}(\vec{q}, \vec{s})$ is defined in Keogh (2002a) and Zhu & Shasha (2003) by:

$$T = \frac{Lb(\vec{q}, \vec{s})}{D_{dtw}(\vec{q}, \vec{s})} \tag{13}$$

T is in the range of [0, 1], because both $Lb(\vec{q}, \vec{s})$ and $D_{dtw}(\vec{q}, \vec{s})$ are positive and $Lb(\vec{q}, \vec{s})$ must be no greater than $D_{dtw}(\vec{q}, \vec{s})$. The higher T is, the tighter $Lb(\vec{q}, \vec{s})$ is. For each dataset of Table 8 we first randomly extracted a set S of 50 subsequences[6] of length $m = 256$. Each sequence was normalized by subtracting from each value the mean of the sequence, so that the resulting sequence has mean 0. Then for each distinct pair (i, j) of sequences, where $1 \leq i < j \leq 50$, we set $\vec{q} = S[i]$, $\vec{s} = S[j]$ and computed the tightness of each bound, according to Eq. (13). Finally, for each bound we averaged its tightness over all pairs. The same methodology was used to evaluate tightness in Keogh (2002a) and Zhu & Shasha (2003). As a warping constraint, we used a Sakoe-Chiba band with width coefficient $w = 0.1$. The PAA and APCA segmentations used by Lb_PAA and our Lb_seg bounds were 16-dimensional (i.e., $N = M = 16$). Table 8 compares the tightness of the bounds that can be applied without a warping (i.e., envelope) constraint. Lb_seg2 is clearly the tightmost bound, being on the average 1.435 times tighter than Lb_Yi and 2.5 times tighter than Lb_Kim. As expected, it is consistently tighter than Lb_seg1, whereas it it (slightly) less tight than Lb_Yi in only few datasets. Moreover, as expected, Lb_glob (our indexable bound) is consistently tighter than Lb_Kim.

Table 9 compares the tightness of various bounds for the case when there exists an envelope constraint, using the same sequences as in Table 8. In this case, the numbers for Lb_Kim, Lb_glob, Lb_Yi, and Lb_seg2 are different from those in Table 8 because the optimal warping distance in the constraint envelope may be larger than the unconstrained one. As expected, Lb_seg3 is always tighter than Lb_seg2, since it considers the envelope constraints. Moreover, on the average, Lb_seg3 is 1.53 times tighter than Lb_PAA and 1.01 times as tight as Lb_Keogh (which, however, does not facilitate indexing). In addition, Lb_glob always improves upon Lb_Kim. In summary, our bounds Lb_seg2 and Lb_seg3 are much tighter than the previous techniques, in the presence or not of envelope constraints. In the next section, we show how this affects the search efficiency of our methodology.

4.2. Pruning power and response time

We evaluated our bounds and the indexing scheme discussed in Section 3.4, by comparing their performance with the previous approaches for DTW with and without envelope constraints. The previous techniques included in the evaluation are (i) indexed search using Lb_Kim (Kim, Park, and Chu, 2001) (denoted by KimSearch) (ii) linear scan using Lb_Yi (Yi, Jagadish, & Faloutsos, 1998) (denoted by YiSearch),[7] (iii) linear scan using Lb_Keogh (Keogh, 2002a) (denoted by KeoghSearch), and (iv) indexed search using Lb_PAA (Keogh, 2002a; Zhu & Shasha, 2003) (denoted by PAASearch). We also included (v) indexed search using Lb_glob and Lb_seg2 (denoted by SegSearch2), (vi) indexed search using Lb_glob and Lb_seg3 (denoted by SegSearch3), and (vii) indexed search using Lb_glob only (denoted

Table 9. Tightness with envelope constraints.

ID	Lb_Kim	Lb_glob	Lb_Yi	Lb_Keogh	Lb_PAA	Lb_seg2	Lb_seg3
1	0.15	0.25	0.22	**0.5**	0.36	0.23	0.37
2	0.14	0.2	0.23	**0.59**	0.5	0.33	0.52
3	0.14	0.22	0.67	0.67	0.63	0.55	**0.70**
4	0.14	0.2	0.55	0.8	0.76	0.57	**0.92**
5	0.43	0.47	0.65	0.6	0.37	0.64	**0.78**
6	0.17	0.23	0.23	**0.48**	0.25	0.28	0.37
7	0.12	0.19	0.63	**0.74**	0.7	0.59	0.7
8	0.13	0.2	0.31	**0.77**	0.75	0.42	0.63
9	0.2	0.27	0.31	**0.48**	0.23	0.34	0.47
10	0.13	0.21	0.61	**0.68**	0.61	0.57	**0.68**
11	0.18	0.23	0.22	0.33	0.11	0.33	**0.38**
12	0.14	0.23	**0.6**	**0.6**	0.46	0.5	0.59
13	0.19	0.23	0.2	0.23	0	0.29	**0.35**
14	0.12	0.17	**0.52**	0.46	0.33	0.42	0.48
15	0.11	0.18	0.28	**0.67**	0.58	0.41	0.6
16	0.18	0.2	0.19	0.23	0.06	0.25	**0.28**
17	0.14	0.21	0.51	0.58	0.49	0.42	**0.65**
18	0.28	0.58	0.33	0.59	0.01	0.44	**0.85**
19	0.21	0.29	0.19	0.44	0.02	0.46	**0.71**
20	0.12	0.22	0.8	0.8	0.75	0.77	**0.85**
21	0.21	0.29	0.13	0.38	0.01	0.28	**0.39**
22	0.15	0.21	0.25	**0.35**	0.09	0.27	0.33
23	0.13	0.23	**0.79**	0.73	0.67	0.69	0.75
24	0.14	0.21	0.55	**0.64**	0.59	0.5	0.63
25	0.17	0.22	0.15	**0.32**	0.08	0.25	0.30
26	0.18	0.27	0.31	0.63	0.49	0.39	**0.77**
27	0.16	0.23	0.43	**0.49**	0.31	0.34	0.44
28	0.13	0.2	0.49	**0.71**	0.68	0.5	0.63
29	0.15	0.23	0.25	**0.65**	0.59	0.37	0.51

