Anticipatory DTW for Efficient Similarity Search in Time Series Databases

Ira Assent ° Marc Wichterich • Ralph Krieger • Hardy Kremer • Thomas Seidl •

°Aalborg University, Denmark
ira@cs.aau.dk

•RWTH Aachen University, Germany
{wichterich, krieger, kremer, seidl}@cs.rwth-aachen.de

ABSTRACT

Time series arise in many different applications in the form of sensor data, stocks data, videos, and other time-related information. Analysis of this data typically requires searching for similar time series in a database. Dynamic Time Warping (DTW) is a widely used high-quality distance measure for time series. As DTW is computationally expensive, efficient algorithms for fast computation are crucial.

In this paper, we propose a novel filter-and-refine DTW algorithm called Anticipatory DTW. Existing algorithms aim at efficiently finding similar time series by filtering the database and computing the DTW in the refinement step. Unlike these algorithms, our approach exploits previously unused information from the filter step during the refinement, allowing for faster rejection of false candidates. We characterize a class of applicable filters for our approach, which comprises state-of-the-art lower bounds of the DTW.

Our novel anticipatory pruning incurs hardly any overhead and no false dismissals. We demonstrate substantial efficiency improvements in thorough experiments on synthetic and real world time series databases and show that our technique is highly scalable to multivariate, long time series and wide DTW bands.

1. INTRODUCTION

Time series data is important for commerce, science, and engineering. It frequently serves as a basis for decision and policy-making. Large amounts of time-dependent data are created, acquired, and analyzed. Examples include stock market data analysis on prices of publicly traded securities and sensor-based monitoring of seismic activities.

The analysis of this data requires a notion of similarity between time series to determine alike patterns. A simple approach for comparing time series consists of aggregating the differences in values for each point in time (e.g., by using the L_2 norm). While computationally inexpensive, similarity measures that regard time series in a point-in-time by point-in-time manner are inadequate for many applications. A slight shift in the start time or in the length of a relevant pattern causes large differences and thus disproportionally low similarity scores.

The Dynamic Time Warping (DTW) concept was introduced in the field of speech recognition [5] to address the shortcomings that stem from inflexibility along the time axis. The DTW distance computes an optimal alignment of two time series by allowing for stretching and squeezing of the time series. This flexibility led to the adoption and adaptation of DTW for many application domains outside speech recognition [1, 7, 17, 2] and gave rise to research on efficient retrieval using DTW [29, 14, 32, 23, 22, 31, 4]. An extensive study of DTW and other time series distance measures can be found in [10].

The DTW distance can be computed using dynamic programming techniques. In this general formulation, the DTW is typically too costly for comparing longer time series as the computational complexity is quadratic in the length of the time series. An important parameter for DTW is the permissible amount of stretching and squeezing along the time axis, the k-band. A wider band allows more flexibility (useful for smaller databases) but also induces higher computational cost.

A further challenge for efficient DTW query processing lies in multivariate time series, where not just a single but several attributes are recorded for each point in time. Examples of multivariate time series include monitoring the overall state of a system that cannot sufficiently be described with only a single value. In a stock market scenario, a number of securities might be aggregated to a multivariate time series in order to watch for monetary shifts (e.g., from mortgage-backed securities to commodities).

In this work, we propose a novel approach called Anticipatory Pruning (AP) for speeding up DTW-based retrieval on both univariate and multivariate time series. Intuitively speaking, AP *anticipates* the DTW refinement result by exploiting previously computed information in multistep filter-and-refine algorithms.

Traditional filter-and-refine algorithms compute the full DTW if the filter distance does not exceed the current pruning distance. An existing improvement of this technique is known as early stopping [23] or abandoning [17], where the computation is stopped as soon as an intermediate result exceeds the pruning distance.

Our anticipatory approach goes beyond early stopping. In each step of DTW computation, we derive a lower-bounding anticipatory distance that takes the intermediate DTW distance computed so far and greatly improves the approxi-

VLDB '09, August 24-28, 2009, Lyon, France

mation quality of the lower bound by anticipating the remaining steps of the full DTW computation. This anticipatory component is derived from the previously computed filter step, thus re-using information that is ignored in existing approaches. In this manner, the incremental DTW computation is stopped as soon as the anticipatory distance exceeds the pruning threshold.

Anticipatory pruning is lossless for a certain class of lower bounding filters, which we characterize in this work. We prove that the most widely used state-of-the-art approaches are in this class. Intuitively, the filters have to be lower bounding themselves, *piecewise*, and *reversible* to be usable in our approach.

As anticipatory pruning is orthogonal to existing lower-bounding filtering, indexing, and dimensionality reduction techniques, it can be flexibly combined with such techniques for additional speed-up.

In the next section, we review related work. Section 3 introduces time series and the DTW as a distance function for similarity search on time series. We then detail how we derive our anticipatory pruning technique in Section 4. Experiments demonstrate the performance of anticipatory pruning in Section 5.

2. RELATED WORK

A number of approximate techniques have been proposed for speeding up DTW queries. In [29] an approximate embedding of DTW into Euclidean space is proposed. This technique is extended to a Haar wavelet transform based technique for fewer false positives [6], but possibly more false negatives. Iterative Deepening DTW computes different levels of dimensionality reduction from piecewise linear approximations [9]. Using a probabilistic model based on sample approximation errors, time series are either pruned or compared at a finer level of approximation. The FastDTW approach computes approximations of the warping path at different levels of granularity [24, 25]. The recent Embedding-Based Subsequence Matching hashes subsequences of the original time series to vectors based on their DTW distance to reference time series [4]. Subsequences that are identified as potentially similar to the query are then refined using the DTW. All of these techniques provide efficiency gains by sacrificing the correctness of the result as they are approximate in nature. Our approach guarantees correct results.

Correctness of the result is guaranteed in lower bounding filter-and-refine techniques [11, 26]. An efficient filter distance function that always underestimates the true DTW distance is used to prune the majority of time series. Only for the remaining candidates the exact DTW is computed. No false negatives can occur.

The Piecewise Aggregate Approximation (PAA) representation is composed of averages of consecutive values along the time axis [16]. The Adaptive Piecewise Constant Approximation adapts the number of values that are averaged to reduce the approximation error [15]. For these representations, lower-bounding distance functions for DTW have been proposed. Another lossless approach is based on four characteristic values of time series: their starting and ending value, their minimum, and their maximum [18]. LB_{Keogh} provides a lower-bound that uses the warping path constraint to compute an envelope on the warping cost for PAA segments [14]. This approach has been extended to 2-dimensional envelopes around trajectories in [28, 20]. Improved versions of the envelope technique have been proposed in [32] and [30]. The Fast search method for Time Warping (FTW) uses different resolutions to gradually refine the optimal warping path [23]. Using *early stopping*, only the warping paths with a DTW distance not exceeding some pruning distance are evaluated. This technique is also known as *early abandoning* [17].

Our technique can be integrated with these lower bounding distance functions to improve performance. We give more details in Section 4 and discuss their possible integration in Section 4.7.

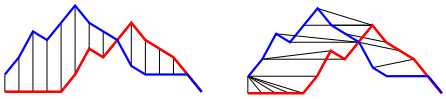


Figure 1: Euclidean Distance (left) & DTW (right)

3. DYNAMIC TIME WARPING

Time series record the value of one or more attributes as they change over a span of time. E.g., temperature measurements from several spatial locations of observation could be recorded at each full hour for each full year enabling year to year climate change analysis. Another example where the value of a number of attributes is recorded are digital videos. In this case, each point in time corresponds to a frame of the video and the values in the time series are either the raw pixel values of each frame or features that describe characteristic properties such as the distribution of colors or the overall lightness.

Formally, a time series is a sequence of (feature) vectors for consecutive points in time:

DEFINITION 1. ***Time series.***
A time series t of length n is a temporally ordered sequence $t = [t_1, ..., t_n]$ where point in time i is mapped to a d-dimensional attribute vector $t_i = (t_{i_1}, ..., t_{i_d})$ of values t_{i_j} with $j \in \{1, ..., d\}$. A time series is called univariate for $d = 1$ and multivariate for $d > 1$.

