Time Warp Edit Distances with Stiffness Adjustment for Time Series Matching

Pierre-François Marteau

Time Warp Edit Distances with Stiffness Adjustment for Time Series Matching

(September 2006)

Pierre-François Marteau[1]

Abstract-- **In a way similar to the *string-to-string correction problem* we address time series similarity in the light of a *time-series-to-time-series-correction problem* for which the similarity between two time series is measured as the minimum cost sequence of "*edit operations*" needed to transform one time series into another. To define the "*edit operations*" we use the paradigm of a graphical editing process and end up with a dynamic programming recursive algorithm we call Time Warp Edit Distance (TWED). TWED is slightly different in form from Dynamic Time Warping or Edit Distance with Real Penalty algorithms. In particular it highlights a parameter that drives a kind of *stiffness* of the elastic measure. We show that the similarity provided by TWED is a metric potentially useful in time series retrieval applications since they could benefit from the triangular inequality property to speed up the retrieval process while tuning the *stiffness* of the elastic measure. Empiric quality of the TWED distance is evaluated on a simple classification task. Compared to Edit Distance, Dynamic Time Warping and Edit Distance with Real Penalty, TWED exhibits promising behaviors.**

Index Terms-- **Pattern Recognition, Clustering, Algorithms, Similarity Measures.**

I. INTRODUCTION

The problem of searching for times series within large datasets which are close to a given query element under some similarity criterion is faced by more and more computer applications. Among numerous examples, we find financial and stock data analysis [14], moving objects identification [4], astronomy [7], medicine [8], meteorology, data mining [1], etc.

Manuscript received September 19, 2006. This work was supported in part by the French Ministère de l'enseignement supérieur et de la recherche, Plan Pluri-formation "Modélisation" Valoria, Université de Bretagne Sud.

[1]P.F. marteau is with the VALORIA Computer Science Lab., Université de Bretagne Sud, BP573, 56017 Vannes, France, (telephone: +33 (0)2 01 72 99 , e-mail: pierre-francois.marteau@ univ-ubs.fr).

All these applications embed time series in a representation space and exploit some similarity measure defined for this space. Similarity measures are not equivalent given a specific application and fall basically into three categories:

- Non elastic metrics such as Lp-norms that do not support time shifting such as Euclidian Distance (ED),
- Elastic similarity measures that tolerate time shifting but are not metrics such as Dynamic Time Warping (DTW) [12], [10]
- Elastic metrics that tolerate time shifting such as Edit distance with Real Penalty (ERP) [5].

When considering time series information retrieval, working in a metric space is appealing since a lot of data structures (essentially tree based structures) and algorithms (partitioning, pivoting, etc.) have been optimized and made available for indexing and retrieving objects efficiently in metric spaces: see [3] for a review. All these structures and algorithms take advantage of the triangular inequality that allows for efficient pruning of a large amount of time series too far from the query.

We address in this paper the case of elastic metrics, namely elastic similarity measures that jointly exploit time shifting and possess all the properties of a distance, in particular the triangle inequality. Our contribution is basically three fold:

- The first contribution of this paper is the proposal of new elastic metrics which we call TWED (“Time Warp Edit Distances”). This contribution has to be placed in the perspective of former works that seek to combine Lp-norms to the edit distance, in particular in the light of the ERP distance [5] that can support local time shifting while

being a metric.

- The second contribution is related to the introduction of a parameter we call *stiffness* that drives the *elasticity* of TWED, placing this kind of distance in between the Euclidian distance (somehow a distance with '*infinite stiffness'*) and DTW (somehow a similarity measure with no '*stiffness'* at all).
- The third contribution of the paper is an empiric evaluation of the quality of TWED based on a simple classification experiment that provides some highlights on the behavior of TWED comparatively to the Euclidian Distance (ED), DTW, and ERP. The influence of the *stiffness* parameter on classification error rates is also analyzed.

II. ELASTIC SIMILARITY IN THE LIGHT OF THE SYMBOLIC EDIT DISTANCE

The Levenshtein Distance (LD) proposed in 1966 [6], also known as the edit distance, is the smallest number of insertions, deletions, and substitutions required to change one string into another. Wagner and Fisher [13] in 1974 developed a computationally efficient algorithm to calculate LD in O(n.m) using dynamic programming [2]. Meanwhile Dynamic Time Warping, that shares many similarities with LD despite the fact that it is not a metric, has been proposed in 1970 [12] and 1971 [10] to align speech utterances, namely time series, with time shift tolerances. More recently, the Edit Distance with Real Penalty (ERP) has been proposed as an edit distance based metric for time series matching with time shift tolerance. We present shortly hereinafter DTW and ERP in the light of the edit distance and develop the TWED metrics as an alternative to ERP.

A. Definitions

Let U the set of finite time series: $U = \{A_1^p \,/\, p \in N^+\} \cup \{\Omega\}$, where Ω is the empty time series (with null length), A_1^p is a time series with time index varying between *1* and *p*.

Let A be a finite discrete time series. Let a'_i be the i^{th} sample of time series A. We will consider that $a'_i \in S \times T$ where $S \subset R^d$ with $d \geq 1$ embeds the multidimensional space variables and $T \subset R$ embeds the time stamp variable, so that we can write $a'_i = (a_i, t_{a_i})$ where $a_i \in S$ and $t_{a_i} \in T$.

A_i^j with $i \leq j$ is the sub time series consisting of the i^{th} through the j^{th} samples (inclusive) of A. So $A_i^j = a'_i\, a'_{i+1} ... a'_j$. $|A|$ denotes the length (the number of samples) of A. Λ denotes the null sample. A_i^j with $i > j$ is the null time series noted Ω.

An edit operation is a pair $(a', b') \neq (\Lambda, \Lambda)$ of time series samples, written $a' \rightarrow b'$. Time series B results from the application of the edit operation $a \rightarrow b$ into time series A, written $A \Rightarrow B$ *via* $a' \rightarrow b'$, if $A = \sigma\, a' \tau$ and $B = \sigma\, b' \tau$ for some time series σ and τ. We call $a' \rightarrow b'$ a match operation if $a \neq \Lambda$ and $b' \neq \Lambda$, a delete operation if $b' = \Lambda$, an insert operation if $a' = \Lambda$.

Similarly to the edit distance defined for string [6], we define $\delta(A, B)$ the similarity between any two time series A and B of finite length, respectively p and q as:

$$\delta(A_1^p, B_1^q) = Min \begin{cases} \delta(A_1^{p-1}, B_1^q) + \Gamma(a'_p \rightarrow \Lambda) \\ \delta(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \rightarrow b_q) \\ \delta(A_1^p, B_1^{q-1}) + \Gamma(\Lambda \rightarrow b'_q) \end{cases}$$

Where $p \geq 1$, $q \geq 1$ and Γ is an arbitrary cost function which assigns to each edit operation $a' \rightarrow b'$ a nonnegative real number $\Gamma(a' \rightarrow b')$.

