Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation

Diego F. Silva* Gustavo E. A. P. A. Batista*

Abstract

Dynamic Time Warping (DTW) is certainly the most relevant distance for time series analysis. However, its quadratic time complexity may hamper its use, mainly in the analysis of large time series data. All the recent advances in speeding up the exact DTW calculation are confined to similarity search. However, there is a significant number of important algorithms including clustering and classification that require the pairwise distance matrix for all time series objects. The only techniques available to deal with this issue are constraint bands and DTW approximations. In this paper, we propose the first exact approach for speeding up the all-pairwise DTW matrix calculation. Our method is exact and may be applied in conjunction with constraint bands. We demonstrate that our algorithm reduces the runtime in approximately 50% on average and up to one order of magnitude in some datasets.

1 Introduction

Dynamic Time Warping (DTW) is certainly the most relevant distance for time series analysis. Such relevance has been evidenced by a large body of experimental research showing that, for instance, the 1-nearest neighbor DTW (1-NN-DTW) algorithm frequently outperforms more sophisticated methods on a large set of benchmark datasets [12].

The main issue with DTW is its computational complexity. A straightforward implementation of DTW is quadratic in time and space. Although a simple trick can make DTW linear in space[1], the time complexity is a more difficult matter.

An important observation is that all the recent advances in speed-up DTW calculations are confined to similarity search. However, there is a significant number of data mining algorithms, including clustering and classification, that require the pairwise distance matrix for all time series objects.

In the particular case of time series clustering, the authors of [13] show that the calculation of the all-pairwise distance matrix for DTW completely dominates the runtime of well-known clustering algorithms. For instance, for a large dataset, the computation of the all-pairwise distance matrix would take approximately 127 days using off-the-shelf desktop computers and a naïve quadratic DTW algorithm. On the same computer, the calculation of a hierarchical clustering given an already computed pairwise distance matrix would take only 4 seconds.

In case a researcher or practitioner is interested in applying DTW with algorithms that require the all-pairwise distance matrix, the only speed-up techniques available are constraint bands (also known as warping windows) [6, 3] or DTW approximations [8, 10].

In this paper, we propose a novel approach for speeding up the all-pairwise DTW matrix calculation. Our method uses an upper bound estimation to prune unpromising warping alignments. In other words, our method prunes DTW partial paths that will lead to unfruitful warping paths by comparing the current calculated value with a distance upper bound.

2 Background

Euclidean distance (ED) is the most established distance measure between time series. The ED measures the dissimilarity between time series comparing the observations at the exact same time. For this reason, the ED can be very sensitive to distortions in the time axis. Many applications require a more flexible observation matching, in which an observation of the time series x_i at time i can be associated to an observation of the time series y_j at time $j \neq i$.

The DTW distance achieves an optimal nonlinear alignment of the observations under boundary, monotonicity and continuity constraints. DTW is usually calculated using a dynamic programming algorithm. Equation 2.1 describes the initial condition of the algorithm[2].

$$dtw(i,j) = \begin{cases} \infty, \text{ if } i = 0 \text{ or } j = 0 \\ 0, \text{ if } i = j = 0 \end{cases} \tag{2.1}$$

Equation 2.2 presents the recurrence relation of DTW algorithm.

*Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Brazil – {diegofsilva, gbatista}@icmc.usp.br

[1]When the only output of interest is the final distance and the warping path can be disregarded.

[2]Further in this document, we will assume that the time series objects may have different lengths. Therefore, $x = x_1, x_2, \dots, x_N$ and $y = y_1, y_2, \dots, y_M$.

$$(2.2)\quad dtw(i,j) = c(x_i, y_j) + min\begin{cases} dtw(i-1,j) \\ dtw(i,j-1) \\ dtw(i-1,j-1) \end{cases}$$

where $i = 1 \ldots N$ and $j = 1 \ldots M$. $c(x_i, y_j)$ is the cost of matching two observations x_i and y_j, usually calculated with squared Euclidean distance.

The resulting value in $dtw(N, M)$ is the DTW distance between x and y. Thus, the algorithm iteratively fills an array with the lowest accumulated cost for all alignments to each pair of observations to be matched. Figure 1 shows an example of the optimal non-linear alignment found by this algorithm and how it is represented in the DTW calculation matrix.

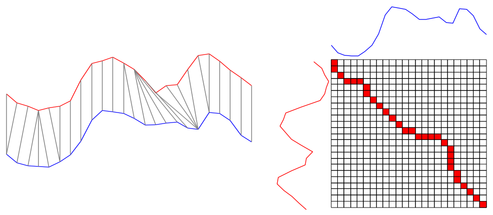


Figure 1: Optimal non-linear alignment (*left*) and the matrix obtained by the dynamic programming algorithm, highlighting the optimal alignment (*right*)

In order to improve the efficiency of DTW calculations, the use of warping windows is common [6, 3]. Warping window, or constraint band, defines the maximum allowed time difference between two matched observations. From the algorithm standpoint, this technique restricts the values that need to be computed to a smaller area around the main diagonal of the matrix.

