Sequence Matching Capable of Excluding Outliers

Longin Jan Latecki, Suzan Koknar-Tezel, Qiang Wang, and Vasileios Megalooikonomou
Temple University
Dept. of Computer and Information Sciences
Temple University, Philadelphia, PA 19094, USA
{latecki, tezel, wang32, vasilis}@temple.edu

ABSTRACT

We consider the problem of elastic matching of sequences of real numbers. Since both a query and a target sequence may be noisy, i.e., contain some outlier elements, it is desirable to exclude the outlier elements from matching in order to obtain a robust matching performance. Moreover, in many applications like shape alignment or stereo correspondence it is also desirable to have a one-to-one and onto correspondence (bijection) between the remaining elements. We propose an algorithm that determines the optimal subsequence bijection (OSB) of a query and target sequence. The OSB is efficiently computed since we map the problem's solution to a cheapest path in a DAG (directed acyclic graph). We obtained excellent results on standard benchmark time series datasets. We compared OSB to Dynamic Time Warping (DTW) with and without warping window. We do not claim that OSB is always superior to DTW. However, our results demonstrate that skipping outlier elements as done by OSB can significantly improve matching results for many real datasets. Moreover, OSB is particularly suitable for partial matching. We applied it to the object recognition problem when only parts of contours are given. We obtained sequences representing shapes by representing object contours as sequences of curvatures.

Categories and Subject Descriptors

I.5.3 [**Pattern Recogntion**]: Clustering – *similarity measures,* H.3.1 [**Information Storage and Retrieval**]: Content analysis and indexing – *indexing methods*

General Terms

Algorithms, Experimentation, Theory.

Keywords

Time series similarity, sequence similarity, Dynamic Time Warping, partial sequence matching, object recognition.

W. on Time Series Classification at ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Jose, Aug. 2007

1. INTRODUCTION

Sequences of real numbers are commonly used in all research fields. Due to historical reasons, they are also called time series in the data mining community. Time series are a ubiquitous and increasingly prevalent type of data, therefore, there has been much research effort devoted to time series data mining in recent years. Many data mining algorithms have similarity measurements of sequences at their core. Examples include motif discovery [3], anomaly detection [8], rule discovery [7], classification [15] and clustering [1]. In this paper we deal with computation of time series distances based on elastic matching. If two sequences of real numbers of equal length n are to be compared, the simplest way is to treat them as vectors in Rn, and compute their Euclidean distance. This approach is based on the assumption that both sequences are well aligned, i.e., that corresponding vector coordinates represent corresponding sequence elements. In many applications, e.g., speech recognition, this assumption is not satisfied. Therefore, as many researchers have mentioned in their work [7,13,15], the Euclidean distance is not always the optimal distance measure for similarity searches of sequences.

To solve the problem of alignment of sequences, Dynamic Time Warping (DTW) [2,17] has been used. The DTW distance between two sequences is the sum of distances of their corresponding elements. Dynamic programming is used to find corresponding elements so that this distance is minimal. The DTW distance has been shown to be superior to the Euclidean distance in many cases [1,4,9,19]. However, DTW is particularly sensitive to outliers, since it is not able to skip any elements of the sequences. In DTW, each element of the query sequence must correspond to some element of the target sequence and vice versa. Thus, the optimal correspondence computed by DTW is a relation on the set of indices of both sequences, i.e., a one-to-many and many-to-one mapping. The fact that outlier elements must participate in the correspondence optimized by DTW often leads to an incorrect correspondence of other sequence elements. This fact is illustrated in Fig. 1(a), where the query sequence on top has two outlier elements (spikes), and the target sequence (at the bottom) has one outlier element. The correspondence of the elements is illustrated with straight lines. Observe that the outliers corrupt the correspondence computed by DTW. In particular, this is reflected in the fact that single elements of one sequence correspond to a large number of elements of the other sequence. For comparison, the result of the proposed Optimal Subsequence Bijection (OSB) is shown in Fig. 1(b)

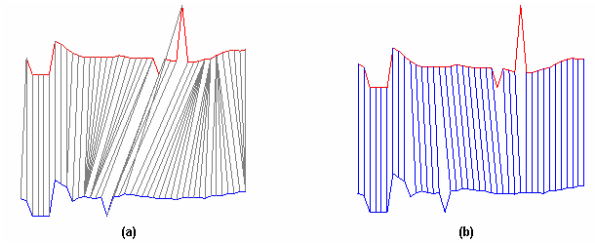


Figure 1. Observe how outliers corrupt the result of DTW in (a). The correspondence obtained by the proposed OSB is shown in (b). The two sequences represent parts of planar contours of two different but very similar corpora callosa. The sequences representing complete contours are shown in Fig. 2(a).