The recursion is initialized setting:

$$\delta(A_1^0, B_1^0) = 0$$
$$\delta(A_1^0, B_1^j) = \sum_{k=1}^{j} \Gamma(\Lambda \rightarrow b'_k), j \in \{1,..,q\}$$
$$\delta(A_1^i, B_1^0) = \Gamma(\sum_{k=1}^{i} a'_k \rightarrow \Lambda), i \in \{1,.., p\}$$

We notice that Dynamic Time Warping (DTW) and Edit Distance with Real penalties (ERP) are special cases of the previous definitions:

B. The DTW special case

The DTW similarity measure [12][10] δ_{DTW} is defined according to the previous notations as:

$$\delta_{DTW}(A_1^p, B_1^q) = d_{LP}(a_p, b_q) + Min \begin{cases} \delta_{DTW}(A_1^{p-1}, B_1^q) \\ \delta_{DTW}(A_1^{p-1}, B_1^{q-1}) \\ \delta_{DTW}(A_1^p, B_1^{q-1}) \end{cases}$$

where $d_{LP}(x, y)$ is the *Lp* norm of vector *x-y* in R^d,

and so for DTW, $\Gamma(a'_p \rightarrow \Lambda_q) = \Gamma(\Lambda_p \rightarrow b'_q) = \Gamma(a'_p \rightarrow b'_q) = d_{LP}(a_p, b_q)$

We notice that the time stamp values are not used so that the costs of each edit operation involve vectors *a* and *b* in *S* instead of vectors *a'* and *b'* in $S \times T$. One of the main restrictions of

δ_{DTW} is that it does not comply with the triangle inequality as shown by the following example:

$$A_1^1 = [1];\ B_1^2 = [1,2];\ C_1^3 = [1,2,2](*)$$
$$\delta_{DTW}(A_1^1, B_1^2) = 1;\ \delta_{DTW}(B_1^2, C_1^3) = 0;\ \delta_{DTW}(A_1^1, C_1^3) = 2$$
$$\text{and thus}: \delta_{DTW}(A_1^1, C_1^3) > \delta_{DTW}(A_1^1, B_1^2) + \delta_{DTW}(B_1^2, C_1^3)$$

(*) *1D time series with no stamp value given*

C. The ERP special case

$$\delta_{ERP}(A_1^p, B_1^q) = Min\begin{cases}\delta_{ERP}(A_1^{p-1}, B_1^q) + \Gamma(a'_p \to \Lambda)\\ \delta_{ERP}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \to b_q)\\ \delta(A_1^p, B_1^{q-1}) + \Gamma(\Lambda \to b'_q)\end{cases}$$

$$\text{with}\quad \begin{aligned}&\Gamma(a'_p \to \Lambda) = d_{LP}(a_p, g)\\ &\Gamma(a'_p \to b_q) = d_{LP}(a_p, b_q)\\ &\Gamma(\Lambda \to b'_q) = d_{LP}(g, b_q)\end{aligned}$$

and g a constant.

where $d_{LP}(x, y)$ is the *Lp* norm of vector *x-y* in *S*. Note that the time stamp coordinate is not taken into account so that δ_{ERP} is a distance on S but not on $S \times T$ since .

We notice here again that the time stamp values are not used so that the costs of each edit operation involve vectors *a* and *b* in R^d instead of vectors *a'* and *b'* in R^{d+1}.

According to the authors of ERP [5], the constant *g* should be set to *0* for some intuitive geometric interpretation and in order to preserve the mean value of the transformed time series when adding gaps samples.

D. The TWED distances

We propose two other alternatives to the definition of the edit operations for time series alignment leading to the definition of the new similarity measures, TWED and TWED1. To understand the semantic associated to the edit operations for TWED and TWED1, we reconsider the editing analogy with string and suggest some differences. The edit distance between two strings is defined as the minimal transformation cost allowing for the transformation of the first string into the second. For string edition, a transformation is a finite sequence of edit operations whose associated cost is the sum over the sequence of edit operations of the elementary costs Γ associated to each edit operation.

For time series we are seeking a sequence of edit operations allowing for the transformation of two time series simultaneously in order to make them superimposed with a minimal cost. If we use a graphical editor paradigm, we can imagine a *2D* representation of time series for which the horizontal axis represents the time scale or the time stamp coordinate and the vertical axis represents a spatial coordinate scale displaying the projection of the *d-1* spatial coordinates of the samples onto a *1D* scale. The graphical editor we imagine allows for the editing of two time series *A* and *B* using three elementary edit operations depicted in Fig. *1a, 1.b* and *1.c*. Instead of the classical *delete*, *insert*, *match* operations, we introduce *delete-A*, *delete-B* and *match* operations as follows:

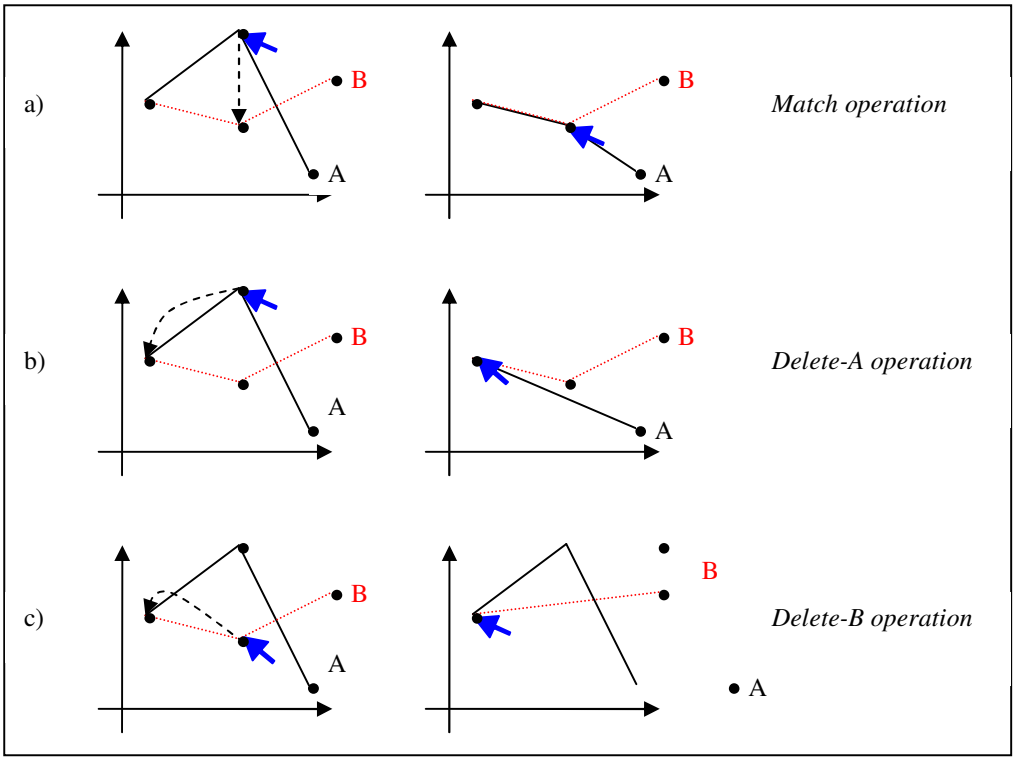

Fig 1: The edit operations in the graphical editor paradigm.