However, the exact window size that would provide the best results for a dataset is data dependent. Outside classification problems with 1-NN, there are no clear guidelines to set this parameter and possibly the best approach is to evaluate the results for several window sizes. We will return to this topic in the Section 6.2.

3 On the Need of the All-Pairwise Distance Matrix

Although the community has mainly focused on the speed-up of DTW calculation for similarity search, other algorithms, for instance in clustering and classification require the all-pairwise DTW matrix.

Recently, the community has devoted some effort to improve the efficiency of time series clustering algorithms. For instance, Zhu et al. have framed the problem of calculating the all-pairwise DTW matrix as an anytime algorithm and applied such approach to clustering [13] and Ulanova et al. have proposed new techniques to cluster temporal objects in admissible time [11].

The need of the all-pairwise distance matrix is easily seen in the clustering task. Many clustering algorithms do not require the objects as input, but their relations. In the case of time series mining, the relations can be defined as the distance between each pair of instances. For example, this is the case of the most methods of the family of hierarchical clustering and the well-known k-medoids.

The need of all-pairwise distance matrix is not restricted to clustering algorithms. A recent advance in time series classification proposes the use of machine learning classifiers with distances representing values of attributes [4]. Being n the number of training examples, this approach constructs an $n \times n$ attribute-value table in which the values of attributes are defined by the distance of one object to all other objects in the training set. Therefore, in order to construct a classification model, this approach needs the whole distance matrix between the training examples. For classifying a new example, it needs the distance between the query time series and all the training examples.

4 Related Work

Due to the relevance of DTW in time series mining and its high computational cost, the scientific community has proposed several approaches to deal with DTW in tasks involving large amounts of data.

4.1 Similarity Search Although similarity search is not the focus of this work, we briefly review the intuition of these approaches to clarify the reasons they are not applicable to all-pairwise distance calculations.

Suppose that we are interested in finding the k most similar objects to a query time series and a variable *best-so-far* stores the true DTW distance to the k-th nearest object known up to a certain moment of the algorithm execution. Consider a lower bound function (LB) that returns a value that is certainly lower or equal the true DTW between two objects. Clearly, if the LB between a given training object (o_k) and a query time series (q) is higher than the *best-so-far*, we know that o_k is not one of the k nearest neighbors of q. Therefore, such an object can be discarded. It is obviously only applicable when we have no interest in those objects whose distance are greater than the *best-so-far*.

The main challenge on speeding up the *exact* computation of the all-pairwise matrix is that we cannot rely on techniques that avoid computing the DTW distance for certain pairs of objects. *The only resource is to improve the internal efficiency of the DTW computation.*

4.2 DTW Approximations Distance functions that approximate DTW are a popular approach to speed up DTW calculations. Since these methods only require two time series objects as input, they are directly applicable to calculating the all-pairwise distance matrix.

Several approaches were proposed in the literature with this purpose. Some examples are FastDTW [8] and Lucky Time Warping [10]. Approximations may also be used to fill the distance matrix in anytime fashion [13].

The main drawback with these approaches is that they do not provide any guarantees in terms of approximation error to the true DTW. In other words, the user has no means of setting an allowable maximum error in reference to the true DTW.

4.3 Biological Sequences Alignment The problem of time series alignment is similar to the alignment of biological sequences, such as proteins and RNA. Usually, biological sequences are also compared with costly similarity functions.

In order to improve the running time of sequence alignment algorithms, Carrillo and Lipman [1] proposed to expand the function calculations only to partial alignments that present promising values. First, the approach calculates a lower bound to the similarity function and use this value as the threshold to decide which cells should be expanded in the DTW matrix. Given a change in any partial alignment, the cells affected by the changed value are calculated only if the current value is higher than the threshold defined by the lower bounding function. Although simple, this procedure guarantees the calculation of the exact similarity value.

This strategy was never used directly to speed-up DTW calculations. However, a similar idea is used as an intermediate step of a similarity search algorithm named FTW, described next.

4.4 FTW Fast search method for dynamic Time Warping (FTW) [7] is the method that shares more similarities with our proposal. The authors propose a similarity search method based on a recursive refinement of a lower bound calculation.

In a coarse representation, the method finds a lower bound to DTW, using a dynamic programming algorithm. If the computed lower bound is smaller than the *best-so-far*, the method proceeds to a finer representation. Otherwise, the calculation is aborted, since the object is not one of the nearest neighbors.