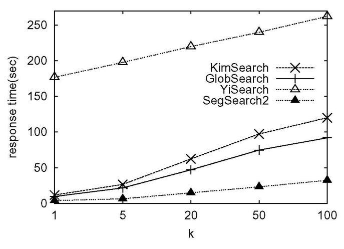
by GlobSearch). GlobSearch corresponds to a technique which directly applies DTW after Lb_glob filtering (without filtering by some Lb_seg before DTW).

For this set of experiments, we used Koski EEG, the longest sequence from Table 8 and Fetal ECG, a dataset with special properties, to be discussed in Section 4.2.2. We extracted all subsequences of length $m = 256$ from it, inserted a random subset 50 of subsequences in a query set, and constructed a dataset S for the remaining ones. Unless otherwise stated, the number of segments used in the segmental approximations was $M = 16$. We then evaluated the performance of the various techniques for range and nearest neighbor queries. Performance is measured in terms of two factors. The first is the average response time for each query. The second, referred to as *candidate ratio*, is the percentage of sequences for which the exact (and expensive) D_{dtw} must be computed for each query. In each experimental instance, we averaged these performance measures over all tested 50 queries.

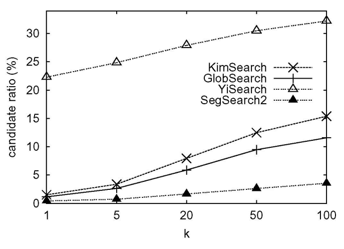
4.2.1. Queries without warping constraints. In the first set of experiments we compare the methods for queries without envelope constraints. We include KimSearch, YiSearch, SegSearch2, and GlobSearch3; these are the only methods that can be applied in this case. figure 16(a) plots the execution cost of applying k-NN queries on the Koski EEG dataset using the four methods. Figure 16(b) plots the performance of the four methods on the same dataset in terms of the percentage of time sequences for which exact DTW has to be applied (*candidate ratio*). Note that the response time is directly proportional to this percentage. Thus, the burden of all algorithms is the number of exact DTW computations, which is greatly reduced thanks to the tight lower bounds. Observe that SegSearch2 is several times faster than the previous similarity search techniques that do not use envelope constraints. YiSearch is not efficient, since it does not employ any index; Lb_Yi has to be applied on all sequences of the database (linear scan) at a non-trivial cost. In addition, figure 16(b) suggests that it is not an effective bound, as a large percentage of DTW computations have to be applied. KimSearch, on the other hand, can effectively prune large parts of the database, since search is directed by the tree and a close NN used for pruning can be found fast. On the other hand, it is looser than our GlobSearch improvement. Finally, the additional application of the Lb_seg2 filter on the candidates found by GlobSearch (i.e., method SegSearch2) further improves search by drastically reducing the number of sequences on which exact DTW has to be applied. Figures 16(c) and (d) show similar trends for range similarity queries. For other datasets such as Fetal ECG (Figures 16(e) to (h)), we derive similar conclusions. We have also included a comparison on the Steamgen dataset, where Lb_Yi is tighter than Lb_seg2 (see Table 8). Observe that even in this case SegSearch2 is much faster than Lb_Yi due to the effectiveness of the index.

4.2.2. Queries with warping constraints. Next, we compare methods that are applicable in the presence of warping constraints. Figures 17(a) to (d) compare the six methods (SegSearch2 is omitted since it is always not better than SegSearch3) for k-NN and range ϵ-range queries on the Koski EEG dataset. Note that SegSearch3 dominates over all other methods. YiSearch is the slowest method, just as for search without warping constraints. On the other hand, KeoghSearch is quite fast, even though it is based on linear scan, since Lb_Keogh is a very tight bound. KimSearch and GlobSearch are fast for small k and ϵ, but their performance degrades for large numbers. Finally, PAASearch is quite efficient, but not as good as SegSearch3 and KeoghSearch. First, the approximations used by PAASeach do not provide very accurate bounds, compared to the APCA segments used by SegSearch3. Second, the area defined of the approximated bounding envelope of $\vec{q}$ covers a large area of the search space and accesses many candidate sequences.