the *delete-A* (delete in the first time series) operation (Fig. *1.b*) consists in clicking on the dot representing the sample in *A* to delete (a'_i) and dragging and dropping this dot onto the previous sample dot (a'_{i-1}). We can suggest that the editing effort or cost associated to this delete operation is proportional to the length of vector ($a'i - a'_{i-1}$).

the *delete-B* (delete in the second time series) operation (Fig. *1.c*) consists in clicking on the dot representing the sample in *B* to delete (b'_i) and dragging and dropping this dot onto the previous sample dot (b'_{i-1}). We can suggest that the editing effort or cost associated to this delete operation is proportional to the length of vector ($b'i - b'_{i-1}$).

The *match* operation (Fig. *1.a*) consists in clicking on the dot representing the sample (a'_i) to

match and then dragging and dropping this dot onto the graphic position corresponding to the matching sample (b'_j). We can suggest that the editing effort or cost associated to the insert operation is proportional to the length of vector ($b'j - a'_i$).

This provides the basis for the first version of TWED distance we propose:

$$\delta_{TWED}(A_1^p, B_1^q) = Min\begin{cases} \delta_{TWED}(A_1^{p-1}, B_1^q) + d(a'_p, a'_{p-1}) \\ \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + d(a'_p, b'_q) \\ \delta_{TWED}(A_1^p, B_1^{q-1}) + d(b'_{q-1}, b'_q) \end{cases}$$

The recursion is initialized setting:

$$\delta_{TWED}(A_1^0, B_1^0) = 0$$
$$\delta_{TWED}(A_1^0, B_1^j) = \sum_{k=1}^{j} d(b'_k, b'_{k-1}),\ j \in \{1,..,q\}$$
$$\delta_{TWED}(A_1^i, B_1^0) = \sum_{k=1}^{i} d(a'_k, a'_{k-1}),\ i \in \{1,..,p\}$$
with $a'_0 = b'_0 = 0$ by convention.

Furthermore, using the graphical editor paradigm, we define the time series matching game as follows: two time series, *A* and *B*, are displayed on the graphic. The goal consists in editing *A* and *B* in order to completely superimpose the two curves.

The editing process is performed left to right: if *i* is an index on the samples of *A* and *j* on the samples of *B*, then the process is initialized setting *i=j=1*. A match operation will increment *i* and *j* simultaneously: $i \leftarrow i+1, j \leftarrow j+1$. A *delete-A* operation will increment *i* only: $i \leftarrow i+1$. A *delete-B* operation will increment *j* only: $j \leftarrow j+1$.

According to the above mentioned constraint, once a sample a'_i in A has been processed using either a *match* or a *delete-A* operation, it is impossible to edit it again and so it is for former samples a'_r, r in $\{1..i\}$. Similarly, once sample b'_j in B has been used either in a *match* or in an *delete-B* operation it is impossible to use former samples b'_r, r in $\{1..j\}$ for future match or insertion operations. Therefore, according to this game, the editing process provides a sequence of edit operations as well as ordered pairs of indexes (i,j) where i is an index in A and j an index in B. In other words, the process provides an ordered sequence of triplets (op_k, i_k, j_k) where op_k is the k^{th} edit operation selected, and i_k and j_k are the values of the index in A and B respectively when the edit operation is performed. A partial order can be defined on the triplets as follows:

$$(op_{k1}, i_{k1}, j_{k1}) < (op_{k2}, i_{k2}, j_{k2}) \text{ iff } i_{k1} \leq i_{k2} \text{ and } j_{k1} \leq j_{k2} \text{ and either } i_{k1} \neq i_{k2} \text{ or } j_{k1} \neq j_{k2}.$$

Since for each step of the editing game, one of the indexes is increased by one while the other is either incremented by one or remains unchanged, all the triplets in the output editing sequence are ordered in increasing order.

Supposing that the game editing process has provided a sequence of edit operations up to i_k and j_k index values, if the sub sequences $\overline{A}_1^{i_{k-1}}$ ($\overline{A}_1^{i_{k-1}}$ refers to the sequence obtained from A after the first $k-1$ edit operations) and $\overline{B}_1^{j_{k-1}}$ are not superimposed, then, as there exists no possibility to process former samples to superimpose them the game process cannot be successful.

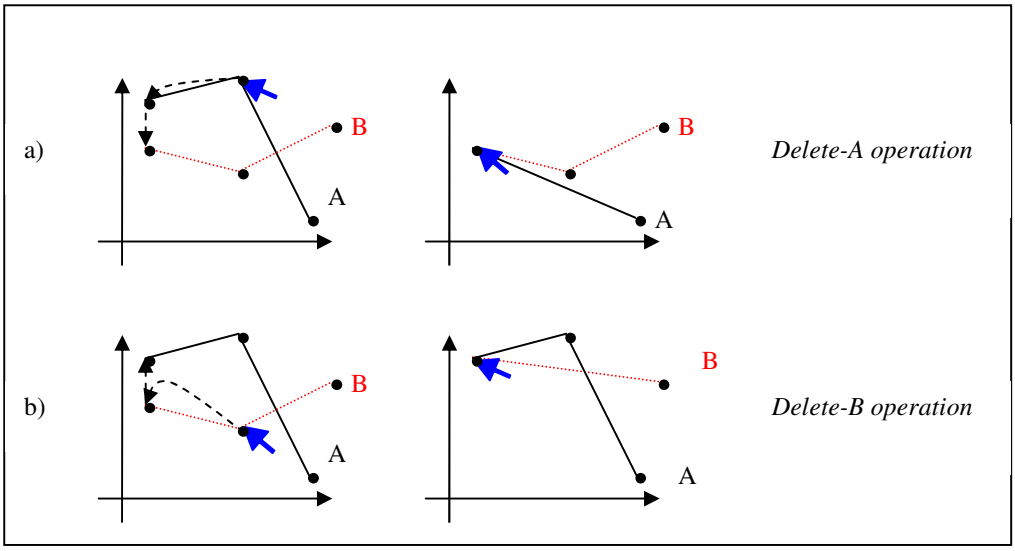

Fig 2: The graphical editor paradigm revisited for the insert and delete operations.