In each level of approximation, the method prunes the DTW matrix values that are guaranteed to be greater than the *best-so-far*. Such an algorithm, called EarlyStopping, is the most similar method to our proposal. However, our method does not rely on a *best-so-far*, which would restrict it to similarity search. Instead, we use an upper bound, which can be initially set as ED and refined as the algorithm proceeds in the calculation of the DTW distance.

In summary, although FTW and our proposal share some similarities, FTW is a similarity search algorithm and relies on a best-so-far. So, it is not directly applicable to calculate the all-pairwise DTW matrix. Our method prunes warping paths that are greater than an upper bound in the actual DTW calculation and is not dependent of a best-so-far. Finally, FTW increasingly refines time series representations while our method is computed in the full resolution only;

5 DTW with Pruned Warping Paths

In this paper, we propose the DTW with Pruned Warping Paths (PrunedDTW). We adapted the traditional DTW algorithm to recognize and prune cells in the DTW matrix that are guaranteed to not lead to alignments that will result in the optimal path.

We emphasize that our method is similar or considerably faster than DTW and always returns the optimal path between two time series objects. In addition, PrunedDTW can be implemented in linear space. The additional variables necessary for pruning are $O(1)$ and for updating the upper bound is $O(max(N, M))$ in space. Finally, PrunedDTW supports warping windows and time series of different lengths.

Even more importantly, PrunedDTW is orthogonal to all of the proposals of speeding DTW up that we are aware of. Therefore, PrunedDTW can be used in conjunction with the literature to further increase the efficiency of DTW in different tasks, including similarity search.

5.1 The Intuition Behind our Proposal Our method is motivated by a very simple observation: frequently, the cells of a DTW matrix vary in a large range of values. Usually, the values around the optimal alignment in the matrix are relatively low. In contrast, even cells moderately distant from the optimal path frequently receive much higher values. Figure 2 shows an example of DTW matrix for two time series from the Mallat dataset. There are several regions in the matrix with accumulated costs that go far beyond the optimal DTW distance (which is 71.77 in this case).

A cell at a position (i, j) of the DTW matrix represents the cost of the optimal alignment that starts at the position $(1, 1)$ and ends at (i, j). For an internal position in the DTW matrix, i.e., $i < N, j < M$, if the cell at (i, j) is part of the optimal path then the optimal path cost has the cost at (i, j) plus the cost of matching observations from (i, j) up to (N, M). As the cost of

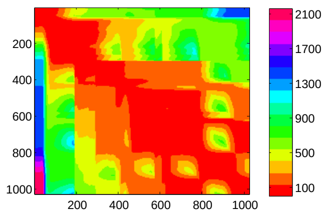


Figure 2: DTW matrix between two time series. The colors indicate the value obtained in each cell of the matrix

matching two observations is zero or positive value, the warping paths are monotonically increasing.

Intuitively, if a cell at (i, j) has a high value, it clearly indicates that such a warping path is very unlikely to lead to the optimal path. However, as we want to propose an exact method, we need to establish a threshold in which we can guarantee that such a partial warping path will not be part of the optimal path. More importantly, we need a *pruning strategy* that explores the cells with large values and the DTW recurrence relation of Equation 2.2 to decide when we can start and stop evaluating the cells. We detail the pruning strategy and the corresponding algorithm in the next section.

5.2 Pruning Strategy

In order to define which values are high enough to be pruned, we can use an upper bound (UB) of the DTW distance. In this paper, we use the squared Euclidean distance (sqED) as UB[3]. This distance measure is a special case of DTW when the optimal alignment corresponds to the main diagonal of the DTW matrix.

Our algorithm calculates the UB as the first step. We note, however, that it represents a small overhead since the UB calculation is linear in time complexity. After estimating the UB, we must specify a criterion so that we can prune the calculation of DTW matrix cells, ensuring not to discard any element that may belong to the optimal alignment. For this purpose, we use different criteria for pruning the beginning and the end of each row[4] of the matrix. Algorithm 1 details the proposed method.

Line 17 represents the pruning technique (not presented so far) that modifies the regular DTW algorithm. The pruning technique manipulates the values of the auxiliary variables to determine the beginning of the calculation of each row (line 11 of the algorithm) and its end (which depends on the assignments made in the lines 14 and 19). Algorithm 2 describes the pruning criteria in details.