The Longest Common Subsequence (LCSS) measure has been used in time series [5,18] to deal with the alignment and outlier problems. Given a query and a target series, LCSS determines their longest common subsequence, i.e., LCSS finds subsequences of the query and target that best correspond to each other. The distance is based on the ratio between the length of longest common subsequence and the length of the whole sequence. The subsequence does not need to consist of consecutive points, the order of points is not rearranged, and some points can remain unmatched. However, when LCSS is applied to sequences of numeric values, one needs to set a threshold that determines when values of corresponding points are treated as equal [18]. The performance of LCSS depends heavily on the correct setting of this threshold, which may be a particularly difficult problem for many applications.

The proposed OSB computes the distance value between two sequences based directly on the distances of corresponding elements, just as DTW does, and it allows the ignoring of outlier points on both the query and target sequences to find the best match, just as LCSS does.

The main difference between DTW and OSB is that OSB can skip outlier elements of the query and target sequences when computing the correspondence while DTW requires that every element of both the query and target sequences be matched. This makes the performance of OSB robust in the presence of outliers. Moreover, OSB defines a bijection on the matched subsequences, which means that we have a one-to-one correspondence of the matched elements.

The main difference between LCSS and OSB is that LCSS optimizes the length of the longest common subsequence and requires a distance threshold, while OSB optimizes directly the sum of distances of corresponding elements. Moreover, the OSB penalty for skipping consecutive elements is proportional to the number of elements skipped, thus skipping one outlier costs less than skipping a consecutive subsequence of several elements. There is no penalty for skipping elements in LCSS, which often leads to accidental matches. This is illustrated in Fig. 2. The query sequence (shown as the top sequence in red) is the same in (a) and (b). It contains 4 outliers shown as spikes. While the query sequence is very similar to the target sequence (in blue) in Fig. 2(a), it is not the case in Fig. 2(b), where the target sequence has a subsequence of a completely different shape in the middle part. However, using the same distance threshold, in both cases LCSS finds a common subsequence of length 73. The incorrect correspondence in Fig. 2(b) is possible, since LCSS is able to skip (jump over) the whole subsequence in the middle part of the target sequence.

The main difference between OSB and assignment algorithms, like the Hungarian algorithm [10,14], is that the Hungarian algorithm, as well as other assignment algorithms, do not preserve the order of sequences.

The paper is organized as follows. In Section 2 we introduce our new method OSB together with its underlying theoretical framework. In Section 3 we propose a method to automatically compute the main parameter for OSB which is the cost of jumping over an element of the query and/or target sequence. In section 4 we discuss the time complexity of OSB. Finally, in Section 5 we present the experimental evaluation of the performance of OSB on two types of datasets: benchmark time series datasets and a shape contour dataset. In particular, we show that it outperforms DTW, LCSS, and the Euclidean distance..

2. OPTIMAL SUBSEQUENCE BIJECTION

The new algorithm, called ***Optimal Subsequence Bijection*** (***OSB***), works for the elastic matching of two sequences of different lengths m and n:

$$a = (a_1, ..., a_m) \text{ and } b = (b_1, ..., b_n).$$

The goal of OSB is to find subsequences a' of a and b' of b such that a' best matches b'. Skipping (not matching) some elements of a and b is necessary because both sequences may contain some outlier elements. However, skipping too many elements of either sequence increases the chance of accidental matches. To prevent this from happening, we introduce a penalty for skipping. We call this penalty the ***jump cost*** and denote it with ***C***. We describe one method to compute the jump cost in Section 3.