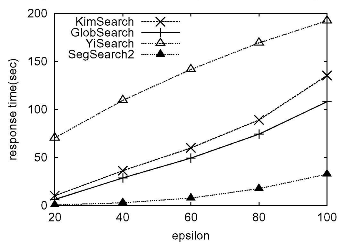
Figures 17(e) to (h) confirm that the PAA-based bounding envelope of $\vec{q}$ can cover a large area. This time, we used another dataset (Fetal ECG), where PAASearch has very poor performance. Note that Lb_PAA in this case hardly prunes any sequence during search (see also Table 9), on most of which exact DTW has to be applied. The sequences in this dataset have special properties where each sequence is quasi-periodic, with no fixed width; every positive spike is immediately followed by a negative one of the same width. Such sequences



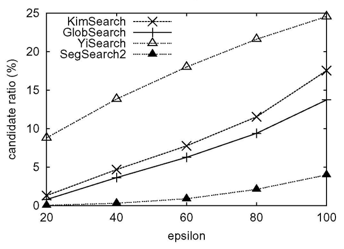
(a) Koski EEG, k-NN (Response Time)



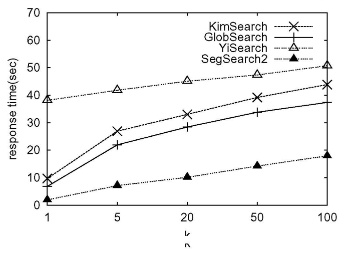
(b) Koski EEG, k-NN (Candidate Ratio)



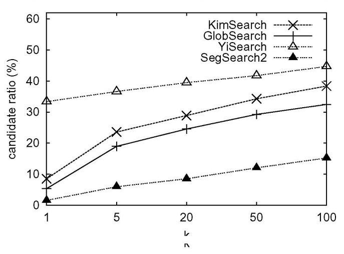
(c) Koski EEG, ϵ-range (Response Time)



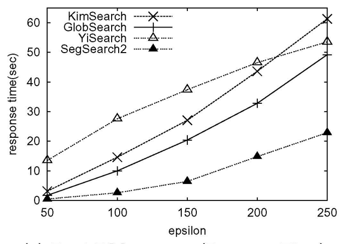
(d) Koski EEG, ϵ-range (Candidate Ratio)



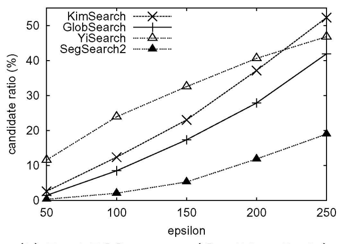
(e) Fetal ECG, k-NN (Response Time)



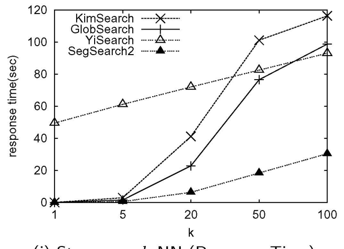
(f) Fetal ECG, k-NN (Candidate Ratio)



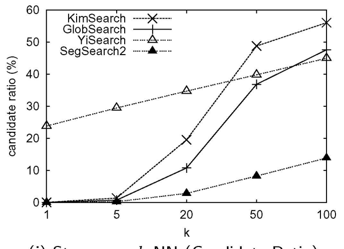
(g) Fetal ECG, ϵ-range (Response Time)



(h) Fetal ECG, ϵ-range (Candidate Ratio)

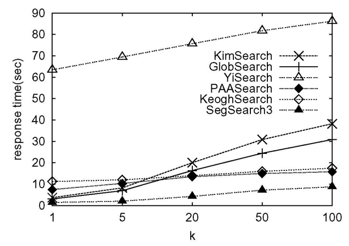


(i) Steamgen, k-NN (Response Time)

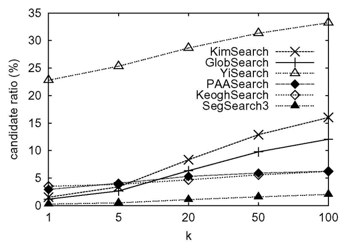


(j) Steamgen, k-NN (Candidate Ratio)

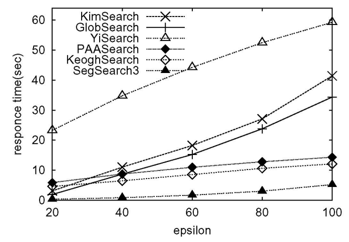
Figure 16. Performance comparison of search techniques with no envelope constraints.