Algorithm 1 PrunedDTW algorithm

```
Input: Time series $x$, with length $N$
    Time series $y$, with length $M$
    Warping window size $ws$
    Upper bound $UB$ of the DTW between $x$ and $y$
Output: The distance between $x$ and $y$ according DTW
    {Auxiliary variable to prune lower triangular}
 1: $sc \leftarrow 1$
    {Auxiliary variable to prune upper triangular}
 2: $ec \leftarrow 1$
    {Initialize the matrix of DTW calculations }
 3: for $i = 1$ to $N$ do
 4:     $D[i, 0] \leftarrow \infty$
 5: end for
 6: for $i = 1$ to $M$ do
 7:     $D[0, i] \leftarrow \infty$
 8: end for
 9: $D[0, 0] \leftarrow 0$
10: for $i = 1$ to $N$ do
11:     $beg \leftarrow max(sc, i - ws)$
12:     $end \leftarrow min(i + ws, M)$
13:     $smaller\_found \leftarrow FALSE$
14:     $ec\_next \leftarrow i$
15:     for $j = beg$ to $end$ do
16:         $D[i, j] = sqED(x_i, y_j)$
                $+ min(D[i-1, j-1], D[i-1, j], D[i, j-1])$
17:         Pruning strategy {copy & paste Algorithm 2 here}
18:     end for
19:     $ec \leftarrow ec\_next$
20: end for
21: return $D[N, M]$
```

Algorithm 2 Pruning criteria implementation

```
 1: if $D[i, j] > UB$ then
 2:     if $smaller\_found = FALSE$ then
 3:         $sc \leftarrow j + 1$
 4:     end if
 5:     if $j \geq ec$ then
 6:         break {break the for loop / jump to the next row}
 7:     end if
 8: else
 9:     $smaller\_found \leftarrow TRUE$
10:     $ec\_next \leftarrow j + 1$
11: end if
```

The main idea of the pruning strategy is that the values related to a row i define which column can be pruned in the row $i + 1$. In particular, the variables sc (start column) and ec (end column) control the range of columns that needs to be analyzed in the next row.

There are two separate strategies, one for pruning cells in the lower triangular matrix and the other one for the upper triangular. For the lower triangular, the pruning is controlled by the variable sc. The idea is that, traversing a row from left to right, as long as we find columns with values greater than UB, it is safe to say that the same columns on the next row will also have values greater than UB.

Figure 3 illustrates this idea. For the row 4, the first two columns have a value greater than UB. Therefore, the variable sc is set to column 3 (Algorithm 2, line

[3]Our approach also works for time series of different lengths in which the sqED is not defined. We will return to this topic in Section 5.3.

[4]Our implementation traverses the matrix in a row-major order. However, the algorithm can also be implemented by traversing the matrix in a column-major order.

3) and the processing can safely start at column 3 in the next row. We can prune the computation of the variables A and B because of the DTW recurrence relation represented by the three arrows. The value of the cell at (i, j) is the cost of matching the observations x_i and y_j added to the minimum of the values in three other cells of the matrix (Algorithm 1, line 16). As the column 0 is initialized with infinity values, the variable A will obligatorily have a value greater than UB. The same occurs to B which depends on $A > UB$ and other two cells in the previous row also greater than UB. In contrast, variable C may have a value smaller than UB since it depends on $D(4, 3) \leq UB$.

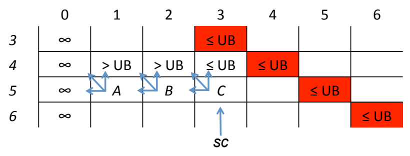


Figure 3: Pruning in the lower triangular matrix

The initial value of the variable sc is 1, i.e., while no values greater than UB are found, the calculation in each row will start at the first column. In the case that a warping window is used, the rows will be initiated in the column with the highest index between the column established by the warping window and the one related to the pruning criteria (Algorithm 1, line 11).

Notice that the main diagonal cells (in red) are marked as smaller than or equal to UB. This occurs because, as mentioned before, all warping paths are monotonically increasing and the main diagonal corresponds to ED. This ensures that for a row i, $sc \leq i$ and our pruning is confined to the lower triangular matrix. This is an important observation, given that we will later update the UB as we fill the DTW matrix.

The second pruning strategy is responsible for pruning the upper triangular matrix. This strategy defines the column where we can stop the calculation of the next row. The variable ec stores the column where the first of a sequence of values greater than UB starts and goes until the end of the current row.

Figure 4 illustrates the idea. In this example, row 1 is processed and ec is set to 4. The variable ec is marking the first value of a sequence of values greater than UB that finishes at the end of the row (Algorithm 2, line 10). We can stop the row 2 as soon as two criteria are met: (i) the calculated value is greater than UB and (ii) the current column index is greater or equal to ec. Suppose that the cell A is greater than UB. In this case, criterion (ii) is not met. We can see that cell B may be smaller than UB since it can use $D(1, 3)$ which is lower or equal to UB. However, if B is greater than UB then both criteria are met and we can stop processing line 2. This occurs because variables C and D can only inherit values from the matrix that are greater than UB.