Finally supposing that the game editing process has provided a successful sequence of edit operations up to i_{k-1} and j_{k-1} index values, e.g. sub sequences $\overline{A}_1^{i_{k-1}}$ and $\overline{B}_1^{j_{k-1}}$ are superimposed. This leads to the following cost policy for the edit operations that depend on the current state of the graphical editor process:

- if the process decides to perform a *delete-A* operation so that $i_k=i_{k-1}+1$ and $j_k=j_{k-1}$, the process will be successful if sequences $\overline{A}_1^{i_k}$ and $\overline{B}_1^{j_k}$ are superimposed. If so, necessarily we should have $\overline{a}'_{i_k} = \overline{a}'_{i_{k-1}} = b'_{j_k}$ and the cost of the delete operation can be chosen proportional to $\left|a'_{i_k} - a'_{i_{k-1}}\right|$ to which an extra cost can be envisaged to make $a'_{i_{k-1}}$ superimposed with $\overline{a}'_{i_{k-1}} = b'_{j_k}$, namely $\left|b'_{j_k} - a'_{i_{k-1}}\right|$ as shown in Fig. 2.a.
- if the process decides to perform a *delete-B* operation so that $i_k=i_{k-1}$ and $j_k=j_{k-1}+1$, the process will be successful if sequences $\overline{A}_1^{i_k}$ and $\overline{B}_1^{j_k}$ are superimposed. If so,

necessarily we should have $\overline{a}'_{i_k} = \overline{a}'_{i_{k-1}} = b'_{j_{k-1}}$ and the cost of the insert operation can be chosen proportional to the length of vector $b'_{j_k} - b'_{j_{k-1}}$. But, since the insert operation is performed from a'_{i_k} instead of $\overline{a}'_{i_k}$ an extra cost proportional to $\left|a'_{i_k} - \overline{a}'_{i_k}\right| = \left|a'_{i_k} - b'_{j_{k-1}}\right|$ can be envisaged as shown in Fig. 2.b.

This leads to a second version of the TWED distance:

$$\delta_{TWED1}(A_1^p, B_1^q) = Min\begin{cases} \delta_{TWED1}(A_1^{p-1}, B_1^q) + d(a'_p, a'_{p-1}) + d(b'_q, a'_{p-1}) \\ \delta_{TWED1}(A_1^{p-1}, B_1^{q-1}) + d(a'_p, b'_q) \\ \delta_{TWED1}(A_1^p, B_1^{q-1}) + d(b'_{q-1}, b'_q) + d(b'_{q-1}, a'_p) \end{cases}$$

where d is any distance on R^{d+1}.

The recursion is initialized setting:

$$\delta_{TWED1}(A_1^0, B_1^0) = 0$$
$$\delta_{TWED1}(A_1^0, B_1^j) = \sum_{k=1}^{j} d(b'_k, b'_{k-1}),\ j \in \{1,..,q\}$$
$$\delta_{TWED1}(A_1^i, B_1^0) = \sum_{k=1}^{i} d(a'_k, a'_{k-1}),\ i \in \{1,..,p\}$$
with $a'_0 = b'_0 = 0$ by convention.

Theorem 1: δ_{TWED} is a distance on the set of finite discrete time series U:

P1: $\delta_{TWED}(A,B) \geq 0$ for any finite discrete time series A and B,

P2: $\delta_{TWED}(A,B) = 0$ iff $A = B$ for any finite discrete time series A and B,

P3: $\delta_{TWED}(A,B) = \delta_{TWED}(B,A)$ for any finite discrete time series A and B,

P4: $\delta_{TWED}(A,B) \leq \delta_{TWED}(A,C) + \delta_{TWED}(C,B)$ for any finite discrete time series *A*, *B* and *C*.

Theorem 2: δ_{TWED1} is also a distance on the set of finite discrete time series *U*, i.e. properties P1, P2, P3 and P4 holds for δ_{TWED1}.

A sketch of proofs for these two theorems is given in appendix.

E. Providing 'stiffness' into TWED

Going back to the graphical editor game we have envisaged that the penalty or cost associated to each edit operation should be proportional to the mouse pointer displacement involved during the edition. If we separate the spatial displacement in *S* from the temporal displacement in *T* then we have to consider a spatial penalty that could be handled by a distance measured in *S* and a temporal penalty more or less proportional to some distance measured in *T*. By doing this, we could parameterized a distance in between the Euclidian Distance, that is characterized with a kind of '*infinite stiffness'*, and DTW that is characterized with a 'null fitness'. In practice, we can choose $d(a',b') = d_{LP}(a,b) + \gamma.d_{Lp}(t_a,t_b)$ where γ is a non negative constant that characterizes the *stiffness* of TWED elastic measures. Notice that $\gamma > 0$ is required for TWED to be a distance. if $\gamma = 0$ then TWED will be a distance on *S* but not on $S \times T$.

Regarding the way *stiffness* is handled, the main difference between δ_{TWED} and δ_{TWED1} lies in deletions operations. In δ_{TWED} *stiffness* is proportional to the time interval between two successive samples while in δ_{TWED1} an extra penalty is introduced proportionally to the time difference

between the sample preceding the deleted one in the first time series and the matching sample of the second time series. Thus in δ_{TWED1} *stiffness* is more or less equivalent for all edit operations while in δ_{TWED} *stiffness* is more effective on match operations comparatively to delete operations.

F. Algorithmic complexity of TWED

The time complexity of TWED is the same as DTW and ERP, namely *O(m.n)*, where *m* and *n* are the lengths of the two time series being matched. The space complexity is also the same as DTW i.e. *O(m.n)*, but as well as the ERP distance, the costs $\gamma(a' \rightarrow \Lambda)$ and $\gamma(\Lambda \rightarrow b')$ can be tabulated leading to a extra space complexity of *O(m+n)* for δ_{TWED} and O(m.n) for δ_{TWED1}.

III. EXPERIMENTATIONS

To evaluate empirically the behavior of the TWED distances comparatively to other metrics or similarity measures, we address a simple classification task experiment. The classification task we have considered consists in affecting one of the possible categories for the 16 data sets available at UCR repository [11] to an unknown time series.

For each dataset, a train subset is defined as well as a test subset. The classification is based on the simple nearest neighbor decision rule: first we select a train data set containing time series for which the correct category is known. To affect a category to an unknown time series selected from a test data set (different from the train set), we select its nearest neighbor (in the sense of a distance or similarity measure) within the train data set, then affect to the unknown time series

the category associated to its nearest neighbor.

Given a dataset, we can adapt the *stiffness* parameter as follows: we use the train dataset to select the '*best stiffness'* value, namely the one leading to the minimal error rate, according to a leave one out procedure performed on the train set (this procedure consists in selecting iteratively one time series into the train set and then in considering it as a test against the remaining time series within the train set itself). Finally, the test dataset is used to evaluate the final error rate with the best '*stiffness value'* estimated on the train set. This leads to OTWED, that implements δ_{TWED} and OTWED1 that implements δ_{TWED1}, the optimized versions for TWED. For our experiment, '*stiffness values'* are selected into $\{1e10^{-5}, 1e10^{-4}, 1e10^{-3}, 1e10^{-2}, e10^{-1}, 1, 10\}$.