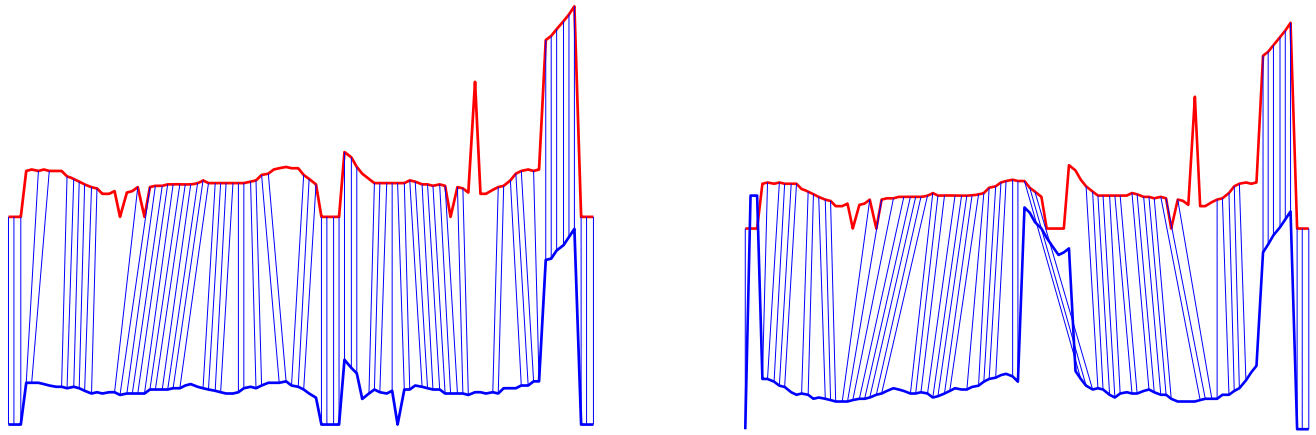
We assume that the distance function d that can be used to compute the dissimilarity value between any elements of sequences a and b, i.e., $d(a_i, b_j)$, is given for $(i,j) \in \{1,\ldots,m\} \times \{1,\ldots,n\}$. We do not have any restrictions on the distance function d, and therefore, any distance function is possible. Usually, for sequences of real numbers we simply have the distance $d(a_i, b_j) = (a_i - b_j)^2$, which is also the case for our experimental results reported in Section 5.

Our goal is to select a subsequence a' of the query sequence a by skipping some outlier elements of a, so that each element of a' matches some element of b in an order preserving manner with possibly skipping some outliers in b as well. The optimal correspondence is obtained by optimizing the balance between the dissimilarity of a' to its image subsequence of b and the penalties of skipping elements of a and of b.

The optimal correspondence can be found with a shortest path algorithm on a DAG (directed acyclic graph). The nodes of the DAG are all index pairs $(i,j) \in \{1,\ldots,m\} \times \{1,\ldots,n\}$ and the edge cost w is defined as

$$w((i,j),(k,l)) = \begin{cases} \sqrt{(k-i-1)+(l-j-1)} \cdot C + d(a_k, b_l) & \text{if } i < k \wedge j < l \\ \infty & \text{otherwise} \end{cases}$$

Thus, the cost of an edge from (i,j) to (k,l) is the Euclidean distance of vertices (i,j) and (k,l) in the matrix $\{1,\ldots,m\} \times \{1,\ldots,n\}$



(a) **(b)**

Figure 2. There is no penalty for skipping elements in LCSS, which often leads to accidental matches. The query sequence (top) is similar to the target sequence in (a) but not to the target sequence in (b). However, LCSS determines the length of the optimal subsequences as 73 in both cases. The three sequences represent planar contours of three different corpora callosa.

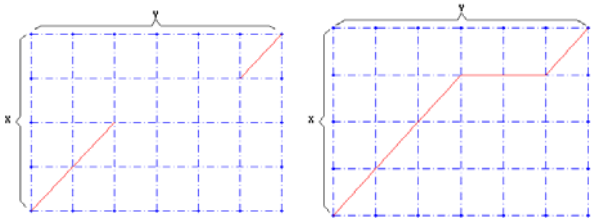
times the jump cost plus the dissimilarity measure of elements a_k and b_l.

OSB can be viewed as extension of DTW. To see this, observe that the edge weight *dtw* for DTW is defined as

$$dtw((i,j),(k,l)) = \begin{cases} d(a_k,b_l) & \text{if } i<k \wedge j<l \wedge ((k-i)+(l-j)=1 \vee 2) \\ \infty & \text{otherwise} \end{cases}$$

This means that if i maps to j, then either k=i maps to l=j+1 or k=i+1 maps to l=j+1 or k=i+1 maps to l=j in the DTW correspondence. Thus, in comparison to DTW, OSB allows penalized jumps. We view as our main contribution the definition of the edge cost w in DAGs with a jump penalty that includes the jump cost defined in Section 3.