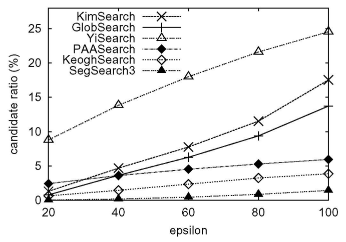
(a) Koski EEG, k-NN (Response Time)



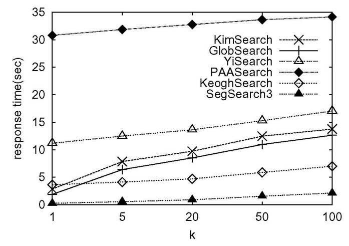
(b) Koski EEG, k-NN (Candidate Ratio)



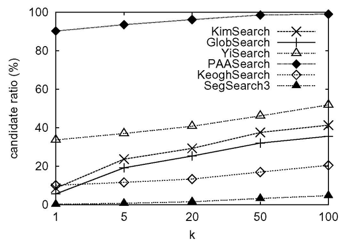
(c) Koski EEG, ϵ-range (Response Time)



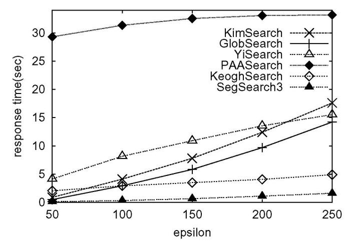
(d) Koski EEG, ϵ-range (Candidate Ratio)



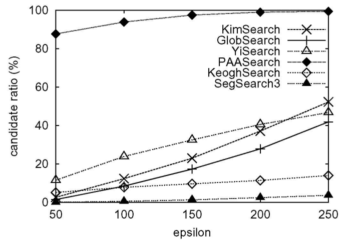
(e) Fetal ECG, k-NN (Response Time)



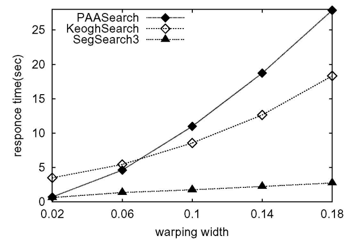
(f) Fetal ECG, k-NN (Candidate Ratio)



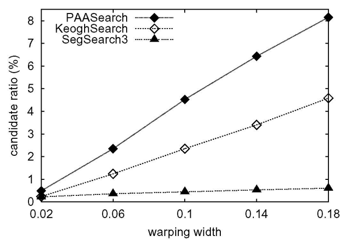
(g) Fetal ECG, ϵ-range (Response Time)



(h) Fetal ECG, ϵ-range (Candidate Ratio)



(i) Koski EEG, ϵ-range (Response Time, $\epsilon = 60$)



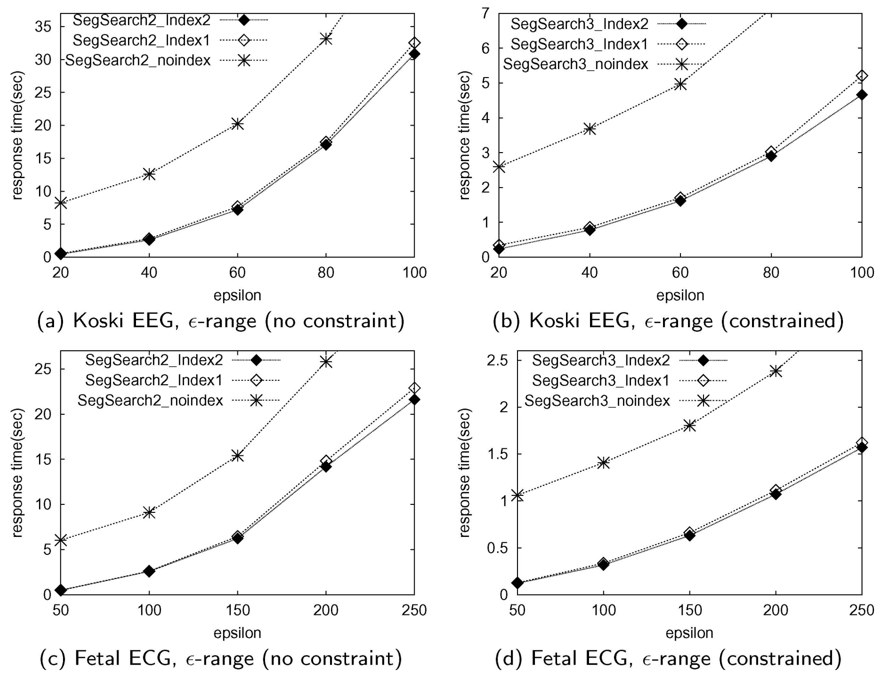
(j) Koski EEG, ϵ-range (Candidate Ratio, $\epsilon = 60$)

Figure 17. Performance comparison of search techniques with envelope constraints.

are very common in medical applications, e.g., Fetal ECG contains cardiograms of patients. In such cases, most PAA segments have the same average, and the approximation of any sequence is close to a flat line. On the other hand, APCA is a more flexible transformation, since it uses more segments for areas of high variance.