In many applications, time series are very long. For example, stock data and temperature measurements are recorded at high frequency over long periods that may span several years or decades. This observation induces that similarity measures for time series have to scale well with the length of the time series in order to be useful for these applications.

The most widely used distance functions for assessing the similarity between time series are the Euclidean distance and Dynamic Time Warping. Further examples of distance functions for time series can be found in [8].

Comparing univariate time series based on Euclidean distance is straightforward. The differences between values of corresponding points in time are squared and summed up: $Eucl(s,t) := \sqrt{\sum_{i=1}^{n}(s_i - t_i)^2}$. The definition can easily be extended to multivariate time series. The Euclidean distance ignores differences in the scale of the time axis as well as shifts along the time axis. As a consequence, two time series that exhibit a very similar pattern from the user's point of view might incur a high distance value when compared using Euclidean distance. To account for this, DTW allows for scaling and shifting of the time axis to model the similarity more appropriately.

For illustration purposes, an example for the simple univariate case with a single attribute is given in Figure 1. Two

time series are compared using the Euclidean Distance (left) and DTW (right). Horizontal lines indicate which values are matched by the respective distance functions. As Figure 1 demonstrates, the Euclidean distance computes a large distance between the two time series even though they show the same pattern shifted along the time axis. Dynamic Time Warping matches the time series such that the pattern is aligned by *warping* the time axis.

For long time series, infinite warping is typically not desirable. To avoid degenerated matchings where many values of one time series are matched to very few values of the other one, warping is usually restricted via global constraints termed *bands*. A band describes how much warping is allowed (i.e., how far apart any two aligned points can be with respect to the time axis).

DTW computes the best possible match between time series with respect to the overall warping cost under the bandwidth constraint. It is defined recursively on the length of the sequences. Formally, the definition of k-band DTW is:

DEFINITION 2. ***k-band DTW.***
The Dynamic Time Warping distance between two time series s, t of length n, m (w.l.o.g. $n \leq m$) with respect to a bandwidth k is defined as:

$$DTW([s_1, ..., s_n],\ [t_1, ..., t_m]) =$$

$$dist_{band}(s_n,\ t_m) + \min \begin{cases} DTW([s_1, ..., s_{n-1}],\ [t_1, ..., t_{m-1}]) \\ DTW([s_1, ..., s_n],\ [t_1, ..., t_{m-1}]) \\ DTW([s_1, ..., s_{n-1}],\ [t_1, ..., t_m]) \end{cases}$$

with

$$dist_{band}(s_i,\ t_j) = \begin{cases} dist(s_i, t_j) & |i - \lceil \frac{j \cdot n}{m} \rceil| \leq k \\ \infty & else \end{cases}$$

$$DTW(\emptyset, \emptyset) = 0, \quad DTW(x, \emptyset) = \infty, \quad DTW(\emptyset, y) = \infty$$

Thus, DTW is defined recursively on the minimal cost of possible matches of prefixes shorter by one element. There are three possibilities: match prefixes of both s and t, match s with the prefix of t, or match t with the prefix of s. The difference between overall prefix lengths is restricted to a band of width k in the time dimension by setting the cost of all overstretched matches to infinity. Note that the above definition corresponds to a so-called Sakoe-Chiba band where the bandwidth is fixed [21]. Another frequently used band, the Itakura band, adjusts the band such that less warping is allowed at the beginning and end [12]. An adaptive approach is presented in [19], where so-called R-K bands are learned from data. In this paper, we assume the Sakoe-Chiba band for ease of discussion, but our technique can be easily adapted to the other two band types as well.

Euclidean distance can be seen as a special case of DTW with bandwidth 0. This corresponds to no warping along the time axis, and thus reduces DTW to summing up the differences of values for corresponding points in time only.

DTW can be computed via a dynamic programming algorithm in $O(mn)$ time, where m, n are the lengths of the time series. Using a k-band, this is reduced to $O(k * \max\{m, n\})$. Instead of computing all possible alignments between the two time series, the recursive definition of DTW is used to fill a cumulative distance matrix. Each matrix entry corresponds to the best alignment between the sub-time series of the corresponding length.

This is illustrated in Figure 2. The two time series are depicted to the left (rotated) and at the top of the figure. Their optimal alignment corresponds to matching of the points in time as indicated by the black line in the matrix. For example, the horizontal segment at the lower left indicates that the first six points in time of the time series at the top are matched to the first element of the time series at the left. Assuming a bandwidth of two, the black path is invalid. The green path above it corresponds to the best alignment under this bandwidth constraint.

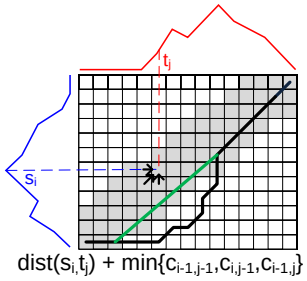


Figure 2: Warping matrix

The DTW cumulative distance matrix is filled analogously to the formula in Definition 2. More precisely, a matrix $(C = [c_{i,j}])$ for two time series $s = [s_1, ..., s_n]$ and $t = [t_1, ..., t_m]$ is filled, where each entry $c_{i,j}$ corresponds to the minimal distance between the subsequences $[s_1, ..., s_i]$ and $[t_1, ..., t_j]$.

DEFINITION 3. ***Cumulative distance matrix.***
The cumulative distance matrix $C = [c_{i,j}]$ for two time series $s = [s_1, ..., s_n]$ and $t = [t_1, ..., t_m]$ is computed recursively such that its entries are calculated as

$$\begin{aligned} c_{i,j} &= dist(s_i, t_j) + min\{c_{i-1,j-1}\ ,\ c_{i,j-1}\ ,\ c_{i-1,j}\} \\ c_{0,0} &= 0, \qquad c_{i,0} = \infty \text{ for } i \geq 1, \qquad c_{0,j} = \infty \text{ for } j \geq 1 \end{aligned}$$

Thus, the entries of the cumulative distance matrix are filled with the minimum of the three cases to be considered (cf. Fig. 2). The best alignment between the time series is recursively obtained from the alignments that are possible for the time series shorter by one. Intuitively, this means that the entry $c_{i,j}$ is computed from its three adjacent entries $c_{i-1,j-1}$, $c_{i-1,j}$, and $c_{i,j-1}$ to its bottom left.

The global warping path constraint expressed in the bandwidth parameter corresponds to setting the distance in cells that are not within the band to infinity.

DEFINITION 4. ***Matrix band.***
The valid cells in the cumulative DTW distance matrix of size $n \times m$ (w.l.o.g. $n \leq m$) with respect to a band parameter k are

$$band = \bigcup_{1 \leq j \leq m} band_j$$

with $band_j$ as the valid cells in column j:

$$band_j = \{(i, j) \mid 1 \leq i \leq n\ ,\ |\ i - \left\lceil \frac{j \cdot n}{m} \right\rceil\ | \leq k\}$$

For initialization, the non-band entries of the matrix are set to infinity. The calculation proceeds column-wise. Starting with entry $c_{1,1}$ the entries within the band of each column are calculated before continuing with the next column to the right. The DTW distance is the value of entry $c_{n,m}$ in

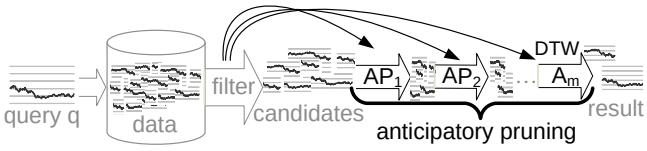


Figure 3: Anticipatory pruning in filter-and-refine algorithms

the top right corner. It corresponds to an alignment called the *warping path*, which can be visualized in the cumulative distance matrix as a sequence of adjacent entries, beginning in cell $(0, 0)$ and ending in cell (n, m).