Tab.1 shows the results obtained for the tested methods, e.g. Euclidian Distance on the original time series, optimized DTW with best warping windows (ODTW) as defined in [9], classical DTW (DTW) with no warping window, Edit distance with Real Penalty *(*ERP*)* as defined in [5] OTWED and OTWED1 for which the parameter value γ is selected for each dataset such as to minimize the classification errors estimated on the train data. During the optimization process, γ takes values into the set $\{1e10^{-4}, 1e10^{-3}, 1e10^{-2}, e10^{-1}, 1, 10\}$. If different γ values lead to the minimal error rate estimated on the train data then the **highest γ value is selected.**

For ERP, OTWED and OTWED1 we used the L1-norm, while the L2-norm has been implemented in DTW and ODTW [9]. The gap value used in ERP has been set to 0 has suggested by the authors [5].

Finally, as time is not explicitly given for theses datasets, we used the index value of the samples as the time stamps for the whole experiment. This leads to the following implementations of TWED:

$$\delta_{TWED}(A_1^p, B_1^q) = Min\begin{cases} \delta_{TWED}(A_1^{p-1}, B_1^q) + \left|a_p, a_{p-1}\right| + \gamma \\ \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + \left|a_p, b_q\right| + \gamma.\left|p-q\right| \\ \delta_{TWED}(A_1^p, B_1^{q-1}) + \left|b_{q-1}, b_q\right| + \gamma \end{cases}$$

$$\delta_{TWED1}(A_1^p, B_1^q) = Min\begin{cases} \delta_{TWED1}(A_1^{p-1}, B_1^q) + \left|a_p, a_{p-1}\right| + \gamma.\left|p-1-q\right| \\ \delta_{TWED1}(A_1^{p-1}, B_1^{q-1}) + \left|a_p, b_q\right| + \gamma.\left|p-q\right| \\ \delta_{TWED1}(A_1^p, B_1^{q-1}) + \left|b_{q-1}, b_q\right| + \gamma.\left|p-q+1\right| \end{cases}$$

Dataset	Nbr of classes | Size of testing set	1-NN ED	1-NN ODTW	1-NN DTW	1-NN ERP	1-NN OTWED	1-NN OTWED1
Synthetic Control	6|300	0.12	0.017	**0.007**	0.036	0.04	0.04
Gun-Point	2|150	0.087	0.087	0.093	0.04	**0.007**	0.026
CBF	3|900	0.148	0.004	**0.003**	**0.003**	0.044	**0.003**
Face (all)	14|1690	0.286	**0.192**	**0.192**	0.202	0.238	0.212
OSU Leaf	6|242	0.483	0.384	0.409	0.397	0.339	**0.335**
Swedish Leaf	15|625	0.213	0.157	0.210	**0.12**	0.139	**0.12**
50Words	50|455	0.369	0.242	0.310	0.281	0.262	**0.185**
Trace	4|100	0.24	0.01	**0.0**	0.17	0.03	0.01
Two Patterns	4|4000	0.09	0.0015	**0.0**	**0**	0.0235	0.0005
Wafer	2|6174	**0.005**	**0.005**	0.020	0.009	**0.005**	**0.005**
Face (four)	4|88	0.216	0.114	0.170	0.102	0.205	**0.068**
Lighting2	2|61	0.246	**0.131**	**0.131**	0.148	**0.131**	0.148
Lighting7	7|73	0.425	0.288	0.274	0.301	0.329	**0.247**
ECG	2|100	0.12	0.12	0.23	0.13	**0.1**	0.11
Adiac	37|391	0.389	0.391	0.396	0.378	0.401	**0.376**
Yoga	02|3000	0.170	0.155	0.164	0.147	0.166	**0.129**
Fish	7|175	0.267	0.233	0.267	0.12	0.171	**0.109**
Coffee	228	0.25	**0.179**	**0.179**	0.25	0.25	0.25
OliveOil	4|30	0.133	0.167	**0.133**	0.167	**0.133**	0.167
Beef	5|30	0.467	**0.467**	0.5	0.5	0.567	0.5

TAB.1: COMPARATIVE STUDY USING THE UCR DATASETS [11]: CLASSIFICATION ERROR RATE OBTAINED USING THE FIRST NEAR NEIGHBOUR CLASSIFICATION RULE FOR ED, DTW, DTW WITH BEST WARPING WINDOW, ERP, AND TWED DISTANCES

Dataset	$\gamma=10^{-5}$	$\gamma=10^{-4}$	$\gamma=10^{-3}$	$\gamma=10^{-2}$	$\gamma=10^{-1}$	$\gamma=1$	$\gamma=10$
Synthetic Control	0.023	0.023	0.023	**0.020**	0.040	0.09	0.090
Gun-Point	0.041	0.02	**0**	0.02	0.082	0.082	0.082
CBF	0.034	0.034	0.034	**0.034**	0.069	0.138	0.172
Face (all)	0.034	0.034	0.034	**0.027**	0.045	0.086	0.082
OSU Leaf	0.286	0.286	**0.281**	0.356	0.457	0.357	0.362
Swedish Leaf	0.212	0.210	0.200	**0.150**	0.190	0.230	0.230
50Words	0.350	0.327	**0.263**	0.322	0.421	0.325	0.332
Trace	0.01	0.01	**0.01**	0.081	0.141	0.151	0.151
Two Patterns	0.024	0.024	0.025	0.033	0.059	**0.023**	0.029
Wafer	0.016	0.016	0.015	0.008	0.005	0.003	**0.003**
Face (four)	0.174	0.174	0.174	**0.130**	0.261	0.304	0.261
Lighting2	**0.051**	0.068	0.102	0.203	0.220	0.254	0.271
Lighting7	0.348	0.348	0.290	0.261	**0.246**	0.319	0.319
ECG	0.212	0.212	0.182	0.162	**0.131**	0.141	0.141
Adiac	0.456	0.458	0.429	0.426	0.419	0.419	**0.419**
Yoga	0.177	**0.167**	0.181	0.241	0.251	0.224	0.224
Fish	0.230	**0.207**	0.218	0.299	0.282	0.236	0.236
Coffee	0.185	**0.185**	0.222	0.259	0.259	0.222	0.259
OliveOil	0.138	**0.138**	0.172	0.172	0.172	0.172	0.172
Beef	0.621	0.621	**0.552**	0.670	0.670	0.670	0.670