Finally, we compared the performance of PAASeach, KeoghSearch, and SegSearch3 as a function of the width coefficient w that constrains the warping path. Figures 17(i) and (j) show that the performance of SegSearch3 is not affected as much by w as KeoghSearch and PAASeach. The performance of PAASeach degrades especially fast, due to the looseness of the warping envelope approximations. In summary, our methodology is much faster compared to previous methods for DTW similarity queries with and without envelope constraints.

4.2.3. Indexing efficiency. In the next experiment, we compare the efficiency of the two indexing schemes proposed in Section 3.4. The first uses a single R*-tree, whereas the second scheme divides the sequences to oscillating and non-oscillating ones and indexes each set by a separate R*-tree, as discussed in Section 3.4.1. In order to show the effectiveness of indexing, we also included two linear scan approaches SegSearch2_noindex and SegSearch3_noindex, that apply the corresponding filters with linear scan. Figures 18(a) and (b) show the performance of SegSearch2 (no warping constraint) and SegSearch3 (warping constraint $w = 0.1$), when applied on one or two indices of Koski EEG (denoted by



(a) Koski EEG, ϵ-range (no constraint)
(b) Koski EEG, ϵ-range (constrained)
(c) Fetal ECG, ϵ-range (no constraint)
(d) Fetal ECG, ϵ-range (constrained)

Figure 18. Performance comparison of the two indexing schemes.

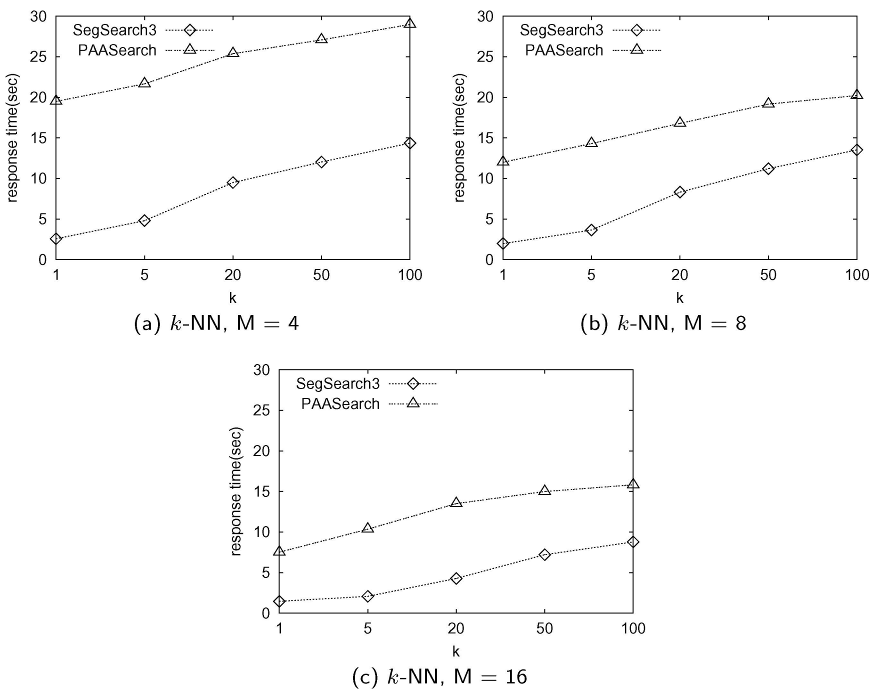


Figure 19. k-NN queries for $M = 4, 8, 16$.

Index1 and Index2, respectively). Using two indices provides a slight performance advantage, since L_2 manages to prune more space compared L_∞ during search. The experiments also demonstrate that using the index before the filter results in much lower response time, compared to using the filters after linear scan. The results for Fetal ECG are similar as shown in figure 18(c) and (d). We only compare response time, since the candidate ratio for exact DTW is not sensitive to the index used. Finally, we did not include experiments on NN search, since they also show similar results.

4.2.4. Effect of the number of segments. In the next experiment, we compare the performances of SegSearch3 and PAASearch as a function of M, the number of segments used by the methods to approximate the sequences. Other methods do not use segmental approximations or other approximation techniques, so we only include these two in the comparison. The experiments were performed, using the Koski EEG dataset. Figures 19 and 20 compare the two algorithms for three values of M on nearest neighbor and range search queries. Observe that our method maintains a performance advantage over PAASearch for different values of M. For the range of tested M values, the efficiency of the two techniques increases with M, since the segmented representations are more accurate and exact DTW must be applied on fewer candidates. Nevertheless, too high values for M degenerate the R^*-tree index (used by PAASearch) and make SDTW (used by SegSearch3) less efficient. We