DEFINITION 5. ***Warping path.***
For two time series $s = [s_1, ..., s_n]$ and $t = [t_1, ..., t_m]$ the warping path P is defined as a series of matrix cells

$$P = p_1, ..., p_l \text{ with } p_1 = (0, 0),\ p_l = (n, m)$$

For any two $p_i = (k, l)$, $p_{i+1} = (q, r)$ the following holds:

- *monotony:* $q - k \geq 0 \wedge r - l \geq 0$
- *continuity:* $q - k \leq 1 \wedge r - l \leq 1$
- *alignment:* $c_{q,r} = dist(s_q, t_r) + c_{k,l}$

From the cumulative distance matrix the warping path which shows the actual minimal alignment can easily be constructed as a sequence of matrix cells that correspond to a series of minima that connect the beginnings and ends of the two time series. A warping path is monotonic in the sense that the time does not go backwards. The path is continuous in that there are no omitted points in time; successive path elements correspond to consecutive points in time. The alignment property ensures that each cell p_i arose from its direct predecessor p_{i-1}.

4. ANTICIPATORY DTW

Our approach in speeding up DTW extends the classical algorithms for DTW that use a multistep filter-and-refine architecture. We call our extended filter-and-refine algorithm *Anticipatory DTW*. It is based on a number of properties (Sections 4.3 to 4.5) and makes use of our anticipatory pruning distance (Section 4.6). We show that the anticipatory pruning distance is a lower bound which implies that Anticipatory DTW guarantees no false dismissals (Theorem 3).

4.1 Filter and Refine

As illustrated in the left part of Figure 3, the basic idea in any multistep algorithm is to devise a filter function. This filter quickly retrieves a set of candidates from the entire database. Only these candidates are refined, i.e., the actual distance (here: DTW) is computed to obtain the final result.

A good filter function should be selective (i.e., the number of candidates should be small). It should also be efficiently computable (i.e., its runtime should be significantly smaller than the DTW runtime itself). And finally, lower bounding filters that always underestimate the true DTW distance ensure that the filter is complete in that there are no false dismissals, which in turn implies that the final result after the refinement step is the same as if the complete database had been queried using only DTW [3, 11, 26].

In multistep filter-and-refine algorithms (e.g., GEMINI or KNOP [11, 26]), the filter distance is compared to some pruning threshold max. In range queries, this max value is the range itself. If the distance with respect to a lower bound exceeds the range, then the exact distance (here: DTW) does so as well. Hence, lossless pruning is possible. In k-nearest neighbor queries, the max threshold corresponds to the maximum distance of the current k nearest neighbor candidate set. If the lower bound exceeds this value, then the exact distance exceeds it, too. The pruning in existing filter-and-refine approaches is based on a single comparison of this filter distance and the max value. If the filter distance is below max, DTW is computed. This approach can be considered to be without memory in the sense that all information computed in the filter step is ignored later on.

4.2 Overview Over Our Proposed Method

We propose a technique that is orthogonal to existing DTW speed-up techniques based on filter-and-refine frameworks. It plugs into these frameworks as illustrated in the right part of Figure 3. Instead of directly computing the complete (and thus costly) DTW refinement after the filter step, our anticipatory pruning distance (AP) is incrementally computed. If it at any time exceeds the given pruning threshold max, the time series can be ruled out as part of the result set – safely discarding many computation steps. If no pruning is possible in intermediate steps, the last step produces the exact DTW distance. By using distance information from the filter step for our AP distance in the refinement step, we are able to provide a lower bounding estimate of the still to be calculated DTW steps. Thus, we go beyond previous approaches known as early stopping[23] or early abandoning[17], which also partially avoid the computation of DTW distances but do not anticipate future computation steps. As our technique shows, this greatly reduces the number of computations in the refinement step.

Our anticipatory pruning distance is based on three properties: First, DTW computation is *incremental*, i.e., the entries in the warping matrix increase with the sequence length. Second, many existing lower bounding filters for the DTW distance can be characterized as *piecewise*, i.e., a subsequence of the filter computation is a valid filter for the subsequences themselves. And finally, DTW is *reversible*, i.e., computing the warping path from the beginning to the end is the same as doing so from the end to the beginning.

The Anticipatory DTW algorithm uses these three properties by calculating piecewise filter distances on reversed time series in the filter step of a filter-and-refine framework. During the incremental computation of the DTW, the information from the piecewise filter is a lower bounding anticipatory estimate of the DTW parts yet to be calculated. The exploitation of this additional information allows for stopping the DTW calculation sooner than would have been possible without the anticipatory part. If no stopping is possible, the last step of computing AP corresponds to the last step of computing the DTW.

Our approach is highly flexible in that it can be combined with many existing lower bounding filters. It is easy to integrate with these approaches, as all the information required for anticipatory pruning is easily derived from existing filter calculations. Moreover, the pruning capability of Anticipatory DTW comes at hardly any additional cost, as the filter distances have been computed in the filter step already.

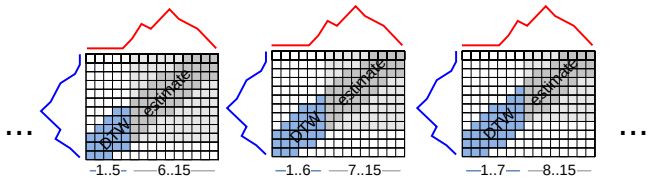


Figure 4: Overview of our technique

4.3 Incremental Computation of DTW

Anticipatory pruning uses filter information to perform additional pruning checks in the refinement. These checks are based on our new anticipatory estimate and the fact that DTW is *incremental* (i.e., the column minima of the cumulative distance matrix increase with the length of the time series).

THEOREM 1. ***DTW is incremental.***
For any cumulative distance matrix $C = [c_{i,j}]$ as defined in Definition 3, the column minima are monotonically non-decreasing

$$\min_{i=1,...,n}\{c_{i,x}\} \leq \min_{i=1,...,n}\{c_{i,y}\} \quad for\ x < y.$$

PROOF. The proof is straightforward from the definition of the cumulative distance matrix:
$c_{i,j} = dist(s_i, t_j) + min\{c_{i-1,j-1}\ ,\ c_{i,j-1}\ ,\ c_{i-1,j}\}$. Any element $c_{i,y}$ in a column y is based on at least one value from the same or a preceding column plus a non-negative distance between the two time series entries s_i and t_y. If the value is from the preceding column, the theorem holds. If the value is from the same column, it is based on the entry immediately below (i.e., $c_{i-1,y}$). For this entry, the same argument holds. There is a finite number of steps in which the entry can be based on an entry in the same column. As soon as the boundary of the k band is reached, the choice of minimum has to be based on the preceding column. Thus, the theorem holds. □

Consequently, many DTW computations do not have to be fully processed, but can be interrupted once the column minimum exceeds the *max* value. A similar idea, called *early stopping*, in the FTW approach is used to avoid refinement of approximate warping paths at different levels of granularity [23]. In our anticipatory pruning technique, we use Theorem 1 for exact DTW computations, but go beyond column minima as we add pieces of filter information for a much closer estimate of the DTW.

Entries $c_{i,j}$ in the cumulative distance matrix correspond to the cost of possible warping paths between subsequences $[s_1, ..., s_i]$ and $[t_1, ..., t_j]$. For the remaining alignment and its cost, we do not yet have the entries in the cumulative distance matrix. We do, however, have a filter distance for the warping path of the complete time series. Provided that we can decompose this filter into one that serves as a lower bound on the remaining warping path, we may combine the filter components with the column minimum for an overall estimate which we call anticipatory pruning distance.

4.4 Piecewise Filter Computation for DTW

Anticipatory pruning can be understood as an incremental refinement of the filter with DTW distance information. As illustrated in Figure 4, the cumulative distance matrix for DTW computation is filled as usual. For example, columns 1 to 5 have been computed in the leftmost figure. As seen above, the column minimum in the fifth column is a lower bound for the DTW distance between the two time series depicted to the left and at the top of the cumulative distance matrix. We do know, however, that only partial time series have been accounted for, namely up to the fifth entry in the time series at the top, and up to the seventh entry (5 plus band of width 2) in the time series to the left. For the remaining subsequences of the time series, starting in the sixth column, we derive an estimate from the preceding filter step. This is illustrated in the darker area at the top right for columns 6 to 15. The estimate is a piece of filter information corresponding to these last columns (the anticipatory part).