TAB.2: ERROR RATES FOR δ_{TWED} EVALUATED ON THE TRAIN DATA FOR THE 20 TESTED DATASETS IN FUNCTION OF γ

Dataset	$\gamma=10^{-5}$	$\gamma=10^{-4}$	$\gamma=10^{-3}$	$\gamma=10^{-2}$	$\gamma=10^{-1}$	$\gamma=1$	$\gamma=10$
Synthetic Control	0.030	0.030	**0.027**	0.033	0.033	0.090	0.090
Gun-Point	0.041	0.041	0.041	**0.020**	0.061	0.081	0.081
CBF	0	0	**0**	0.034	0.103	0.172	0.172
Face (all)	0.018	0.018	**0.016**	0.018	0.038	0.082	0.082
OSU Leaf	0.241	0.246	**0.241**	0.276	0.357	0.362	0.362
Swedish Leaf	0.144	0.144	0.144	**0.128**	0.204	0.230	0.230
50Words	0.216	0.214	**0.189**	0.232	0.323	0.332	0.332
Trace	0.01	0.01	**0.01**	0.05	0.151	0.151	0.151
Two Patterns	0	0	0	**0**	0.01	0.029	0.029
Wafer	0.003	0.003	0.003	**0.001**	0.002	0.002	0.003
Face (four)	0.087	0.087	0.087	**0.087**	0.0217	0.261	0.261
Lighting2	0.153	0.169	**0.153**	0.237	0.254	0.271	0.271
Lighting7	0.304	0.304	0.275	**0.261**	0.304	0.319	0.319
ECG	0.172	0.172	0.172	0.152	**0.152**	0.141	0.141
Adiac	0.406	0.404	**0.393**	0.409	0.419	0.419	0.419
Yoga	0.150	0.147	**0.147**	0.187	0.224	0.224	0.224
Fish	**0.167**	0.172	0.184	0.230	0.236	0.236	0.236
Coffee	0.259	0.259	0.259	0.259	**0.222**	0.259	0.259
OliveOil	0.207	0.172	0.172	0.172	0.172	0.172	**0.172**
Beef	0.655	0.690	0.690	0.690	0.690	0.690	**0.690**

TAB.3: ERROR RATES FOR δ_{TWED1} EVALUATED ON THE TRAIN DATA FOR THE 20 TESTED DATASETS IN FUNCTION OF γ

Tab.2 shows the evolution of the error rate evaluated on the train data in function of the '*stiffness'* for some datasets. Clearly, we show that the error rates are quite sensitive to the '*stiffness'* magnitude, but the relative regularity of the error rates indicates that adjusting the '*stiffness'* parameter makes sense for most of the tested datasets.

This experiment shows that the TWED distances perform well, especially the δ_{TWED1} distance that exhibits the lowest error rates on the test data in average.

IV. CONCLUSION

From a graphical curve editing perspective and from earlier work on symbolic edit distance and dynamic time warping we have developed elastic similarity measures called TWED to match time series with some time shifting tolerance. We have proved that the two versions of TWED measures we proposed are metrics and as such they present all methods developed for searching in metric spaces as potential solutions for time series searching and retrieval applications that require some time shift tolerance. The originality of TWED comparatively to the ERP distance, apart from the way insertions and deletions are managed, lies in the introduction of a parameter that drives the '*stiffness'* of the elastic measure that places TWED in between Euclidian distances and DTW similarity measures. This parameter can be optimized for each application or dataset. The empirical quality of the distance is evaluated on a classification experiment based on the first near neighbour classification rule for 20 different datasets. Globally, this experiment shows that at least one version of TWED outperforms in average the ERP distances and DTW similarity measures on the experimented datasets. Furthermore, the experiment shows that classification error rates are quite sensitive to the '*stiffness'* parameter while showing some regular behaviour.

Large classification improvements can be obtained while adjusting this parameter according to the processed dataset.

REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns IEEE, pages 3–14, 1995

[2] R. Bellman, Dynamic Programming, Princeton Univ. Press, 1957, New Jersey.

[3] E. Chávez, G. Navarro, R. A. Baeza-Yates, José L. Marroquín, Searching in Metric Spaces, ACM Computing Surveys (CSUR), Volume 33 , Issue 3, P: 273 – 321, 2001

[4] L. Chen., M. T , Ozsu,., V. Oria, Robust and fast similarity search for moving object trajectories. *SIGMOD*, 2005.

[5] L. Chen & R. Ng. On the marriage of Lp-norm and edit distance . In Proc. 30th Int'l Conf. on Very Large Data Bases, pp 792–801, 2004.

[6] V. I. Levenshtein**,** Binary codes capable of correcting deletions, insertions, and reversals, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966.

[7] M.K, Ng, & , Z, Huang. (1997). Temporal data mining with a case study as astronomical data analysis. *Lecture Notes in Computer Sciences*. Springer. pp. 2-18.

[8] Lin, J., Keogh, E., Fu, A. & Van Herie, H. (2005). Approximations to Magic: Finding Unusual Medical Time Series.. In *proceedings of the 18th IEEE Int'l Symposium on Computer-Based Medical Systems.* June 23-24. Dublin, Ireland.

[9] C. A. Ratanamahatana E. Keogh. Making Time-series Classification More Accurate Using Learned Constraints, SDM, Orlando, USA, 2004

[10] H. Sakoe and S. Chiba, "A dynamic programming approach to continuous speech ecognition," in Proc. 7th Int. Congr. Acoust., Budapest, Hungary, Aug. 1971, pp. 65-68.

[11] UCR Time Series Classification/Clustering Page, Keogh & al., http://www.cs.ucr.edu/~eamonn/time_series_data/

[12] V. M. Velichko and N. G. Zagoruyko, "Automatic recognition of 200 words," Int. J. Man-Machine Studies, vol. 2, pp. 223-234, 1970.

[13] R. A. Wagner , Michael J. Fischer, The String-to-String Correction Problem,Journal of the ACM (JACM), Volume 21 , Issue 1, P: 168 – 73, 1974.

[14] H. Wu, B. Salzberg,, D. Zhang. Online event-driven subsequence matching over financial data streams. *SIGMOD*, 2004.

APPENDIX

Let U be the set of finite discrete time series: $U = \{A_1^p / p \in N^+\} \cup \{\Omega\}$, where Ω is the empty time series (with null length). Let define δ_{TWED} and δ_{TWED1} as:

$$\delta_{TWED}(A_1^p, B_1^q) = Min \begin{cases} \delta_{TWED}(A_1^{p-1}, B_1^q) + d(a'_p, a'_{p-1}) \\ \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + d(a'_p, b'_q) \\ \delta_{TWED}(A_1^p, B_1^{q-1}) + d(b'_{q-1}, b'_q) \end{cases}$$

$$\delta_{TWED1}(A_1^p, B_1^q) = Min \begin{cases} \delta_{TWED1}(A_1^{p-1}, B_1^q) + d(a'_p, a'_{p-1}) + d(b'_q, a'_{p-1}) \\ \delta_{TWED1}(A_1^{p-1}, B_1^{q-1}) + d(a'_p, b'_q) \\ \delta_{TWED1}(A_1^p, B_1^{q-1}) + d(b'_{q-1}, b'_q) + d(b'_{q-1}, a'_p) \end{cases}$$

where d is any distance on R^{d+1}. In practice, we will choose $d(a',b') = d_{LP}(a,b) + \gamma . d_{Lp}(t_a, t_b)$ where γ is a parameter that characterizes the *stiffness* of the elastic distance δ_{TWED}.