The edge cost can be easily extended to impose a warping window constraint, i.e., we can limit the number of elements that can be jumped over in one step by setting a maximal value for the index differences k–i–1 and l–j–1.



(a) **(b)**

Figure 3. Time series alignment with (a) OSB and (b) DTW.

We illustrate the proposed method on a simple example. Figure 3(a) demonstrates the correspondence found with OSB as a shortest path for two sequences

$$X = (1, 2, 8, 6, 8) \text{ and } Y = (1, 2, 9, 3, 3, 5, 9)$$

with the distance between two elements being the squared difference. The corresponding sequence indices computed by OSB (not the values) are (1,1), (2,2), (3,3), (4,6), (5,7). Observe that the outliers Y_4=Y_5=3 are skipped. The automatically computed jump cost is C=1.15 (see Section 3). For comparison, DTW yields the sequence indices (1,1), (2,2), (3,3), (4,4), (4, 5), (4, 6), (5,7). Thus, X_4=6 is forced to match to the outliers Y_4=Y_5=3 in addition to the correct match of X_4 = 6 to Y_6=5.

With the output of OSB we obtain a ***correspondence*** defined as a mapping on sequence indices

$$f: \{1,\ldots,m\} \rightarrow \{1,\ldots,n\}$$

that is a monotonic injection, i.e., $f(i) < f(i+1)$ for i $\in$ {1, …,m}. The sets of indices (i_k) and ($f(i_k)$) $i_k \in \{1, \ldots, m\}$ define the subsequences a' of a and b' of b, such that f restricted to (i_k) is a bijection. This explains the phrase "subsequence bijection" in the *Optimal Subsequence Bijection* (*OSB*).

To illustrate the benefit of the proposed one-to-one matching with skipping the outliers, consider again the matching of the two sequences shown in Fig. 1. The query sequence on top has two outlier elements (spikes), and the target sequence (at the bottom) has one outlier element. The proposed OSB method is able to skip them as shown in Fig. 1(b). For comparison, the correspondence obtained with the Dynamic Time Warping (DTW) shown in Fig. 1(a) is significantly corrupted by the outliers.

3. PENALTY FOR SKIPPING ELEMENTS

In this section we describe how to determine the jump cost C. As we stated in Section 2, while in most applications $d(a_i, b_j)$ is given for $(i,j) \in \{1, \ldots,m\} \times \{1, \ldots,n\}$, the jump cost should be carefully selected. We propose to compute the jump cost in two phases.

Let B be a set of all target sequences to which query sequence a should be compared. In the first phase, the query sequence a is compared to a target sequence b $\in$ B, and we define

$$C(a,b) = \underset{i}{\text{mean}}(\min_j(d(a_i,b_j))) + \underset{i}{\text{std}}(\min_j(d(a_i,b_j)))$$

Thus, for every element a_i we find the closest element b_j, and then we take the mean plus one standard deviation (std) of the

Table 1. Classification results on benchmark datasets from [16].

Name	Number of Classes	Size of Training Set	Size of Testing Set	Time Series Length	Euclidean Distance Accuracy	DTW with Best Warping Window (r)	DTW without Warping Window	OSB
Synthetic Control	6	300	300	60	0.120	0.017 (6)	0.007	0.047
Gun-point	2	50	150	150	0.087	0.087 (0)	0.093	0.040
CBF	3	30	900	128	0.148	0.004 (11)	0.003	0.012
Face(all)	14	560	1690	131	0.286	0.192 (3)	0.192	0.156
OSU Leaf	6	200	242	427	0.483	0.384 (7)	0.409	0.417
Swedish Leaf	15	500	625	128	0.213	0.157 (2)	0.210	0.136
50Words	50	450	455	270	0.369	0.242 (6)	0.310	0.301
Trace	4	100	100	275	0.240	0.010 (3)	0.000	0.200
Two Patterns	4	1000	4000	128	0.090	0.002 (4)	0.000	0.000
Wafer	2	1000	6174	152	0.005	0.005 (1)	0.020	0.002
Face (four)	4	24	88	350	0.216	0.114 (2)	0.170	0.057
Lightening-2	2	60	61	637	0.246	0.131 (6)	0.131	0.148
Lightning-7	7	70	73	319	0.425	0.288 (5)	0.274	0.369
ECG	2	100	100	96	0.120	0.120 (0)	0.230	0.130
Adiac	37	390	391	176	0.389	0.391 (3)	0.396	0.386
Fish	7	175	175	463	0.217	0.160 (4)	0.167	0.120
Beef	5	30	30	470	0.467	0.467	0.500	0.467
Coffee	2	28	28	286	0.250	0.179	0.179	0.250
OliveOil	4	30	30	570	0.133	0.167	0.133	0.133