This estimate on partial time series alignments requires that the filter be *piecewise*, i.e. decomposable into a series of lower bounds for all subsequences of increasing lengths that start at the beginning of the time series.

DEFINITION 6. ***Piecewise DTW lower bound.***
A piecewise lower bounding filter for the DTW distance is a set $f = \{f_0, ..., f_m\}$ with the following property:

$$j = 0: \quad f_j(s,t) = 0$$
$$\forall j > 0: \quad f_j(s,t) \leq \min_{(i,j)\in band_j} DTW([s_1, ..., s_i], [t_1, ..., t_j])$$

Intuitively speaking, a piecewise lower bounding filter can be decomposed into a series of lower bounds for all possible partial DTW warping paths that start at the beginning of both time series and end in a respective column. The piecewise property is not a major constraint, as most existing lower bounding filter for DTW can be decomposed in such a manner. We will prove this property for some of the most widely used approaches in Section 4.7.

4.5 Reversible Computation of DTW

The piecewise property of a DTW lower bound and the incremental computation property of DTW together do not suffice to be able to combine pieces of the filter with partial DTW computation for a close lower bound of DTW. To derive an overall lower bound, we require a combination of a partial DTW computations up to some column j of the cumulative distance matrix and piecewise filter information for the remaining columns $(j+1), ..., m$. However, the piecewise filter only provides information for ranges of columns starting with the first column. Fortunately, DTW computation is *reversible*, i.e., computing the distance for the reversed time series (from the end to the beginning) yields exactly the same result as computing it for the original time series.

THEOREM 2. ***DTW is reversible.***
For any two time series $[s_1, ..., s_n]$ and $[t_1, ..., t_m]$ their DTW distance is the same as for the reversed time series:

$$DTW([s_1, ..., s_n],[t_1, ..., t_m]) = DTW([s_n, ..., s_1],[t_m, ..., t_1])$$

PROOF. Assume that this does not hold and to the contrary (w.l.o.g.):
$DTW([s_1, ..., s_n], [t_1, ..., t_m]) < DTW([s_n, ..., s_1], [t_m, ..., t_1])$.
Let $p_1, ..., p_l$ and $p_1^{\leftarrow}, ..., p_l^{\leftarrow}$ be warping paths with minimal cost for the non-reversed and for the reversed time series. Since reversing path $p_1, ..., p_l$ also yields a valid warping path for the reversed time series (properties monotony, continuity, and alignment of Definition 5 follow from the according

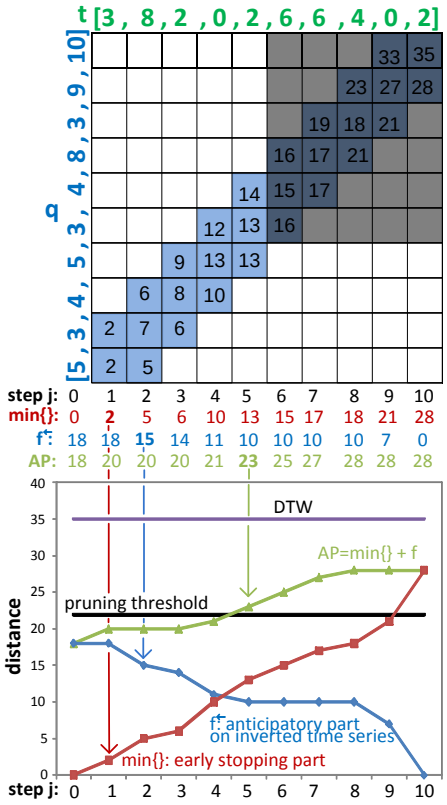


Figure 5: Anticipatory pruning example

properties of the non-reversed path) and since its cost equals $DTW([s_1, ..., s_n], [t_1, ..., t_m])$, it follows that $p_1^{\leftarrow}, ..., p_l^{\leftarrow}$ cannot be a warping path with minimal cost. As this contradicts the assumption, DTW is reversible. □

Reversibility is important for our anticipatory pruning, as it allows deriving a lossless estimate of the DTW from any piecewise lower bounding filter. By reversing the order of time series during filter computation, we obtain the same filter distance. However, we may now use the pieces of the filter computation from the end of the time series backwards up to the current point of DTW computation as an estimate of the remaining path cost. As this estimate is lower bounding, and so are column minima, anticipatory pruning is lower bounding as well, as we will prove formally in Theorem 3.

4.6 Anticipatory Pruning Distance

Anticipatory DTW includes a sequence of pruning checks. After every column of the cumulative distance matrix that has been computed, piecewise filter information is added to yield an overall lower bound for the exact DTW distance. It can therefore be seen as a series of m filters for time series s, t of length n, m, where the final step provides the actual DTW distance. Formally, our Anticipatory Pruning Distance is defined as:

DEFINITION 7. ***Anticipatory Pruning Distance.*** *Given two time series s and t of length n and m, a cumulative distance matrix $C = [c_{i,j}]$ as defined in Definition 3, a piecewise lower bounding filter f for the reversed time series $s^{\leftarrow}$ and $t^{\leftarrow}$ as defined in Definition 6, and a value $j \in \{1, ..., m\}$, the j^{th} step of the Anticipatory Pruning Distance for the k-band DTW distance between s and t is defined as*

$$AP_j(s,t) \quad := \quad \min_{i=1,...,n} \{c_{i,j}\} + f_{m-j}(s^{\leftarrow}, t^{\leftarrow}).$$

The definition of anticipatory pruning distance is thus a set of DTW column minima with added pieces of filter information for the remainder of the time series. During each incremental computation step j, the column minimum of the partial path between $[s_1, ..., s_i]$ and $[t_1, ..., t_j]$ according to DTW is combined with the anticipatory information for the rest of the alignment available from the lower bound from the filter step. The resulting estimate of the entire path in step j is AP_j.

Let us illustrate anticipatory pruning in the small example depicted in Figure 5. The upper part of the Figure shows two univariate time series: a query q, a database element t, and the corresponding cumulative distance matrix. For reasons of simplicity, this example uses an L_1-based variant of the LB_{Keogh} lower bound (which is piecewise as discussed later on). The relationship between the DTW distance (35), the pruning threshold (22), and the components of AP_j are depicted in the lower part of Figure 5.

The filter distance between $q^{\leftarrow}$ and $t^{\leftarrow}$ yields the following piecewise results: $f(q^{\leftarrow}, t^{\leftarrow}) = (0, 7, ..., 18)$. Thus, the filter step distance is 18. To make the computation of our anticipatory pruning distance more intuitive to the reader, we reversed the piecewise results in the figure: $f_j^{\leftarrow}(s^{\leftarrow}, t^{\leftarrow}) := f_{m-j}(s^{\leftarrow}, t^{\leftarrow})$. Assume that our current pruning threshold max is 22. Hence, the filter step distance of 18 does not exceed max, and t cannot be pruned by the filter. As illustrated at the top of Figure 5, we start filling the cumulative distance matrix for $DTW(q, t)$ and compute the minimum of column j (denoted by $min\{\}$) in step j. For example, the first column minimum is 2, which accounts for the mappings of the 1-prefix of t to those prefixes of q permitted by $band_1$. As can be seen in the figure, the minima of the first few columns are only loose bounds. For anticipatory pruning, we add the first filter entry $f_1^{\leftarrow}$, which lower bounds all possible reversed alignments starting in (10,10) and ending in column 2. We obtain "$min+f$" $= 2 + 18 = 20$ as the first anticipatory pruning distance, which is much closer to the DTW distance than the column minimum alone. Yet it still is less than the pruning threshold max. Continuing with filling the matrix, the second column minimum is 5, and the filter entry $f_2^{\leftarrow}$ is 15, yielding 20 again. We proceed until step 5, where the column minimum is 13. Combined with the corresponding filter distance of 10, the sum is 23, which exceeds the pruning threshold max. Hence, we may immediately stop the DTW computation, skip the remaining five columns, and discard t. In this example, early stopping without our anticipatory component takes 5 more steps before the time series can be pruned and thus our technique saves considerable computational cost.