The recursion is initialized setting:

$$\delta_{TWED}(A_1^0, B_1^0) = 0$$

$$\delta_{TWED}(A_1^0, B_1^j) = \sum_{k=1}^{j} d(b'_k, b'_{k-1}), j \in \{1,..,q\}$$

$$\delta_{TWED}(A_1^i, B_1^0) = \sum_{k=1}^{i} d(a'_k, a'_{k-1}), i \in \{1,..,p\}$$

with $a'_0 = b'_0 = 0$ by convention.

Proof of theorem 1: δ_{TWED} is a distance on the set U of finite discrete time series:

P1: non-negativity

For all (A_1^p, B_1^q) in $U \times U$ let $m=p+q$. Non-negativity of δ_{TWED} is proved by induction on m.

P1 is true for $m=0$ by definition of δ_{TWED} and the induction hypothesis holds.

Suppose *P1* is true for all $m \in \{0,..n-1\}$ for some $n>0$. Then for all (A_1^p, B_1^q) in $U \times U$ such that $m = n$, as $\delta_{TWED}(A_1^{p-1}, B_1^q), \delta_{TWED}(A_1^{p-1}, B_1^{q-1})$ and $\delta_{TWED}(A_1^p, B_1^{q-1})$ are assumed positive and as the non-negativity of distance d holds, $\delta_{TWED}(A_1^p, B_1^q)$ is necessary non-negative, showing that P1 is true for all $m \in \{0,..n\}$. By induction, *P1* holds for all $m \in N$. □

P2: identity of indiscernibles

For all (A_1^p, B_1^q) in $U \times U$, if $A_1^p = B_1^q$ then $p=q$ and $\forall i \in \{1,..., p\}$, $a'_i = b'_i$. It is easy to show by induction on p that $0 \leq \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^p, B_1^p) \leq \sum_{i=1}^{p} d(a'_i, b'_i) = 0$ leading to $\delta_{TWED}(A_1^p, B_1^q) = 0$.

Now consider the backward proposition *P'2*: $\delta_{TWED}(A_1^p, B_1^q) = 0 \Rightarrow A_1^p = B_1^q$.

P'2 is proved by induction on $m=p+q$. □

P3: Symmetry

Proof: Since the distance d on the sample space $S \times T$ is symmetric, it is easy to show that $\delta_{TWED}(A_1^p, B_1^q)$ is symmetric for all (A_1^p, B_1^q) in $U \times U$ by induction on $m=p+q$. □

Lemma 1: For all A_1^p in U , $d(a_{p-1}, a_p) \geq \delta_{TWED}(A_1^{p-1}, A_1^p)$

Proof: For all A_1^p in U ,

$$d(a_{p-1}, a_p) = 0 + d(a_{p-1}, a_p) = \delta_{TWED}(A_1^{p-1}, A_1^{p-1}) + d(a_p, a_{p-1})$$

$$= \delta_{TWED}(A_1^{p-1}, A_1^{p-1}) + \Gamma(a_p \rightarrow \Lambda) \geq \delta_{TWED}(A_1^p, A_1^{p-1}) \square$$

(P4): Triangle inequality

For all (A_1^p, B_1^q, C_1^r) in $U \times U \times U$, $\delta_{TWED}(A_1^p, C_1^r) \leq \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^q, C_1^r)$.

Proof: We will proved *P4* by induction on *m=p+q+r*.

P4 is true for *m=0* since $\delta_{TWED}(\Omega, \Omega) = 0 \leq \delta_{TWED}(\Omega, \Omega) + \delta_{TWED}(\Omega, \Omega)$ and the induction hypothesis holds.

(H4): Suppose *P4* is true for all $m \in \{0,..n-1\}$ for some *n>0*. Let $\Sigma = \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^q, C_1^r)$. Then for all (A_1^p, B_1^q, C_1^r) in $U \times U \times U$ such that *m=n*, we have basically 9 different cases to explore for the decomposition of $\delta_{TWED}(A_1^p, B_1^q)$ and $\delta_{TWED}(B_1^q, C_1^r)$:

1st Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \rightarrow b'_q) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + \Gamma(b'_q \rightarrow c'_r) \end{cases}$, then:

$$\Sigma \geq \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + d(a'_p, b'_q) + d(b'_q, c'_r)$$
$$\geq \delta_{TWED}(A_1^{p-1}, C_1^{r-1}) + d(a'_p, c'_r) \underset{def}{\geq} \delta_{TWED}(A_1^{p}, C_1^{r}) \quad \text{since } (H4) \text{ applies.}$$

2nd Case: if $\begin{cases} \delta_{TWED}(A_1^{p}, B_1^{q}) = \delta_{TWED}(A_1^{p}, B_1^{q-1}) + \Gamma(\Lambda \rightarrow b'_q) \\ \delta_{TWED}(B_1^{q}, C_1^{r}) = \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + \Gamma(b'_q \rightarrow c'_r) \end{cases}$, then:

$$\Sigma = \delta_{TWED}(A_1^{p}, B_1^{q-1}) + d(b'_q, b'_{q-1}) + \delta_{TWED}(B_1^{q}, C_1^{r})$$

since (*lemma 1*) $d(b'_q, b'_{q-1}) \geq \delta_{TWED}(B_1^{q-1}, B_1^{q})$ we have

$$\Sigma \geq \delta_{TWED}(A_1^{p}, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, B_1^{q}) + \delta_{TWED}(B_1^{q}, C_1^{r})$$
$$\geq \delta_{TWED}(A_1^{p}, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^{r}) \geq \delta_{TWED}(A_1^{p}, C_1^{r}) \text{ since } (H4) \text{ applies twice.}$$

3rd Case: if $\begin{cases} \delta_{TWED}(A_1^{p}, B_1^{q}) = \delta_{TWED}(A_1^{p-1}, B_1^{q}) + \Gamma(a'_p \rightarrow \Lambda) \\ \delta_{TWED}(B_1^{q}, C_1^{r}) = \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + \Gamma(b'_q \rightarrow c'_r) \end{cases}$, then:

$$\Sigma = \delta_{TWED}(A_1^{p-1}, B_1^{q}) + d(a'_p, a'_{p-1}) + \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + d(b'_q, c'_r)$$

$$\Sigma \geq \delta_{TWED}(A_1^{p-1}, B_1^{q}) + \delta_{TWED}(A_1^{p}, A_1^{p-1}) + \delta_{TWED}(B_1^{q}, C_1^{r}) \quad (\textit{lemma } 1)$$
$$\geq \delta_{TWED}(A_1^{p}, A_1^{p-1}) + \delta_{TWED}(A_1^{p-1}, C_1^{r}) \quad (H4) \text{ applies}$$
$$\geq \delta_{TWED}(A_1^{p}, C_1^{r}) \quad (H4) \text{ applies twice.}$$