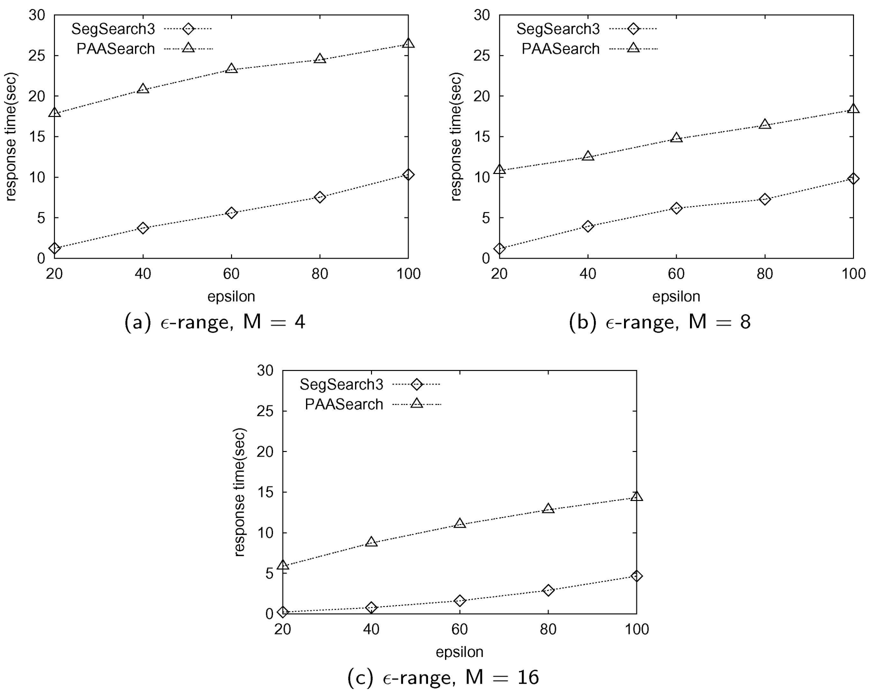


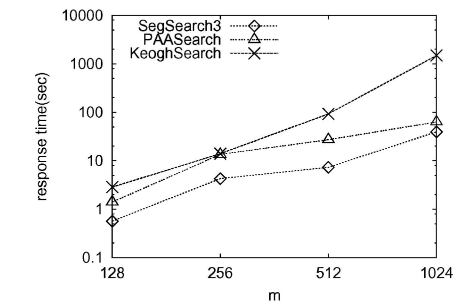
Figure 20. ϵ-range queries for $M = 4, 8, 16$.

have found that $M = 16$ is a good trade-off between approximation accuracy and filtering efficiency.

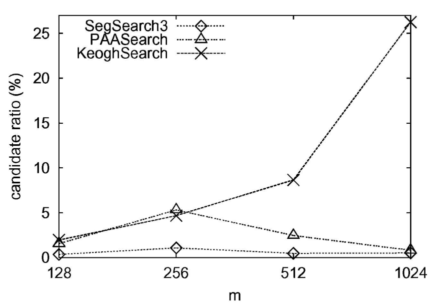
4.2.5. Effect of the sequence length. We also tested the robustness of our technique as a function of the lengths of the sequences in $\mathcal{S}$. Again, we used Koski EEG, set $M = 16$, and constructed four datasets, extracting all subsequences of length $m = 128$, 256, 512, and 1024, respectively and excluding a random sample of 50 queries from them.

Figure 21 compares the relative performance of SegSearch3, PAASearch, and KeoghSearch for nearest neighbor ($k = 20$) and range search queries ($\epsilon = 40$) on the three collections of sequences. We did not include experiments for similarity search without warping constraints, since we found SegSearch2 much faster compared to previous methods that are applicable in this case (see Section 4.2.1). Also, we did not include YiSearch, KimSearch, and GlobSearch due to their inferior performance.

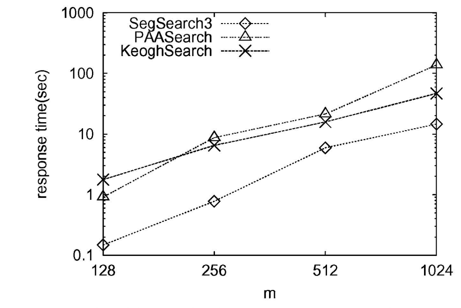
The results show that SegSearch3 is faster than its competitors in all cases. Observe that for both k-NN and range queries, the performance of the three methods degrades fast with m mainly because the cost of DTW increases quadratically with m. On the other hand, from the diagrams (b) and (d), we can see that the effectiveness of the PAA and APCA approximations is not affected much by the lengths of the sequences. For



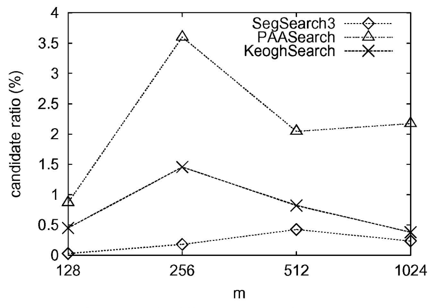
(a) Koski EEG, k-NN ($k = 20$)



(b) Koski EEG, k-NN ($k = 20$)



(c) Fetal ECG, ϵ-range ($\epsilon = 40$)



(d) Fetal ECG, ϵ-range ($\epsilon = 40$)

Figure 21. Performance as a function of m.

k-NN search, we observe that it becomes harder for KeoghSearch to find a good nearest-neighbor bound that can prune the search space early, using the effective Lb_Keogh. This can be attributed to the "curse of dimensionality" effect; for large m, most sequences have large distances to $\vec{q}$ and it takes longer to find the close neighbors. On the other hand, PAASearch and SegSearch3 are affected less by this problem, since they use indices to guide search.