We now prove that the anticipatory pruning distance is indeed a lower bound of the DTW and therefore guarantees lossless query processing:

THEOREM 3. ***AP lower bounds DTW.*** *The anticipatory pruning distance AP_j as defined in Definition 7 lower bounds the k-band DTW distance between two time series s and t of length n and m.*

$$AP_j(s,t) \leq DTW(s,t) \; \forall \; j \in \{1, ..., m\}$$

PROOF. Anticipatory pruning is essentially a series of partial DTW paths combined with a lower bound estimate of the remainder taken from the previous filter step. We show that it lower bounds DTW by arguing that **(1)** for all possible steps $j, 1 \leq j \leq m$, the column minimum lower bounds the true path. It holds **(2)** that DTW is reversible, i.e., we can replace a path between two time series by the path between reversed time series. Finally, in **(3)** we show that we can combine the first two properties to derive anticipatory pruning as a lower bound of DTW.

(1) The cost of any DTW warping path ending in column j of the cumulative distance matrix $C = [c_{i,j}]$ is obviously lower bounded by the minimum cost of all these paths $\min_{i=1,...,n}\{c_{i,j}\}$.

(2) As Theorem 2 states, DTW is reversible. Thus, we can make use of the fact that $DTW([s_1, ..., s_n],\ [t_1, ..., t_m]) = DTW([s_n, ..., s_1],\ [t_m, ..., t_1])$ in (3).

(3) We now prove that anticipatory pruning, as the sum of a column minimum and a lower bound of any valid path that starts in the subsequent column (or reverse path that ends in this column), lower bounds DTW.

$$AP_j(s,t) \quad := \quad \min_{i=1,...,n}\{c_{i,j}\} + f_{m-j}(s^{\leftarrow}, t^{\leftarrow}) \leq DTW(s,t)$$

Due to the continuity of DTW, the minimal path (see Definition 5) passes through all columns, and thus also through both column j and column $j+1$. The optimal path $P = p_1, ..., p_l$ for $DTW(s,t)$ can be decomposed into two parts: the first part from the beginning to column j: $p_1, ..., p_u$ and one which continues in column $j+1$ to the end: $p_{u+1}, ..., p_l$. (If more than one of the p_k is in column j, let p_u denote the last one). Let the positions $p_u := (x, j)$ and $p_{u+1} := (y, j+1)$. Then, we can rewrite the right side of the inequation as $DTW(s,t) = c_{p_u} + DTW([s_y, ..., s_n], [t_{j+1}, ..., t_m])$. From (1), we have that $\min_{i=1,...,n}\{c_{i,j}\} \leq c_{p_u}$. From (2), we have that $DTW([s_y, ..., s_n], [t_{j+1}, ..., t_m]) = DTW([s_n, ..., s_y], [t_m, ..., t_{j+1}])$, which is underestimated by $f_{m-j}(s^{\leftarrow}, t^{\leftarrow})$ by definition of f. Thus, we indeed have that anticipatory pruning lower bounds the DTW. □

Now that we have shown that anticipatory pruning is a lower bound of the actual DTW, it immediately follows that query processing with anticipatory pruning is lossless in multistep filter-and-refine algorithms. More details and proofs that lower bounding guarantees completeness can be found in [11, 26]. Moreover, we can easily see from the definition of anticipatory pruning that eventually the series of computations results in the exact DTW distance between the two time series. Our approach incurs very little overhead, as we only maintain piecewise information on the filter distances.

Our approach can also easily be generalized to multiple filter steps. As long as the filters are both lower bounding and piecewise, anticipatory pruning can be applied to any step in the filter chain. As the filter is computed for increasing time series length, an estimate on the remainder of the time series can be obtained from the previous filter step. To do so, merely the order in which the filter steps process the time series has to be alternated between steps.

Anticipatory pruning is computed as outlined in the pseudo code in Algorithm 1. Given two time series q, t, a pruning threshold max, and a vector $f = (f_0(q^{\leftarrow}, t^{\leftarrow}), ..., f_m(q^{\leftarrow}, t^{\leftarrow}))$ that represents the distances of the piecewise lower bounding filters, we start by initializing a column vector *col*. Then, in each incremental step j, the next column of the exact DTW computation is calculated by *CalcDTWMatrixColumnBand* and the new anticipatory pruning distance $AP_j(q,t)$ is computed as APj. The value APj is used for pruning iff it exceeds max. Otherwise, the next pruning value AP_{j+1} is computed. If no pruning is possible, the exact $DTW(q,t)$ is returned unless it itself exceeds max.

Algorithm 1 Anticipatory pruning

```
Procedure AP (Time series q, t, Real max, Vector f)
1: Vector col[q.length +1] = [0, ∞, ..., ∞] ▷ first index is 0
2: for j = 1 to t.length do
3:     col = CalcDTWMatrixColumnBand(q, t, col, j)
4:     Real APj = min(col[1],...,col[q.length])+f[t.length-j]
5:     if APj > max then
6:         return ∞                                  ▷ pruning
7: if col[t.length] > max then
8:     return ∞                                      ▷ pruning
9: return col[t.length]              ▷ no pruning, return DTW
```

In the next section, we study different existing lower bounding filter techniques for DTW. We show that they are indeed piecewise and thus can be used for our Anticipatory DTW.

4.7 Piecewise Lower Bounding Filters

In this section, we describe state-of-the-art methods that provide lower bounding filter distances for DTW. While the methods themselves have been introduced elsewhere, our contribution lies in showing that all of these different methods fulfill the requirement of being piecewise as defined in Definition 6. From this, we have with Theorem 3 that anticipatory pruning of DTW with these methods is lossless.

The lower bounds presented in Sections 4.7.1 and 4.7.2 are only capable of comparing sequences of the same length n, while FTW in Section 4.7.3 can cope with different lengths.

4.7.1 Linearization

The basic idea for linearization of the DTW computation for efficient and exact indexing is based on the idea of computing an *envelope* of upper and lower values U and L around the query time series q with respect to the k-band:

$$\begin{aligned} U_i &= max_{i-k \leq j \leq i+k}\{q_j\} \\ L_i &= min_{i-k \leq j \leq i+k}\{q_j\}. \end{aligned}$$

The squared Euclidean distance between values above or below the envelope of the other time series t lower bound the exact k-DTW distance [14].

$LB_{Keogh}(q,t) = \sqrt{\sum_{i=1}^{n} MinDist(t_i, L_i, U_i)}$ with

$$MinDist(t_i, L_i, U_i) = \begin{cases} (t_i - U_i)^2 & t_i > U_i \\ (t_i - L_i)^2 & t_i < L_i \\ 0 & else \end{cases}$$

An extension providing a tighter envelope when PAA dimensionality reduction is applied is given in [32] as:

$\overline{U}_i = \frac{N}{n}(u_{\frac{n}{N}(i-1)+1} + ... + u_{\frac{n}{N}(i)})$ and

$\overline{L}_i = \frac{N}{n}(l_{\frac{n}{N}(i-1)+1} + ... + l_{\frac{n}{N}(i)})$.

As we can see from the definition of LB_{Keogh}, this approach is a dimension-wise summation of distances. Thus, it constitues a piecewise lower bound of DTW.

THEOREM 4. *LB_{Keogh} is a piecewise lower bounding filter as defined in Definition 6.*

PROOF. Decomposition into a set of filters f_j for increasing sequence length is straightforward:

$$f_j = \sqrt{\sum_{i=1}^{j} MinDist(t_i, L_i, U_i)}.$$

That is, the summation up to the current subsequence length j. For both U, L and $\overline{U}, \overline{L}$ [14, 32] prove that the lower bounding property holds. □

In our experiments, we evaluate anticipatory pruning for LB_{Keogh} with the tighter envelope $\overline{U}, \overline{L}$.