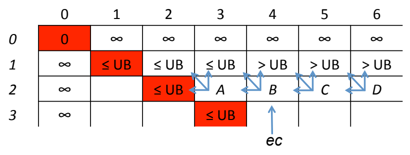


Figure 4: Pruning in the upper triangular matrix

There is a subtle detail in the algorithm that allow us to initialize $ec = 1$, instead of $ec = M$. This occurs because of the initialization of the DTW matrix with infinite values. Therefore, $ec = 1$ actually marks the first column in row 0 that is greater than UB.

5.3 Iteratively Updating the Upper Bound

An interesting point about the DTW algorithm is that the matrix D stores the costs of the optimal paths from $(1, 1)$ to (i, j). This means we can use such partial optimal matching to update the UB value. In the case where sqED is adopted as UB, every time we compute a main diagonal cell $D(i, i)$ we are in position to update UB.

For this purpose, we also need the partial values of the UB calculation. Since the calculation of sqED does not depend on the order in which we calculate the distance between each pair of observations, we calculate this measure in reverse order, i.e., from the end to the beginning of the time series. At each step of the calculation, we store the partial value obtained in a vector $sqED_{partials}$, according to Equation 5.3. Notice that the calculation of $sqED_{partials}$ is still $O(N)$ when computed in the reverse order.

$$sqED_{partials}(i) = \sum_{j=i}^{N} (x_j - y_j)^2 \tag{5.3}$$

Once we computed $D(i, i)$, we have all the necessary information to update the UB, according to Equation 5.4.

$$UB = D(i, i) + sqED_{partials}(i + 1) \tag{5.4}$$

This equation updates the UB with the optimal DTW alignment of the first i observations summed to the squared Euclidean distance between the observations ahead of i. Thus, the value of UB becomes increasingly tighter in relation to the optimal DTW value, as we proceed in the computation of the matrix. Notice that $D(i, i)$ is always smaller or equal to the Euclidean distance of the first i observations of the time series. Therefore, the pruning power of our method increases at each iteration.

At first glance, the use of Euclidean distance as UB restricts our proposal to the comparison of time series of the same length. Such restriction occurs because ED is only defined for objects of the same length. However, we can easily circumvent such restriction. Let N be the length of the shorter time series. A UB can be obtained by calculating the squared Euclidean distance between the first N observations summed to the distance between the remaining values of the longer time series to the last observation of the shorter time series. There are other possible UB since any warping path is a UB for DTW. We will further explore this fact in the next section.

Our implementation has some additional features such as a linear space complexity. We do not describe such features in details here because they are not the main contributions of this paper. We have built a website in which we made available all detailed numerical results, source code and supplemental material not included in this paper [9]. However, we note that our paper is completely self-contained. In addition, our website provides a proof of the correctness of our algorithm.

5.4 Other UB Approaches So far we have only considered the sqED as UB. However, any measure that is an upper bound of DTW can be used, as far as the cost of its calculation does not compromise the overall cost of the algorithm.

The use of other UB approaches has two immediate consequences for our work. The first one is that we can use the true DTW distance as UB. Even if such an approach is not practical in real situations, it provides us the best-case analysis in which the algorithm would prune the highest possible number of cells. Figure 5 shows an example of this fact. Note how the pruning in is much more aggressive by using the actual DTW as UB. The use of a tight UB may even prune cells in the main diagonal of the matrix.

The second consequence will be better explored in Section 6.4. In most of experimental analyses, we need to perform a search for the best warping window size. The naïve approach for this is simply run the DTW algorithm multiple times, calculating the all-pairwise matrix for each warping window size. However, we can speed-up this approach by using the optimal DTW distance calculated for a smaller warping window as UB for the next (larger) window size to be assessed.

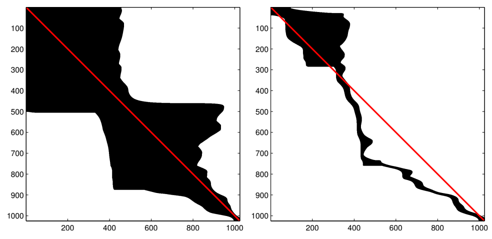


Figure 5: Regions of the DTW matrix pruned by our proposed criteria (in white) by using the sqED (left) and the true DTW (right) upper bounds for the same pair of time series. The red lines show the positions of the main diagonal cells

6 Experimental Evaluation

As we are proposing an exact method, the main way of assessing our results is the running time of the algorithm for the computation of the all-pairwise distance matrix.

We were *extremely* careful to measure the runtimes of the algorithms in order to provide meaningful experimental results. We used *identical* DTW implementations with and without the pruning method. Therefore, the difference in time between them can be only explained by the proposal in this paper.

We ran all the experiments on the same computer[5]. At any time, there was only one process computing DTW distances. In order to reduce the variance caused by other processes outside our control[6], we executed each method three times and reported the average running times.