distances to the closest elements. So, for example, if sequences a and b are similar with the exception of one outlier element, call it a_k, then for every a_i with $i \neq k$ we find an element b_j with a small distance $d(a_i, b_j)$. Consequently, $C(a,b)$ will be small, so that the distance to the closest element in b for a_k will be greater than $C(a,b)$, and the element a_k will be excluded from matching by this relatively small penalty, i.e., we will jump over it.

In order to ensure a fair comparison of sequence a to all target sequences $b \in B$, we need to fix the jump cost for all the comparisons. This is obtained in the second phase by simply taking the mean of all the jump costs:

$$C(a) = \text{mean}\{C(a,b) : b \in B\}.$$

4. TIME COMPLEXITY

It is well known that the time complexity of DTW is O(mn) for two sequences with m and n elements respectively. Since OSB considers all possible jumps, its time complexity is O(m^2n^2).

It is also well known that in practice DTW is run with a warping window restriction, which means that element a_i from the query sequence can only match to an element b_j from the target sequence if $| i - j |$ is smaller than a predefined constant, say r. The time complexity of DTW with this warping window restriction is O(m). We can also impose the warping window restriction on OSB. In addition, we can limit the number of elements that can be jumped over in one step by setting maximal values for the index differences $k–i–1$ and $l–j–1$ in formula (1). With these restrictions, the time complexity of OSB is also O(m).

As our experimental results in Section 5 demonstrate, these restrictions do not influence the matching results of OSB. In all experimental results, we set all three index differences to be bounded by five, i.e., $| i - j | < 5$, $k–i–1 < 5$, and $l–j–1 < 5$. In contrast, the performance of DTW is evaluated with respect to the best possible warping window restriction.

5. EXPERIMENTAL RESULTS

We tested our approach on the benchmark time series datasets published on the UCR Time Series Classification/Clustering Page [21] and on the MPEG-7 shape dataset [22]. These datasets are online to serve the data mining/machine learning community as an effort to encourage reproducible research for time series classification and clustering, and for shape matching. Section 5.1 will discuss the results for the UCR time series datasets, and Section 5.2 will discuss the results for partial sequence similarity on the MPEG-7 dataset.

In addition, we entered the *Workshop and Challenge on Time Series Classification at SIGKDD 2007* [20]. In this competition, we were given twenty time series datasets of unknown taxonomy and we had 24 hours to run our algorithm, classify the data, and submit the results. Due to technical problems, we could finish only twelve of the twenty datasets, but using OSB, we had the highest accuracy of all competitors on three of the datasets.

5.1 The UCR Time Series Datasets Results

We were able to download 19 datasets from [21] that came from various application domains. The description of these datasets can be found on the UCR webpage.

The table published on [21] reports 1-Nearest Neighbor (1-NN) classification results for three methods: Euclidean Distance, DTW (Dynamic Time Warping) without warping window, and DTW with best scoring warping window. In Table 1, we copied the results from [21] and compared them to the results obtained with OSB (reported in the last column). We followed exactly the same 1-NN classification procedure as reported on [21]. Each dataset is divided into two parts; one part for training and the other for testing. The division is fixed by the authors of each dataset. Only the class labels of the training data are known to the classification algorithm. Every time series in the testing dataset is taken as a query. After calculating the distance between the test queries and each of the training time series, the nearest neighbor for each test query is found in the target training set and this target's class label is assigned to the query. The overall accuracy is reported as the percentage of misclassified queries (i.e., the smaller the value, the better is the result).

To summarize, OSB yields the best results on 10 datasets, followed by DTW without warping window with best results on 8 datasets, followed by DTW with best scoring warping window with best results on 6 datasets. The Euclidean Distance obtained the best score on only two datasets. The total count of winners is 26, since in the case of ties we declared all the equally scoring methods the best (there are ties on six datasets). The best scoring results for each dataset are highlighted in yellow in Table 1.