4.7.2 Corner Boundaries

A different type of lower bound can be obtained as piecewise corner-like shapes in the warping matrix through which every warping path has to pass [30]. Formally, for a time series q, its corner shapes are:

$$C_i^L(q) = \begin{cases} \min_{\max(1,i-k)\le j\le i}\{q_j\} & i \le \lceil \frac{n}{2} \rceil \\ \min_{i\le j\le \min(n,i+k)}\{q_j\} & i > \lceil \frac{n}{2} \rceil \end{cases},$$

$$C_i^U(q) = \begin{cases} \max_{\max(1,i-k)\le j\le i}\{q_j\} & i \le \lceil \frac{n}{2} \rceil \\ \max_{i\le j\le \min(n,i+k)}\{q_j\} & i > \lceil \frac{n}{2} \rceil \end{cases},$$

$$C_i(q) = (C_i^L(q), C_i^U(q)).$$

These approximations are constructed for both time series and enclose the positions in the time series that correspond to the corner shapes in the warping matrix. A corner shape i in the warping matrix for two time series t and q is a combination of the two envelope positions $C_i(q)$ and $C_i(t)$. In [30], experiments showed that a hybrid approach of corner shapes at the beginning and the end of the warping matrix and straight line shapes in the middle leads to an even tighter lower bound called LB_{Hybrid}. The idea of straight line shapes is very similar to the LB_{Keogh} approach, thus the hybrid corner approach is defined as

$$LB_{Hybrid}^2(q,t) = \sum_{i=1}^{n} \begin{cases} MinDist(t_i, L_i, U_i) & k+2 \le i \le n-k-1 \\ \min\{CD_i(q,t), CD_i(t,q)\} & else \end{cases}$$

with the Corner Distance CD_i that calculates the minimal distance for a warping path that passes through the corner shape i:

$$CD_i(q,t) = \begin{cases} (q_i - C_i^U(t))^2 & q_i > C_i^U(t) \\ (q_i - C_i^L(t))^2 & q_i < C_i^L(t) \\ 0 & else \end{cases}$$

THEOREM 5. *LB_{Hybrid} is a piecewise lower bounding filter as defined in Definition 6.*

PROOF. Since LB_{Hybrid} uses the linearization of LB_{Keogh} in the first case, this part of the proof follows from Theorem 4. In the *corner* case, we immediately obtain the pieces f_j from the definition of the corner shapes. □

In our experiments, we demonstrate the usefulness of anticipatory pruning for corner boundaries in terms of its pruning capability.

4.7.3 Path Approximation

Another approach for speeding up DTW is taken in the FTW (Fast search method for Dynamic Time Warping) technique [23]. The idea is to generate approximations of the optimal warping. It works for DTW with and without band constraints by checking continuously for any approximate segment of the time series whether the *max* value of the current candidates has been exceeded. If this is not the case, a finer approximation is generated. The efficiency gains are based on the fact that coarse DTW computations are computationally less expensive.

At differing levels of granularity, segments are approximated using minimum and maximum values U, L. The Lower Bounding distance measure with Segmentation (LBS) is exactly the DTW on the approximated segments. For time series segments of different lengths a normalization factor is used. The distance is refined locally (i.e., each segment is only refined while below the *max* value).

Since LBS is the DTW distance on segments, it also constitutes a piecewise lower bound:

THEOREM 6. *FTW is a piecewise lower bounding filter as defined in Definition 6.*

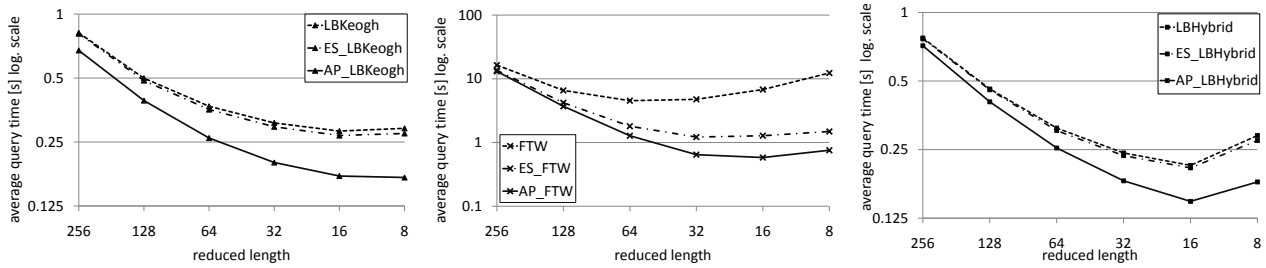
PROOF. We obtain a series of piecewise filters as $f_j = DTW([Q_1, ..., Q_J], [T_1, ..., T_{J+K}])$ where Q and T are approximations of time series q and t. J denotes the smallest value such that the approximated segment $[Q_1, ..., Q_J]$ fully contains all points of the time series up to index j. Similarly, $J+K$ denotes the smallest value such that the approximated segment $[T_1, ..., T_{J+K}]$ fully contains all points of the time series up to index $j + k$. While the exact value of J and $J+K$ may vary for individual time series, at least the coarsest level of approximation is computed for the entire time series, yielding valid pieces for the entire length. As the computation is in the same cumulative matrix procedure as for DTW, we obtain a piecewise filter via column minima as desired. For the lower bounding property see [23]. □

We study the performance of anticipatory pruning for FTW in our experimental section.

4.7.4 Dimensionality Reduction and Indexing

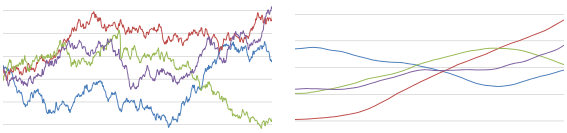
Dimensionality reduction is a very useful technique for efficiency gains in time series similarity search as many time series are very long, i.e., high dimensional. Several approaches have been suggested, such as the Piecewise Aggregate Approximation (PAA) used, e.g., in the LB_{Keogh} approach. The idea is to replace parts of the original time series by constant approximations. All segments, either of fixed or of adaptive length (e.g., Adaptive Piecewise Constant Approximation (APCA) [15]), are piecewise by their very nature. Consequently, they fulfill the requirements for anticipatory pruning and can be used with Anticipatory DTW as well.

A number of DTW speed-up techniques also use indexing structures (e.g., R-trees in [14] or sequential structures in [23]). As anticipatory pruning is orthogonal to such techniques, the efficiency benefit is maintained in our approach. Using anticipatory pruning requires merely a change in the computation of the refinement step, which is independent of any underlying indexing approach.



(a) LBKeogh (b) FTW (c) LBHybrid

Figure 7: Absolute efficiency improvement (average query time) for different reductions on RW2



(a) RW1 (b) RW2

Figure 6: Sample of synthetic time series

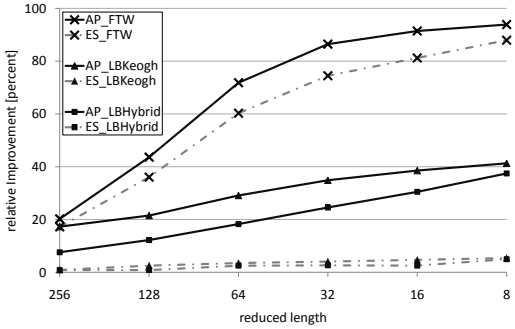


Figure 8: Relative improvement (average query time) on RW2

5. EXPERIMENTS

All experiments were executed on 2.33GHz Intel XEON CPU cores running JAVA implementations. The following default parameters were used where not stated otherwise: width of DTW band $k = 40$, length of the time series $n = 512$, number of nearest neighbors retrieved 10 (per query). The query workload was 200. Several synthetic data sets for scalability studies and real world data sets for different parameters were used. For scalability in the number of attributes per point in time, we generated two multivariate random walk data sets (univariate examples are shown in Figure 6). Both contain time series of length 512 and are of cardinality $10,000$. The number of attributes d was varied between 1 and 50.

RW1: The non-normalized value of the j^{th} component ($j \in \{1, ..., d\}$) of t_{i+1} is a random value drawn from a normal distribution with parameters $\mu = 0$, $\sigma^2 = 1$ added to the value of the j^{th} component of t_i: $t_{(i+1)_j} = t_{i_j} + N(0, 1)$. RW1 was normalized to an average value of 0.