We notice that we are very committed to the reproducibility of our results. For this purpose, the web page for this paper [9] contains all source code and detailed results obtained in our experiments.

6.1 Benchmark Datasets In order to assess the efficiency of PrunedDTW, we performed an experimental evaluation using 10 freely-available benchmark datasets. Specifically, all datasets are from the UCR Time Series Classification/Clustering Page [2].

We chose data sets with time series with at least 500 observations. Although our method can be used with time series of any length, the calculation of all-pairwise distance matrix of short time series can be done relatively fast with the traditional algorithm.

6.2 On the Warping Window Length Our experimental analysis is highly dependent on the warping window length. For this reason, before we introduce

[5]The experiments were carried out in a desktop computer with 12 Intel(R) Core(TM) i7 $-$ 3930K CPU @ 3.20GHz and 64Gb of memory running Debian GNU/Linux 7.3.

[6]Such as process running operating system tasks.

our results, we will discuss the relevance of this parameter. Empirical evidence points to the fact that small window sizes provide superior 1-NN classification accuracy [5]. On the other hand, there are no conclusive studies about this parameter for different algorithms or mining tasks, including clustering.

The assumption that small windows are the most suitable for time series matching is commonly accepted in the literature. However, there are few exceptions. An example is the classification method proposed by [4]. In their empirical evaluation, the addition of DTW features calculated with no warping window improved the results considerably in comparison to the classifier that just uses features with constrained DTW. Actually, such improvement allowed their method to obtain a statistically significant difference over 1-NN-DTW.

We performed a quick experiment to evaluate whether the assumption of small warping windows also applies to some well-known clustering algorithms. We evaluated the performance of the hierarchical clustering with complete linkage and k-medoids algorithm using eight different values for warping window length: 5%, 10%, 15%, 20%, 30%, 40%, 50%, and 100% of the time series length. For each of the 10 datasets evaluated in our experiment, we measured the rand index and the silhouette.

Figure 6 summarizes the results. This graph presents the count of cases in which the best result for a given clustering algorithm and an evaluation measure was obtained for each value of warping window length. In case of a tie, all tied windows are counted. The distribution of the best results among the different lengths supports the recommendation to researchers and practitioners to look for several values of window length for new data sets, including the DTW with no warping window.

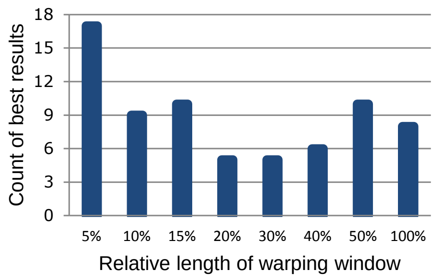


Figure 6: Count of cases in which the best result was obtained by each evaluated warping window length

6.3 Runtime for All-Pairwise DTW Matrices

Given the importance of warping windows in both accuracy and execution times, we performed our experiments varying such parameter. We evaluated the time to calculate the distances matrices using windows with relative length of 10%, 20%, 30%, 40%, 50%.

Figure 7 graphically shows the results. We present the results of the three methods: DTW stands for the standard DTW algorithm, PrunedDTW is our proposal using sqED as UB and OracleDTW is our proposal with the true DTW as UB. Although OracleDTW cannot be used in practice, its results are optimal in the sense that they represent the highest possible number of pruned cells that our method could achieve with a (imaginary) perfect constant time UB. So, OracleDTW provides a reference of the best performance that could be obtained by PrunedDTW.

PrunedDTW outperformed DTW in most of the cases. This indicates that the overhead of the pruning tests (which increases the constant of the computational complexity) are usually compensated by the number of pruned cells. In some cases in which the warping window is small (usually 10%), PrunedDTW could not achieve a significant speed-up. This is expected since warping windows significantly reduce the number of cells that need to be computed, leaving little space for improvements. As the warping windows are increased, PrunedDTW obtains more significant speed-ups. For a 50%-band size, the speed-ups of PrunedDTW relative to the computation time of DTW are in the range of 12.01% to 89.61%.