4.2.6. Using Lb_Keogh with Lb_glob indices. As we have seen in Table 9, Lb_Keogh is a tight bound, however, we have used it so far only in combination with linear scan. We can replace Lb_seg by Lb_Keogh in the algorithms of Section 3.4 in order to make better use of the bound. In other words, we can use *Feature*($\vec{s}$) to index each sequence $\vec{s}$ using an $R*$-tree, and for each candidate that passes Lb_glob we can apply Lb_Keogh (instead of Lb_seg) as a filter prior to DTW matching. In this section, we compare the efficiency of this search method, denoted by GlobKeoghSearch, against SegSearch3.

Figure 22 shows the performance of the two schemes for queries on the Koski EEG and Fetal ECG datasets. Observe that SegSearch3 is faster than GlobKeoghSearch for the tested queries. We expect the performance of GlobKeoghSearch to degrade with the warping width constraint w, since Lb_Keogh more sensitive to it compared to our Lb_seg3 bound, as shown in Section 4.2.2.

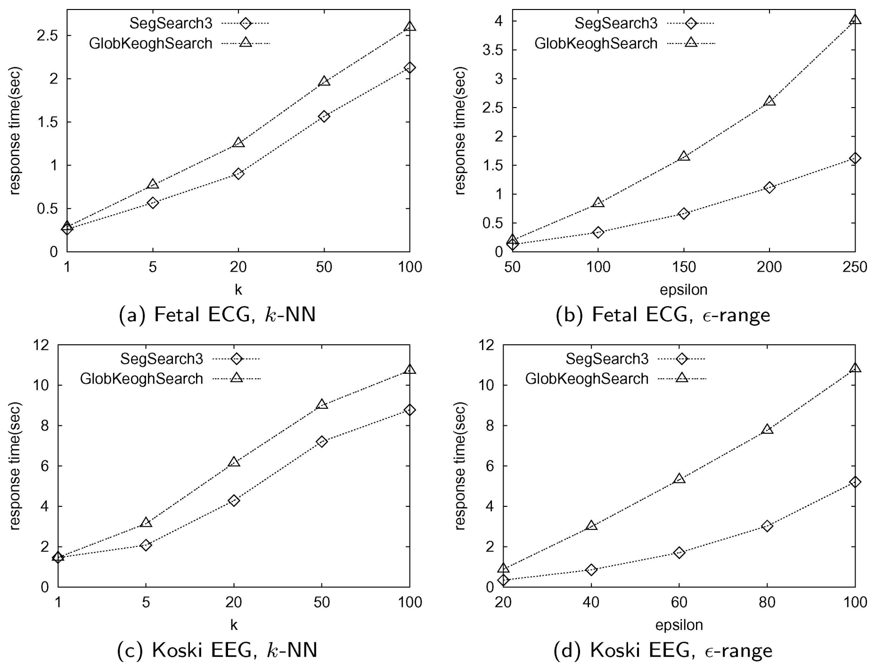


Figure 22. Effect of using Lb_Keogh filter after GlobSearch.

5. Conclusions

In this paper, we presented an efficient methodology for indexing and querying time series using dynamic time warping (DTW). We have shown how to use APCA approximations to compute fast three progressively more tight lower bounds for DTW distance, which are based or not on warping path constraints. In addition, we showed an improvement of the Lb_Kim global bound (Kim, Park, and Chu, 2001), which can be used to index the segmentations. Finally, we proposed a multi-step query processing technique that applies two levels of filtering, minimizing the search effort and the number of sequences on which exact DTW has to be applied.

The proposed methodology is generic enough to be applied in the case where no warping constraints are specified, whereas it can also be optimized to take advantage of warping constraints. It was compared with previous state-of-the-art methods (Yi, Jagadish, & Faloutsos, 1998; Kim, Park, and Chu, 2001; Keogh, 2002a; Zhu & Shasha, 2003) and found consistently superior to them, for both cases where warping path constraints exist or not.

The main contributions of this work can be summarized as follows:

- This is the first work which uses APCA approximations to compute fast progressively tight lower bounds for DTW distance. Due to the nice properties of APCA, which can approximate sequence parts of low variance with few segments and parts of high variance with many segments, the proposed lower bounds are still very tight for the data sets with special features (i.e., quasi-periodic); while Keogh's and Zhu's lower bounds using PAA representations are much looser for these datasets.

- We have proposed a multi-step query processing technique, which applies two levels of filtering. The advantage of this approach is that it is able to filter out unqualified sequences at very low cost at the first step, and use another much refined filter to make second level filtering, such that the search effort and the number of sequences on which exact DTW has to be applied is minimized.
- We have conducted an thorough experimental study, which suggests that our multi-step query processing technique is more efficient and robust than existing state-of-the-art methods. Our proposed indexing and filtering schemes are consistently superior for various types and lengths of sequences and different numbers of segments used to represent the sequences. Finally, our method adapts well to the presence and extent of envelope constraints.