RW2: The first two elements of the time series are generated as in RW1. For the remaining points in time of RW2, the average value μ depends on the last increase/decrease: $t_{(i+1)_j} = t_{i_j} + N(t_{i_j} - t_{(i-1)_j}, 1)$. As early values of RW2 have a low variance while the later values have a high variance, RW2 was normalized to an average value of 0.

In addition to these synthetic time series, we use three multivariate real world data sets.

SignLanguage: This multivariate data set is derived from the 11 real valued attributes of the sign language finger tracking data from [13]. For the efficiency experiments, the time series were created from the concatenated raw data by extracting non-overlapping windows of equal length. The length n of the non-overlapping time series was varied between 64 and 512 and the band k was varied between 10 and 150. The number of time series was fixed at $1,400$.

Video: We use two video data sets. The first one is the **TRECVid** benchmark data [27]. The second dataset, **NEWSVid**, consists of TV news we recorded at 30 fps. 20-dimensional HSV histograms were computed for each video frame (i.e., the time series are multivariate with $d = 20$). The length n of the non-overlapping video sequences was varied between 64 and 2048 frames. The cardinality of the database depends on the length of the time series and varies between 650 and $2,000$ for the TRECVid data and between $2,000$ and $8,000$ for the TV news.

In the experiments, we thoroughly investigate the runtime improvements of anticipatory pruning over the three piecewise base techniques Linearization (LB_{Keogh}), Corner boundaries (LB_{Hybrid}), and Path Approximation (FTW).

5.1 Dimensionality Reduction

We start by evaluating the efficiency of anticipatory pruning with respect to dimensionality reduction on the RW2 data set. Figure 7 shows the average runtime for the three lower bounding filters on a logarithmic scale. Anticipatory pruning, denoted as AP, yields substantial runtime improvements compared with the base methods LB_{Keogh}, FTW, and LB_{Hybrid}. It also outperforms the early stopping approach (ES) [23] for all three methods. Anticipatory pruning is especially helpful for strong reductions where it makes up for the loss in information of the filter step by pruning during DTW refinement. Figure 8 summarizes the gains by showing the relative improvement for the same experiment. To abstract from implementation issues, Figure 9 demonstrates that the runtime gains are due to a reduction in the number of required calculations.

For the second synthetic data set, a summary of relative improvement with respect to the required number of calculations is given in Figure 10. We observe similar gains in

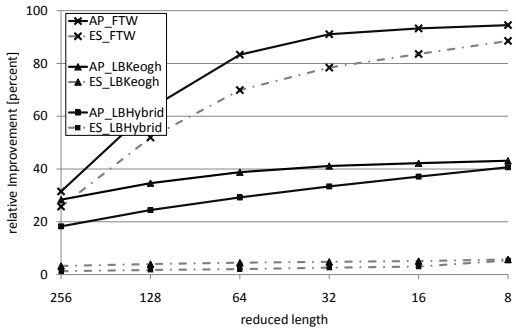


Figure 9: Efficiency improvement (#calc.) for varying reductions on RW2

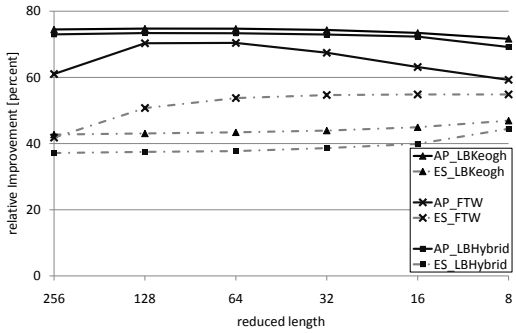


Figure 10: Efficiency improvement (#calc.) for varying reductions on RW1

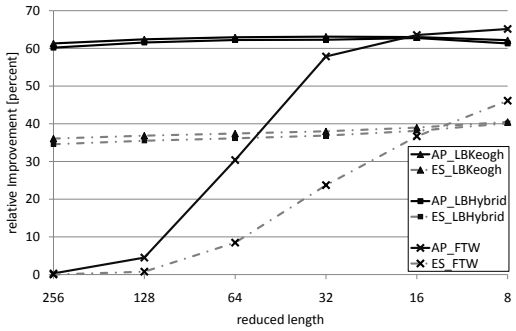


Figure 11: Efficiency improvement (#calc.) for different reductions on NEWSVid

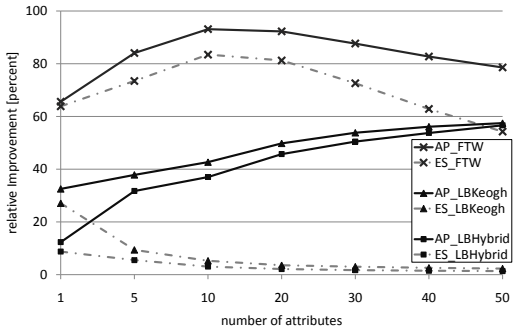


Figure 12: Efficiency improvement (#calc.) for varying numbers of attributes on RW2

performance, where the difference for the LB_{Hybrid} method is most pronounced and shows a reduction from around 75% (AP) to 40% (for early stopping).

On the real world NEWSVid data set, the behavior is similar for LB_{Keogh} and LB_{Hybrid} (see Figure 11). For FTW, however, we observe relatively little improvement for low reductions, but rapid improvement for strong reductions.

5.2 Univariate and Multivariate Time Series

Our next set of experiments evaluates the effect of the number of attributes on the performance of anticipatory pruning. Based on the results of the preceding section, a reduction to 16 dimensions was chosen. As depicted in Figure 12 for the RW2 data set, the general tendency is the same as for the dimensionality reduction. The performance gain of anticipatory pruning even increases with the number of attributes. While anticipatory pruning avoids some DTW computations in the univariate case, this effect is much more pronounced for multivariate time series.

Figure 13 shows the same experiment for the RW1 data set. For this dataset, univariate time series benefit considerably from anticipatory pruning and the behavior is fairly consistent over the evaluated range of attributes.

5.3 Bandwidth

In Figure 14, we study the influence of the bandwidth constraint on the NEWSVid dataset. Dimensionality reduction was again set to 16 dimensions. A remarkable reduction in the number of calculations can be observed for all three lower bounding filters. There is a slight decrease with respect to the bandwidth, but even for extremely wide bands of 150, anticipatory pruning yields substantial improvements. This effect is also present on the TRECVid data set (Figure 15) and the SignLanguage data set (Figure 16).

5.4 Length of Time Series

Our next study empirically validates the scalability of anticipatory pruning with respect to the length of time series on three multivariate time series data sets. Because of the varying length, no fixed dimensionality reduction was possible; thus, reduction was carried out with a segment length of 4 (e.g., a sequence of length 256 was reduced to 64 dimensions). Figure 17 shows performance gains for all three lower bounding filters on the NEWSVid data set with remarkable improvements for LB_{Keogh} and LB_{Hybrid}. These gains scale very well with the length of the time series.

As shown in Figure 18, the results are similar for the smaller SignLanguage data set with sequences of length up to 512. Even though this data set shows a jump in the pruning power of FTW for length 128, the performance gains of anticipatory pruning are robust.

On the third real world data set, TRECVid, anticipatory pruning shows similar pruning power and scalability (see Figure 19).

5.5 Number of Nearest Neighbors

In our last experiment, we evaluate the effect of varying the number of nearest neighbor retrieved during query processing. This parameter has only very limited effect on the pruning capability of our technique as demonstrated in Figure 20 by the great performance gains of anticipatory pruning for the NEWSVid data set.