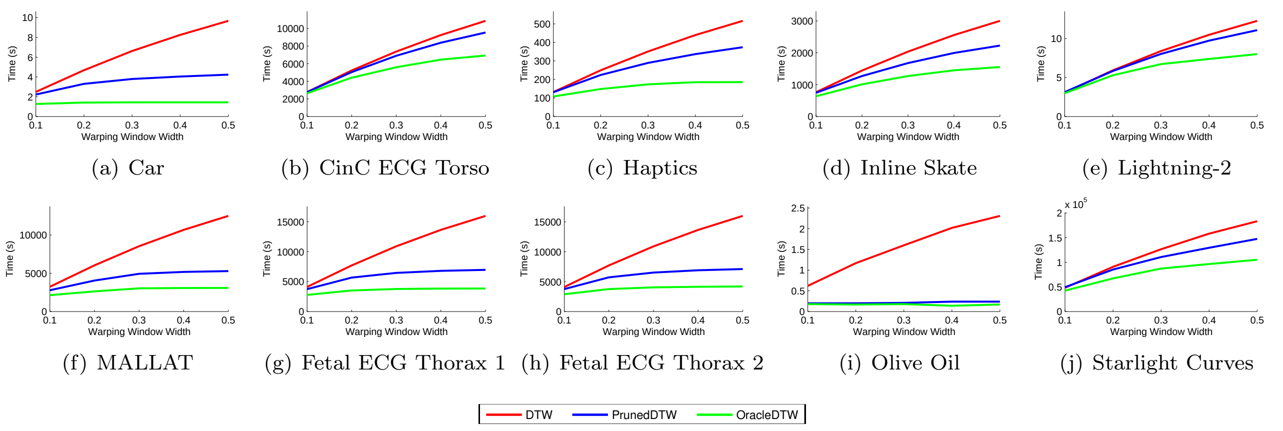
The gain obtained by PrunedDTW over DTW is commonly larger than the gain of OracleDTW over PrunedDTW, indicating that ED was able to obtain most of the achievable speed-up for these datasets. However, the cases in which it is not true to reveal the need of research for other upper-bound methods.

We note that we have performed additional experiments with DTW approximations as UB, whose results are not described in this paper. The ED presented the best compromise between tightness to DTW and time complexity among the evaluated functions.

6.4 Accumulative Runtime for All-Pairwise DTW Matrices

The results in Figure 7 represent the scenario in which the user knows which warping window size is the best for its application domain. A more realistic experiment would involve computing the running times for the calculation of pairwise distance matrices for several warping window sizes, simulating a search for an optimal value to this parameter.

As we showed in the Section 6.2, the search for the optimal value of warping window length is required for a good performance of some time series mining algorithms. For this reason, we believe that it is important for any DTW speed-up method to support



(a) Car (b) CinC ECG Torso (c) Haptics (d) Inline Skate (e) Lightning-2

(f) MALLAT (g) Fetal ECG Thorax 1 (h) Fetal ECG Thorax 2 (i) Olive Oil (j) Starlight Curves

Figure 7: Time (in seconds) to calculate the all-pairwise DTW distances with different warping window sizes

warping windows so that the user can observe the impact of such parameter over the data at hand.

In the case of the interest in calculating the all-pairwise DTW matrix with the conventional algorithm, the trivial approach is to run the algorithm multiple times. In that case, no information is reused. This occurs because the newly computed cells are very likely to influence the already computed cells. In that case, most of the matrix cells would have to be recomputed.

Regarding PrunedDTW, we can simply use the distance computed for a smaller window size as UB for a larger window size. A distance computed for a smaller warping window is always smaller or equal to the sqED and, therefore, can be a more effective UB than sqED. For the first run, i.e., for the smallest window, there is no previous DTW distance to be used as UB. In this case, we can naturally adopt the sqED as UB.

Figure 8 presents the results. There is a significant difference between these results and the ones presented in Figure 7. In Figure 8 the running times for a given warping window size r means the *accumulative* time necessary to calculate the matrices for all warping windows smaller and equal to r. In this setting, PrunedDTW is even more effective, obtaining very similar speed-ups to the ones obtained with OracleDTW.

6.5 Comparison with FastDTW As we are not aware of other approaches to speed-up the exact all-pairwise DTW computation, it is natural to find difficulties to compare to the state-of-the-art. One possibility is the comparison to approximate DTW methods. This is not an entirely fair comparison, since the approximate methods will trade approximation accuracy by speed when PrunedDTW has zero approximation error.