We do not claim that OSB is always superior to DTW, we claim only that skipping outlier elements as done by OSB can significantly improve matching results for many real datasets, and this fact is clearly demonstrated by the results in Table 1.

There are two data sets (Synthetic Control and Trace) on which OSB performed significantly worse than DTW. These are datasets without any outlier elements (according to our visual inspection), in which case the skipping outlier behavior of OSB is a clear disadvantage.

Note that the results for DTW with warping window are obtained in a supervised setting in that the best warping window size is obtained by training on part of the dataset. The best warping window size for DTW is provided as *r* in Table 1.

Although in OSB experiments, we also utilize a warping window for the concern of computing efficiency, the OSB warping window was set to 5 in all the experiments. Tuning the warping window size in a way similar to DTW would clearly further improve the performance of our method. We did not perform any training of the warping window in order to demonstrate that such tuning is not necessary for robust performance of our method.

The jump cost of OSB is computed as described in Section 3. Again we could use the training part of the datasets to learn the best performing jump cost. However, we did not do this, since our main goal is to demonstrate the advantages of setting only the jump cost of the proposed method.

No results on the test datasets are reported for LCSS. This is probably due to the fact that these are sequences of numeric values, which means that it is necessary to set a threshold to determine when their elements are treated as equal [18]. As stated in the introduction, the performance of LCSS depends heavily on the correct setting of this threshold, and no systematic method is known for this. However, we perform comparisons to LCSS in the next subsection.

5.2 The MPEG7 Dataset Results

The goal of this section is to test OSB's ability as a measure for partial sequence similarity. Our query sequences represent significant contour parts of shapes in the MPEG-7 Shape 1 Part B dataset. The MPEG-7 dataset is a standard dataset for testing shape-matching algorithms [11]. The dataset consists of 70 object classes (e.g. "Birds") and within each class there are 20 shapes for a total of 1400 shapes. Each shape is represented with 100 equidistant sample points on the contour.

We manually selected 10 query sequences as contour segments representing shapes from 10 different classes. They are represented as the black parts of the contours in the first column in Fig. 4. Their lengths range from 30 to 57 points. We then converted both the target shapes and the query segments into sequences by calculating the curvature of each point with respect to its 5 neighbors on each side. This particular transformation makes the sequence representation invariant to rotation and scale changes, which is particularly relevant for finding similar contour parts. In other words, the shape of a cell phone with its antenna pointing up can still match with the same cell phone shape scaled and rotated so that its antenna is now pointing down. This curvature conversion was done for each target shape (yielding target sequences of length 100), and on each query segment (giving query sequences with lengths 10 less than the original number of points for each segment).

We ran the OSB algorithm for each query against each target to find the optimal subsequence between each query segment and target shape. For each query/target pair we used a step of 5 points to move the starting point of the query. First we compared the query against target elements 1 - 100. Then we compared the query against target elements 6 - 105 (doing a circular shift of the elements), then against target elements 11 - 110, and so on (incrementing by 5).

The final score for a query/target pair is the dissimilarity value of the best matching starting value. Since we have only 10 query sequences, and we want a statistically significant comparison, we used 1, 5, 10 and 20 nearest neighbors' classification. For each query we looked at the one/five/ten/twenty targets that yielded the smallest distance measure among 1400 shapes and calculated the rate of correct classification, i.e., how many of the target contours in the top one/five/ten/twenty were from the same class as the query segment. The accuracy reported is the recall rate (sometimes also called the retrieval rate), i.e., of all the possibly