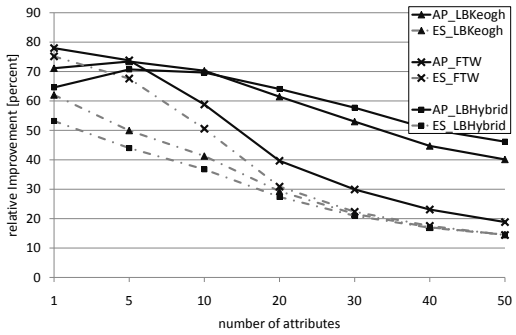


Figure 13: Efficiency improvement (#calc.) for varying numbers of attributes on RW1

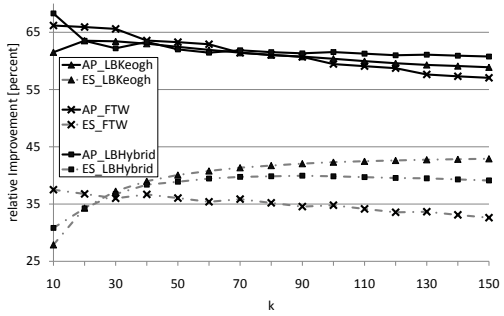


Figure 14: Efficiency improvement (#calc.) for varying DTW bandwidths on NEWSVid

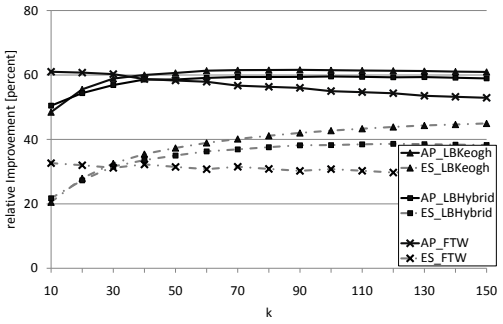


Figure 15: Efficiency improvement (#calc.) for varying DTW bandwidths on TRECVid

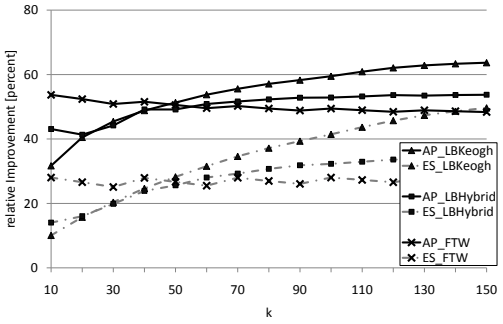


Figure 16: Efficiency improvement (#calc.) for varying DTW bandwidths on SignLanguage

6. CONCLUSION & OUTLOOK

In this work we speed up DTW (Dynamic Time Warping), a widely used distance function for time series similarity search. Our novel anticipatory pruning makes best use of a family of speed-up techniques based on multistep filter-and-refine architectures. By computing an estimated overall DTW distance from already available filter information, a series of lower bounds of the DTW is derived that requires hardly any overhead. Our experimental evaluation demonstrates a substantial reduction in the number of calculations and consequently a significantly reduced runtime. Our technique can be flexibly combined with existing and future DTW lower bounds. Based on the reduction of calculations, in future work we plan on exploring the combination of the DTW with complex ground distances such as the Earth Mover's Distance for multivariate time series.

7. ACKNOWLEDGMENTS

This work was partially funded by DFG grant SE 1039/1-3. The authors would like to thank the anonymous reviewers for their very helpful feedback that led to a revised presentation of this work.

8. REFERENCES

[1] J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[2] I. Assent and H. Kremer. Robust adaptable video copy detection. In *SSTD*. Springer, 2009.

[3] I. Assent, A. Wenning, and T. Seidl. Approximation techniques for indexing the Earth Mover's Distance in multimedia databases. In *ICDE*, page 11, 2006.

[4] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos. Approximate embedding-based subsequence matching of time series. In *SIGMOD*, pages 365–378, 2008.

[5] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on KDD*, pages 229–248, 1994.

[6] F. K.-P. Chan, A. W.-C. Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. *TKDE*, 15(3):686–705, 2003.

[7] A.-P. Chen, S.-F. Lin, and Y.-C. Cheng. Time registration of two image sequences by dynamic time warping. In *Proc. ICNSC*, pages 418–423, 2004.

[8] L. Chen and R. Ng. On the marriage of Lp-norms and Edit Distance. In *VLDB*, pages 792–803, 2004.

[9] S. Chu, E. J. Keogh, D. Hart, and M. J. Pazzani. Iterative deepening dynamic time warping for time series. In *SDM*, 2002.

[10] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. In *VLDB*, 2008.

[11] C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.

[12] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 23(1):67–72, 1975.

[13] M. W. Kadous. Australian sign language data set 1,

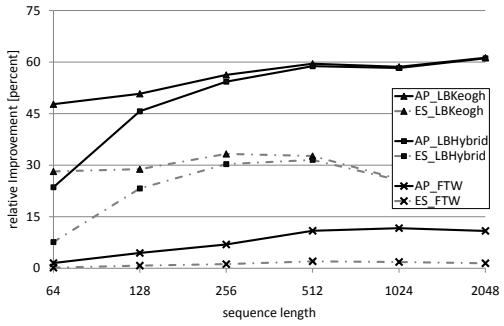


Figure 17: Efficiency improvement (#calc.) for varying time series lengths on NEWSVid

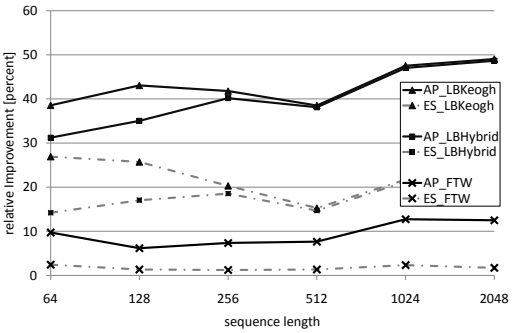


Figure 19: Efficiency improvement (#calc.) for varying time series lengths on TRECVid

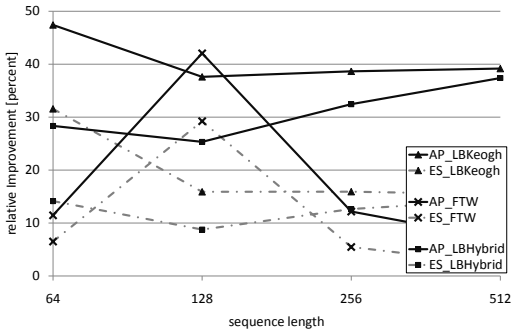


Figure 18: Efficiency improvement (#calc.) for varying time series lengths on SignLanguage

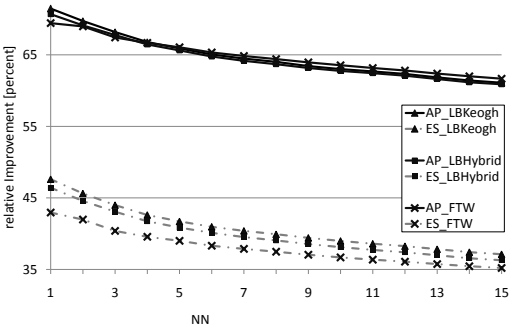


Figure 20: Efficiency improvement (#calc.) for varying numbers of nearest neighbors on NEWSVid

1999. `www.cse.unsw.edu.au/~waleed/tml/data/`.

[14] E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[15] E. J. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD Rec.*, 30(2):151–162, 2000.

[16] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *KAIS*, 3(3):263–286, 2001.

[17] E. J. Keogh, L. Wei, X. Xi, S. Lee, and M. Vlachos. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *VLDB*, pages 882–893, 2006.

[18] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, 2001.

[19] C. A. Ratanamahatana and E. J. Keogh. Making time-series classification more accurate using learned constraints. In *SDM*, 2004.

[20] T. M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series, 2003.

[21] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 26(1):43–49, 1978.

[22] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *ICDE*, pages 1046–1055, 2007.

[23] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: fast similarity search under the time warping distance. In *PODS*, pages 326–337, 2005.

[24] S. Salvador and P. Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *KDD TDM*, 2004.

[25] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[26] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD*, pages 154–165. ACM, 1998.

[27] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *Proc. MIR*, 2006.

[28] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. J. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *SIGKDD*, pages 216–225, 2003.

[29] B. K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[30] M. Zhou and M. H. Wong. Boundary-based lower-bound functions for dynamic time warping and their indexing. In *ICDE*, pages 1307–1311, 2007.

[31] M. Zhou and M. H. Wong. Efficient online subsequence searching in data streams under dynamic time warping distance. In *ICDE*, pages 686–695, 2008.

[32] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD*, 2003.