In the future, we plan to further optimize this multi-step query processing methodology by exploring the application of additional lower-bounds and filters. In addition, we plan to test its efficiency as a module of classification and clustering algorithms that use DTW to measure similarity.

Notes

1. The subsequence matching problem is converted to the whole matching problem by employing sliding window techniques (Faloutsos, Ranganathan, & Manolopoulos, 1994; Moon, Whang, & Han, 2002).
2. The value of a cell (i, j) in the warping matrix WM is the warping distance $D(i, j)$, however, each element of a warping path W corresponds to $d(i, j)$; W accumulates these distances.
3. If all elements of one sequence are larger than all elements of the other, we should set min_max = max_min, before the computation of Lb_Yi.
4. The term Segmented Dynamic Time Warping was used in Keogh, and Pazzani (1999) to describe an approximate DTW distance which, however, is not a lower bound. Our proposed SDTW derives a lower bound computed and used in a different way.
5. In fact, there is no need to construct $\mathcal{C}$, since the candidates can directly be accessed at the time they pass the lower bounding filters. $\mathcal{C}$ is included in figure 13 for the ease of presentation.
6. If a dataset consists of more than one long sequences, we first pick a random sequence and then a subsequence of it.
7. In Yi, Jagadish, & Faloutsos (1998) a multi-step similarity query processing method that uses Lb_Yi and a multi-dimensional index was proposed. However, it is not appropriate for *exact* DTW search, since it allows false dismissals of query results.

References

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. *Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms (FODO)* (pp. 69–84).

Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 322–331). Atlantic City, New Jersey.

Berndt, D. J. & Clifford, J. (1994). using dynamic time warping to find patterns in time series. *Proc. of AAAI Workshop: Knowledge Discovery in Databases* (pp. 359–370). Seattle, Washington, .

Chu, S., Keogh, E., Hart, D., & Pazzani, M. (2002). Iterative deepening dynamic time warping for time series. In: *Proc of SIAM International Conference on Data Mining*.

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 419–429). Minneapolis, Minnesota.

Goldin, D. & Kanellakis, P. (1995). On similarity queries for time-series data: constraint specification and implementation. *Proc. of the 1st Intl Conference on the Principles and Practice of Constraint Programming* (pp. 137–153). Cassis, France.

Hjaltason, G. & Samet, H. (1999). distance browsing in spatial databases. *ACM Transactions on Database Systems 24:2*, 265–318.

Kaufman, L. & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Keogh, E. (2002a). Exact indexing of dynamic time warping. *Proc. of 28th International Conference on Very Large Data Bases* (pp. 406–417). Hong Kong, China.

Keogh, E. (2002b). *The UCR Time Series Data Mining Archive*. Computer Science & Engineering Department, University of California, Riverside CA, http://www.cs.ucr.edu/˜eamonn/TSDMA/index.html.

Keogh, E., Chakrabarti, K., Mehrotra, S., & Pazzani, M. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 151–162). Santa Barbara, California, .

Keogh, E. & Pazzani, M. (1999). scaling up dynamic time warping to massive datasets. *Proc. of the Eur. Conf on Principles of Data Mining and Knowledge Discovery (PKDD)* (pp. 1–11). Prague, Czech Republic.

Kim, S. W., Park, S., & Chu, W. W. (2001). An indexed-based approach for similarity search supporting time warping in large sequence databases. *Proc. of the 17th International Conference on Data Engineering* (pp. 607–614). Heidelberg, Germany.

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

Moon, Y. S., Whang, K. Y., & Han, W. S. (2002). GeneralMatch: A subsequence matching method in time-series databases based on generalized windows. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 382–393). Madison, Wisconsin.

Munich, M. E. & Perona, P. (1999). continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. *Proc. of the 8 th Int'l Conf. on Computer Vision* (pp. 20–25). Corfu, Greece.

Rabiner, L. & Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Ratanamahatana, C. A. & Keogh, E. (2004). making time-series classification more accurate using learned constraints. *Proc. of SIAM International Conference on Data Mining (SDM '04)* (pp. 11–22). Lake Buena Vista, Florida.

Sakoe, H. & Chiba, S. (1978). dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing 26:1*, 43–49.

Seidl, T. & Kriegel, H.-P. (1998). Optimal multi-step k-nearest neighbor search. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 154–165). Seattle, Washington.

Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., & Keogh, E. (2003). indexing multi-dimensional time-series with support for multiple distance measures. *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 216–225). Washington, DC.

Yi, B., Jagadish, H., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. *Proc. of the Fourteenth International Conference on Data Engineering*. pp. 201–208, Orlando, Florida.

Zhu, Y. & Shasha, D. (2003). warping indexes with envelope transforms for query by humming. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data* (pp. 181–192). San Diego, California.

Received March 18, 2004
Revised October 9, 2004
Accepted October 9, 2004
Final manuscript October 9, 2004