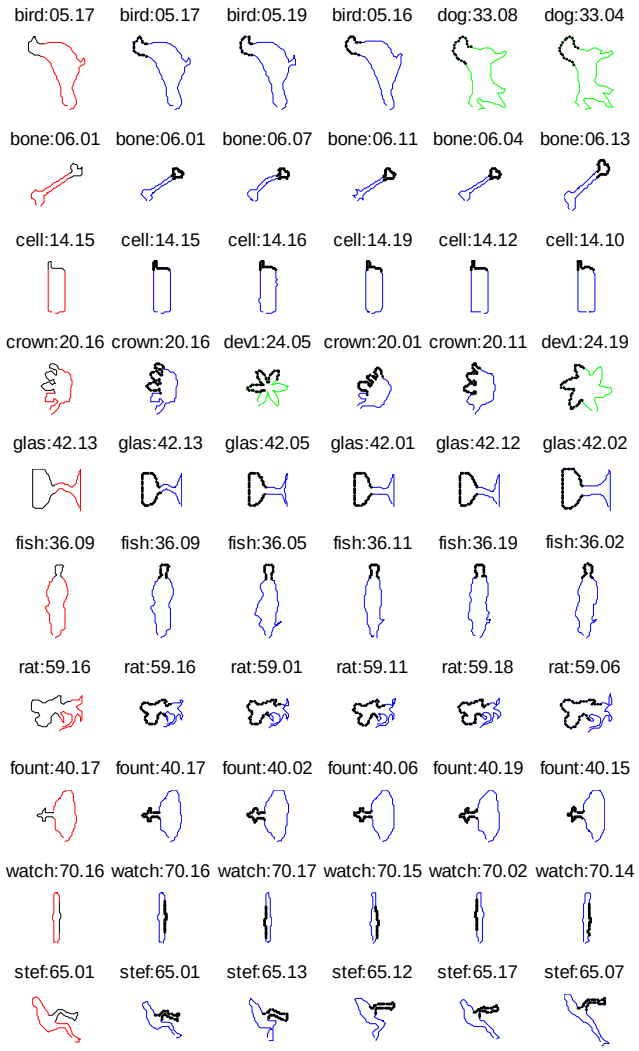


Figure 4. OSB results on the MPEG-7 dataset for our 10 queries. Each contour is identified by *label*:*class*:*id*. The first column shows the query part (in black) in its original contour (red). Columns 2 through 6 are the top five matches out of the 1400 target shapes. The black points on the targets indicate the corresponding points as computed by OSB.

correct matches, how many were actually found in the top one/five/ten/twenty.

For comparison, we used the same query and target sequences in three other algorithms: DTW, DTWCW (DTW with Correspondence Window, described below), and LCSS. As shown in Table 2, OSB performed significantly better than any of the other algorithms. OSB was the only algorithm to achieve 100% recall rate in the 1NN classification; it had a 92% recall rate in the 5NN classification.

The extremely low performance of DTW is due to the fact that we did not use any window restriction, e.g., if the query sequence is of length 20, we still matched it to the target sequences of length hundred 1-100, 6 – 105, and so on. In order to eliminate this problem, we also considered DTW with Correspondence Window (DTWCW), where we matched the query sequence against a target sequence of the same length, i.e., the correspondence window has the same length as the query sequence. For example, if the query sequence is of length 20, we matched it to subsequences of the target with indices 1-20, 6-25, and so on. The results for DTWCW improved dramatically, but still did not match the performance of OSB. It may be possible to further improve the performance of DTW, by varying the size of the correspondence window, but this would add one more parameter and increase the time complexity (from the linear time of DTW with a warping window to quadratic time if the size of the correspondence window is added). The excellent performance of OSB illustrates that such a window size parameter is not needed, since OSB computes a bijective embedding of the query sequence. This fact clearly demonstrates that it is more suitable for partial sequence similarity.

Table 2. Recall rates on the MPEG-7 dataset.

	OSB	DTW	DTWCW	LCSS
1NN	100%	0%	90%	90%
5NN	92%	2%	72%	42%
10NN	84%	2%	67%	34%
20NN	67%	3%	59%	26%

For DTW, DTWCS, and OSB we used the same warping window of size five. Thus, all three algorithms have the same linear time complexity. LCSS requires a threshold that decides whether two sequence elements are matching or not. We used LCSS with various thresholds and selected the results that gave the best performance. (It was 0.3 times the value of the smallest element.) For OSB, we used a fixed jump cost of 10 for each query, which was determined experimentally. (OSB was not sensitive to the value of the jump cost, since we obtained the same results for a large range of jump cost values.)

Fig. 4 shows the retrieval results of OSB. It can be observed that the performance of OSB is actually better than is reflected by the classification based on class labels. For example, the query contour segment representing a bird head in the first row matches to dog heads, which actually have very similar shape.

The best match for each segment (shown in the second column) is the shape from which we took the query segment. This holds for all the queries and is not a trivial result. Because of the step of 5 we used to position the starting point of the query sequence, the query and target sequences are not identical. In fact, DTW was not able to find any of the queries' original shapes, and DTWCW and LCSS both could find only 9 of the 10.

To summarize, the OSB algorithm performed significantly better than DTW, DTWCW, and LCSS. This difference clearly shows that the ability of skipping outlier elements is essential in partial sequence similarity.