4th Case: if $\begin{cases} \delta_{TWED}(A_1^{p}, B_1^{q}) = \delta_{TWED}(A_1^{p}, B_1^{q-1}) + \Gamma(\Lambda \rightarrow b'_q) \\ \delta_{TWED}(B_1^{q}, C_1^{r}) = \delta_{TWED}(B_1^{q}, C_1^{r-1}) + \Gamma(\Lambda \rightarrow c'_r) \end{cases}$, then:

$$\Sigma = \delta_{TWED}(A_1^p, B_1^{q-1}) + d(b'_q, b'_{q-1}) + \delta_{TWED}(B_1^q, C_1^{r-1}) + d(c'_r, c'_{r-1})$$

since (*lemma 1*) $d(b'_q, b'_{q-1}) \geq \delta_{TWED}(B_1^{q-1}, B_1^q)$ and $d(c'r, c'_{r-1}) \geq \delta_{TWED}(C_1^{r-1}, C_1^r)$ we have:

$$\begin{aligned}
\Sigma &\geq \delta_{TWED}(A_1^p, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, B_1^q) + \delta_{TWED}(B_1^q, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \\
&\geq \delta_{TWED}(A_1^p, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \qquad (H4) \text{ applies} \\
&\geq \delta_{TWED}(A_1^p, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^r) \geq \delta_{TWED}(A_1^p, C_1^r) \qquad (H4) \text{ applies twice.}
\end{aligned}$$

5th Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \rightarrow b'_q) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^q, C_1^{r-1}) + \Gamma(\Lambda \rightarrow c'_r) \end{cases}$, then:

$$\begin{aligned}
\Sigma &= \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^q, C_1^{r-1}) + d(c'_r, c'_{r-1}) \\
&\geq \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^q, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \quad (lemma 1) \\
&\geq \delta_{TWED}(A_1^p, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \geq \delta_{TWED}(A_1^p, C_1^r) \quad (H4) \text{ applies twice.}
\end{aligned}$$

6th Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^{p-1}, B_1^q) + \Gamma(a'_p \rightarrow \Lambda) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^q, C_1^{r-1}) + \Gamma(\Lambda \rightarrow c'_r) \end{cases}$, then:

$$\begin{aligned}
\Sigma &= \delta_{TWED}(A_1^p, B_1^q) + + \delta_{TWED}(B_1^q, C_1^{r-1}) + d(c'_r, c'_{r-1}) \\
&\geq \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^q, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \quad (lemma 1) \\
&\geq \delta_{TWED}(A_1^p, C_1^{r-1}) + \delta_{TWED}(C_1^{r-1}, C_1^r) \geq \delta_{TWED}(A_1^p, C_1^r) \quad (H4) \text{ applies twice.}
\end{aligned}$$

7th Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \rightarrow b'_q) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^{q-1}, C_1^r) + \Gamma(b_q \rightarrow \Lambda) \end{cases}$, then:

$$\begin{aligned} \Sigma &= \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^{q-1}, C_1^r) + d(b'_q, b'_{q-1}) \\ &\geq \delta_{TWED}(A_1^p, B_1^q) + \delta_{TWED}(B_1^{q-1}, C_1^r) + \delta_{TWED}(B_1^q, B_1^{q-1}) \quad (lemma\,1) \\ &\geq \delta_{TWED}(A_1^p, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^r) \geq \delta_{TWED}(A_1^p, C_1^r) \quad (H4) \text{ applies twice.} \end{aligned}$$

8th Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^{p-1}, B_1^q) + \Gamma(a'_p \rightarrow \Lambda) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^{q-1}, C_1^r) + \Gamma(b_q \rightarrow \Lambda) \end{cases}$, then:

$$\begin{aligned} \Sigma &= \delta_{TWED}(A_1^{p-1}, B_1^q) + d(a'_p, a'_{p-1}) + \delta_{TWED}(B_1^q, C_1^r) \\ &\geq \delta_{TWED}(A_1^{p-1}, B_1^q) + \delta_{TWED}(A_1^p, A_1^{p-1}) + \delta_{TWED}(B_1^q, C_1^r) \quad (lemma\,1) \\ &\geq \delta_{TWED}(A_1^p, A_1^{p-1}) + \delta_{TWED}(A_1^{p-1}, C_1^r) \geq \delta_{TWED}(A_1^p, C_1^r) \quad (H4) \text{ applies twice.} \end{aligned}$$

9th Case: if $\begin{cases} \delta_{TWED}(A_1^p, B_1^q) = \delta_{TWED}(A_1^p, B_1^{q-1}) + \Gamma(\Lambda \rightarrow b'_q) \\ \delta_{TWED}(B_1^q, C_1^r) = \delta_{TWED}(B_1^{q-1}, C_1^r) + \Gamma(b_q \rightarrow \Lambda) \end{cases}$, then:

$$\begin{aligned} \Sigma &\geq \delta_{TWED}(A_1^p, B_1^{q-1}) + \delta_{TWED}(B_1^{q-1}, C_1^r) \\ &\geq \delta_{TWED}(A_1^p, C_1^r) \quad (H4) \text{ applies.} \end{aligned}$$

So property *P4* holds for all *m* in $\{0,..n\}$. By induction it holds for all m in N and so P4 holds for all (A_1^p, B_1^q, C_1^r) in $U \times U \times U$.□

Proof of theorem 2: δ_{TWED1} is a distance on the set U of finite discrete time series:

Similarly to the proof given for δ_{TWED}, P1, P2 and P3 are easily proved for δ_{TWED1}.

For the proof of P4, the triangle inequality, we need to establish a second lemma:

Lemma 2: For all A_1^p in U, $d(a_{p-1}, a_p) \geq \delta_{TWED1}(A_1^{p-1}, A_1^p)$

Proof: For all A_1^p in U,

$$d(a_{p-1}, a_p) = 0 + d(a_{p-1}, a_p) + 0 = \delta_{TWED1}(A_1^{p-1}, A_1^{p-1}) + d(a_p, a_{p-1}) + d(a_{p-1}, a_{p-1})$$

$$= \delta_{TWED1}(A_1^{p-1}, A_1^{p-1}) + \Gamma(a_p \rightarrow \Lambda) \geq \delta_{TWED1}(A_1^p, A_1^{p-1}) \square$$

The proof of P4 for δ_{TWED1} is then obtained by induction similarly to the way we prove P4 for δ_{TWED} using *lemma 2* instead of *lemma 1* $\square$