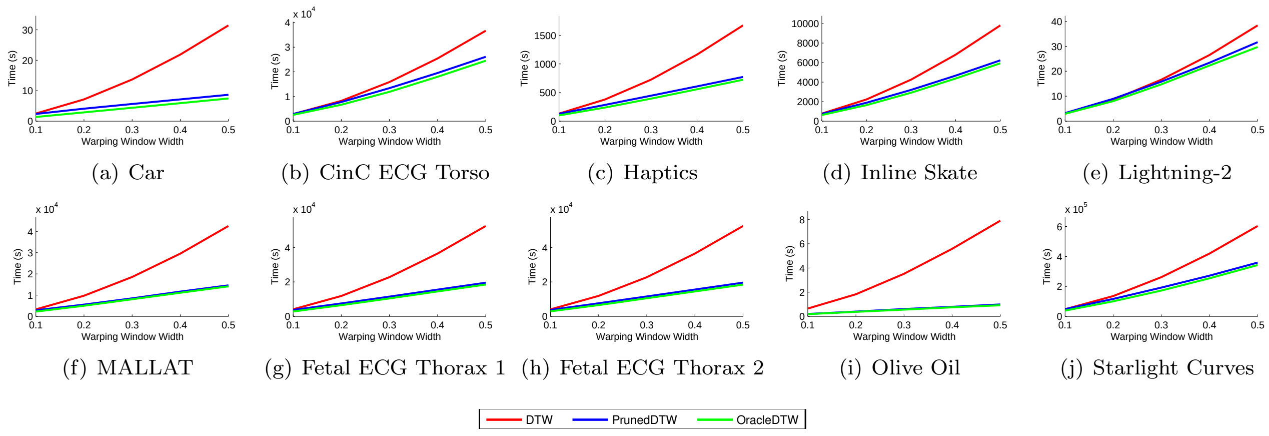
We chose to compare our approach to FastDTW, since it is the most well-known approximation of DTW in the literature. However, we believe that our method is superior to FastDTW, for the following reasons: our method is exact, FastDTW is approximated; our method supports warping windows, FastFTW only works with unconstrained matrices; our method has no parameters, FastDTW has a parameter r (radius) that influences its accuracy and running time; and our method is simple to understand and implement, FastDTW is a sophisticated method with intricate details such as increasing data resolutions.

Since FastDTW does not support Sakoe-Chiba band, we evaluated the time to calculate the all-pairwise matrix for the unconstrained DTW. Table 1 presents the results in percentage of the time spent by conventional DTW algorithm.

Table 1: Percentual runtime of FastDTW and PrunedDTW relative to the conventional DTW algorithm

Dataset	FastDTW	PrunedDTW
Car	40.53%	39.68%
CinC ECG Torso	26.52%	80.02%
Haptics	28.01%	63.99%
InlineSkate	23.29%	63.96%
Lightning-2	43.85%	79.05%
MALLAT	27.63%	31.74%
Non-Invasive Fetal ECG Thorax1	37.53%	32.98%
Non-Invasive Fetal ECG Thorax2	37.26%	33.71%
Olive Oil	38.21%	10.96%
Starlight Curves	30.52%	61.95%

Both methods are more efficient than the conventional DTW for all datasets. Although we noticed it is an unfair comparison, PrunedDTW achieved better or similar results (differences under 7%) than FastDTW in half of the evaluated datasets. Besides the fact that FastDTW achieved better runtime results in the other 5 datasets, they are the datasets in which results of PrunedDTW are far from OracleDTW, reinforcing the need of new upper bound approaches.



(a) Car (b) CinC ECG Torso (c) Haptics (d) Inline Skate (e) Lightning-2

(f) MALLAT (g) Fetal ECG Thorax 1 (h) Fetal ECG Thorax 2 (i) Olive Oil (j) Starlight Curves

Figure 8: Accumulative time (in seconds) to calculate the all-pairwise DTW distances with different warping window sizes. The distance calculated for a smaller window size is used as UB for the next larger window size

7 Conclusion and Future Work

In this paper, we proposed a novel method to speed-up the calculation of Dynamic Time Warping between time series. Differently from the previous work in literature, our method does not rely on a *best-so-far* or distance approximations. So, our proposal is adequate for any application that requires the distance between every pair of objects. The results show that our method is faster than the conventional DTW algorithm, especially in cases in which we want to calculate distances for different warping windows values or we want to use large constraint bands or unconstrained DTW.

As for the future work, we intend to explore the application of our proposal in different time series mining tasks and also in multidimensional time series scenarios. We also intend to evaluate the impact of different upper bounds. A possible direction to find new upper bound functions is to look for adaptations of bounding functions used in other applications domains, such as the computational biology.

Acknowledgements

The authors would like to thank FAPESP by the grants #2013/26151-5 and #2015/07628-0 and CNPq by the grants #303083/2013-1 and #446330/2014-0.

References

[1] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

[2] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UCR time series classification archive, Datasets available before July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[3] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

[4] R. J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, pages 1–30, 2015.

[5] C. A. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *SDM*, pages 506–510, 2005.

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

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

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

[9] D. F. Silva and E. A. P. A. B. Batista. Speeding up all-pairwise dynamic time warping matrix calculation - website. `http://sites.labic.icmc.usp.br/prunedDTW/`, September 2015.

[10] S. Spiegel, B.-J. Jain, and S. Albayrak. Fast time series classification under lucky time warping distance. In *ACM SAC*, pages 71–78, 2014.

[11] L. Ulanova, N. Begum, and E. Keogh. Scalable clustering of time series with u-shapelets. In *SDM*, pages 900–908, 2015.

[12] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[13] Q. Zhu, G. E. A. P. A. Batista, T. Rakthanmanon, and E. Keogh. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. In *SDM*, pages 999–1010, 2012.