6. CONCLUSIONS AND FUTURE WORK

The proposed sequence matching method (OSB) can be viewed as an extension of DTW. In comparison to DTW, OSB allows penalized skipping of outlier elements, which we view as one of our main contributions.

As demonstrated in the test results in Section 5.1, the ability of skipping outlier elements leads to improved retrieval performance for most datasets. However, for some datasets without significant outliers it may lead to reduced retrieval performance.

As demonstrated in the test results in Section 5.2, when dealing with partial shape similarity in the presence of noise, the ability of skipping outlier elements is essential. Here the performance of OSB in the 5NN classification was 90 percentage points higher than the performance of DTW, 20 percentage points higher than that of DTWCW, and 50 percentage points higher than that of LCSS.

7. ACKNOWLEDGMENTS

We would like to thank Eamonn Keogh for maintaining the UCR Time Series Classification/Clustering Page. This work was supported in part by the NIH under grant No. R01 MH068066-03 (funded by NIMH, NINDS and NIA), by the NSF under Grant Nos. IIS-0534929 and IIS-0237921, and by DOE Grant No. DE-FG52-06NA27508.

8. REFERENCES

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

[2] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proc. AAAI-94 W. on Knowledge Discovery and Databases*, pages 229–248, 1994.

[3] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proc. ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining*, Washington, 2003.

[4] S. Chu, E. Keogh, D. Hart, and M. Pazzani. Iterative deepening dynamic time warping for time series. *In Proc. SIAM Int. Conf. on Data Mining*, 2002.

[5] G. Das, D. Gunopoulos, and H. Mannila. Finding similar timie series. In *Proc. 1st PKDD Symposium*, pages 88–100, 1997.

[6] D. Goldin and P. Kanellakis. On similarity queries for timeseries data: Constraint specification and implementation. In *Int. Conf. on the Principles and Practice of Constraint Programming*, pages 137–153, 1995.

[7] F. Hoeppner. Discovery of temporal patterns. learning rules about the qualitative behavior of time series. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 192–203, Freiburg, 2001.

[8] E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Seattle, 2004.

[9] G. Kollios, M. Vlachos, and D. Gunopoulos. Discovering similar multidimensional trajectories. In *Proc. Int. Conf. On Data Engineering*, p. 673–684, 2002.

[10] H. W. Kuhn, "The Hungarian Method for the assignment problem", *Naval Research Logistic Quarterly*, 2:83-97, 1955.

[11] L. J. Latecki, R. Lakämper, and U. Eckhardt: Shape Descriptors for Non-rigid Shapes with a Single Closed Contour. *Proc. of IEEE CVPR*, pp. 424-429, June 2000.

[12] L. J. Latecki, V. Megalooikonomou, Q. Wang , R. Lakaemper, C. A. Ratanamahatana, and E. Keogh: Partial Elastic Matching of Time Series. *IEEE Int. Conf. on Data Mining (ICDM)*, pp. 701-704, Houston, TX, 2005.

[13] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *Proc. IEEE Int. Conf. on Data Engineering* (ICDE05), pages 668–679, Tokyo, 2005.

[14] J. Munkres, "Algorithms for the Assignment and Transportation Problems", *J. of the Society of Industrial and Applied Mathematics*, 5(1):32-38, 1957.

[15] D. Rafiei. On similarity-based queries for time series data. In *Proc. Int. Conf. on Data Engineering*, pages 410–417, Sydney, 1999.

[16] C. A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *W. on Mining Temporal and Sequential Data*, Seattle, 2004.

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

[18] M. Vlachos, M. Hadjieleftheriou, D. Gunopoulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proc. of ACM SIGKDD*, pages 216–225, Washington, 2003.

[19] B. Yi, K. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. Int. Conf. on Data Engineering*, pages 23–27, 1998.

[20] E. Keogh, C. Shelton, and F. Moerchen. Workshop and Challenge on Time Series Classification at SIGKDD 2007. http://www.cs.ucr.edu/~eamonn/SIGKDD2007TimeSeries.html

[21] E. Keogh, X. Xi, Li Wei, and C. Ratanamahatana. UCR Time Series Classification/Clustering Page, http://www.cs.ucr.edu/~eamonn/time_series_data/

[22] L. J. Latecki, Shape Data for the MPEG-7 Core Experiment CE-Shape-1, http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz