Dynamic Time Warping Algorithm Review

Pavel Senin
Information and Computer Science Department
University of Hawaii at Manoa
Honolulu, USA
senin@hawaii.edu

December 2008

Contents

1 Introduction **3**

2 DTW Algorithm **3**

3 DTW Customization **9**

3.1 Step Function (Slope Constraint) 10

3.2 Weighting 11

3.3 Global path constraints 11

4 DTW Optimization **12**

4.1 Scaling 13

5 Query-By-Example **13**

6 Time-series indexing **16**

7 Software metrics application **16**

8 Future work **17**

9 Notes **19**

1 Introduction

The Dynamic Time Warping algorithm (DTW) is a well-known algorithm in many areas. While first introduced in 60s [1] and extensively explored in 70s by application to the speech recognition [2], [3] it is currently used in many areas: handwriting and online signature matching [4] [5], sign language recognition [6] and gestures recognition [7] [6], data mining and time series clustering (time series databases search) [8] [9] [10] [11] [12] [13], computer vision and computer animation [14], surveillance [15], protein sequence alignment and chemical engineering [16], music and signal processing [17] [14] [18]. This work aims to benefit the software metrics analysis through the application of the Dynamic Time Warping algorithm to the software development telemetry data.

In this report we introduce and discuss the "naive" DTW and idea behind it in the section 2. The DTW customization through local and global parameters summarized in the section 3. The DTW speedup through the scaling discussed in the section 4. In the section 5 we will show how DTW could be employed to identify similar to query subsequences in the long data streams. Finally we will show the current progress in the DTW application to the software development telemetry data in 7 and outline future directions in 8.

2 DTW Algorithm

DTW algorithm has earned its popularity by being extremely efficient as the time-series similarity measure which minimizes the effects of shifting and distortion in time by allowing "elastic" transformation of time series in order to detect similar shapes with different phases. Given two time series $X = (x_1, x_2, ...x_N),\ N \in \mathbb{N}$ and $Y = (y_1, y_2, ...y_M),\ M \in \mathbb{N}$ represented by the sequences of values (or curves represented by the sequences of vertices) DTW yields optimal solution in the $O(MN)$ time which could be improved

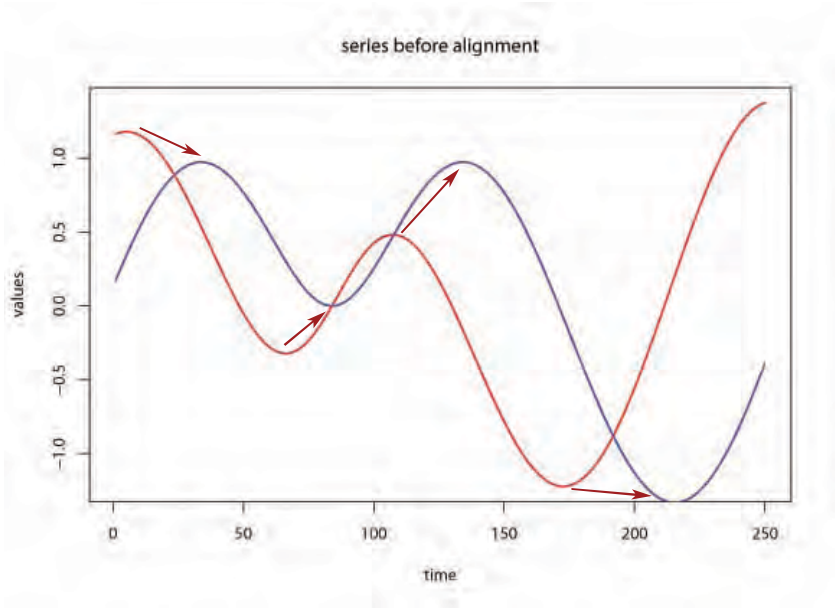

Figure 1: Raw time series, arrows show the desirable points of alignment.

further through different techniques such as multi-scaling [17] [19]. The only restriction placed on the data sequences is that they should be sampled at equidistant points in time (this problem can be resolved by re-sampling).

If sequences are taking values from some feature space Φ than in order to compare two different sequences $X,\ Y \in \Phi$ one needs to use the local distance measure which is defined to be a function:

$$d : \Phi \times \Phi \rightarrow \mathbb{R} \geq 0 \tag{1}$$

Intuitively d has a small value when sequences are similar and large value if they are different. Since the Dynamic Programming algorithm lies in the core of DTW it is common to call this **distance function** the "**cost function**" and the task of optimal alignment of the sequences becoming the task of arranging all sequence points by minimizing the cost function (or distance).

Algorithm starts by building the distance matrix $C \in \mathbb{R}^{N \times M}$ representing all pairwise distances between X and Y. This distance matrix called the

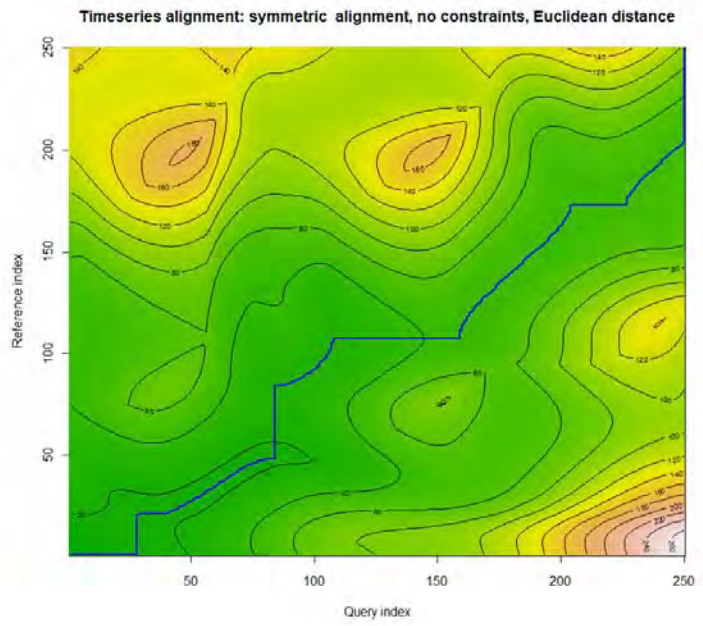

Figure 2: Time series alignment, cost matrix heatmap.

local cost matrix for the alignment of two sequences X and Y:

$$C_l \in \mathbb{R}^{N \times M} \ : \ c_{i,j} = \|x_i - y_j\| \, , \ i \in [1:N], \ j \in [1:M] \tag{2}$$

Once the local cost matrix built, the algorithm finds the **alignment path** which runs through the low-cost areas - "valleys" on the cost matrix, Figure 2. This alignment path (or **warping path**, or **warping function**) defines the correspondence of an element $x_i \in X$ to $y_j \in Y$ following the boundary condition which assigne first and last elements of X and Y to each other, Figure 3.

Formally speaking, the alignment path built by DTW is a sequence of points $p = (p_1, p_2, ..., p_K)$ with $p_l = (p_i, p_j) \in [1:N] \times [1:M]$ for $l \in [1:K]$

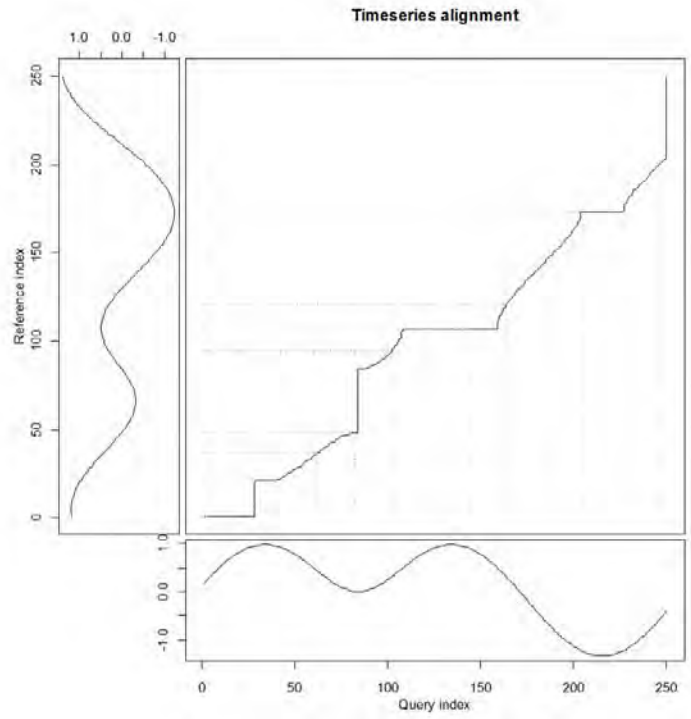

Figure 3: The optimal warping path aligning time series from the Figure 1.

which must satisfy to the following criteria:

1. **Boundary condition**: $p_1 = (1, 1)$ and $p_K = (N, M)$. The starting and ending points of the warping path must be the first and the last points of aligned sequences.

2. **Monotonicity condition**: $n_1 \leq n_2 \leq ... \leq n_K$ and $m_1 \leq m_2 \leq ... \leq m_K$. This condition preserves the time-ordering of points.

3. **Step size condition**: this criteria limits the warping path from long jumps (shifts in time) while aligning sequences. While this condition will be discussed in greater details in the Section 3, for now will use the basic step size condition formulated as $p_{l+1} - p_l \in \{(1, 1), (1, 0), (0, 1)\}$.

The **cost function** associated with a warping path computed with respect to the local cost matrix (which represents all pairwise distances) will be:

$$c_p(X,Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}) \tag{3}$$

The warping path which has a minimal cost associated with alignment called the **optimal warping path**. We will call this path P^*.

By following the optimal warping path definition in order to find one, we need to test every possible warping path between X and Y which could be computationally challenging due to the exponential growth of the number of optimal paths as the lengths of X and Y grow linearly. To overcome this challenge, DTW employs the Dynamic Programming - based algorithm with complexity only $O(MN)$.

The Dynamic Programming part of DTW algorithm uses the **DTW distance function**

$$DTW(X,Y) = c_{p^*}(X,Y) = min\left\{c_p(X,Y), p \in P^{N\times M}\right\} \tag{4}$$

where $P^{N\times M}$ is the set of all possible warping paths and builds the **accumulated cost matrix** or **global cost matrix** D which defined as follows:

1. First row: $D(1,j) = \sum_{k=1}^{j} c(x_1, y_k),\ j \in [1, M]$.
2. First column: $D(i,1) = \sum_{k=1}^{i} c(x_k, y_1),\ i \in [1, N]$.
3. All other elements: $D(i,j) = min\left\{D(i-1,j-1), D(i-1,j), D(i,j-1)\right\} + c(x_i, yj), i \in [1,N], j \in [1,M])$.

The time cost of building this matrix is $O(NM)$ which equals the cost of the following algorithm, where X and Y are the input time series and C is the local cost matrix representing all the pairwise distances between X and Y:

Algorithm 2.1 AccumulatedCostMatrix(X, Y, C)

```
1: n ← |X|
2: m ← |Y|
3: dtw[] ← new [n × m]
4: dtw(0, 0) ← 0
5: for i = 1; i ≤ n; j++ do
6:     dtw(i, 1) ← dtw(i − 1, 1) + c(i, 1)
7: end for
8: for j = 1; j ≤ m; j++ do
9:     dtw(1, j) ← dtw(1, j − 1) + c(1, j)
10: end for
11: for i = 1; i ≤ n; j++ do
12:     for j = 1; j ≤ m; j++ do
13:         dtw(i, j) ← c(i, j)+min {dtw(i − 1, j); dtw(i, j − 1); dtw(i − 1, j − 1)}
14:     end for
15: end for
16: return dtw
```

Once the accumulated cost matrix built the warping path could be found by the simple backtracking from the point $p_{end} = (M, N)$ to the $p_{start} = (1, 1)$ following the greedy strategy as described by the Algorithm 2.2:

Algorithm 2.2 OPTIMALWARPINGPATH(dtw)

```
1: path[] ← new array
2: i = rows(dtw)
3: j = columns(dtw)
4: while (i > 1) & (j > 1) do
5:   if i == 1 then
6:     j = j − 1
7:   else if j == 1 then
8:     i = i − 1
9:   else
10:    if dtw(i−1, j) == min{dtw(i − 1, j); dtw(i, j − 1); dtw(i − 1, j − 1)}
       then
11:      i = i − 1
12:    else if dtw(i, j−1) == min{dtw(i − 1, j); dtw(i, j − 1); dtw(i − 1, j − 1)}
       then
13:      j = j − 1
14:    else
15:      i = i − 1; j = j − 1
16:    end if
17:    path.add((i, j))
18:  end if
19: end while
20: return path
```

3 DTW Customization

In order to improve performance and customize the sensitivity of the "naive" Dynamic Time Warping algorithm various modification were proposed. This section outlines the major modifications such as the step size conditions, step

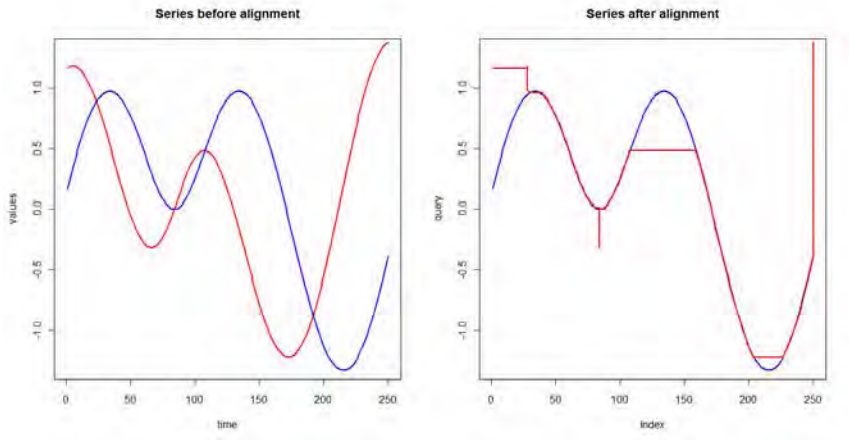

Figure 4: The optimal alignment of the time series from the Figure 1.

weighting and the global path constraints.

3.1 Step Function (Slope Constraint)

When there is no differences between the time-series, the warping path coincides with the diagonal line $i = j$, but as differences between time-series increase, the warping path deviates more from the diagonal line by matching similar time-axis fluctuations. While DTW finds the optimal alignment of the time-series, sometimes it tends to create an unrealistic correspondence between time-series features by aligning very short features from the one of the series to the long features on the second time-series. The feature-stretching phenomena could be seen at Figure 10: the red-colored time-series shifted to the right matching blue time series peak and collapsing low-amplitude features. In order to avoid such a phenomena the warping path is subject to constraints on the each step. This constraints implemented as the possible relations between several consecutive points on the warping path. For example after moving in the same direction say horizontal for k consecutive points warping path is not allowed to continue in the same direction before stepping l points in the diagonal direction, Figure 5.

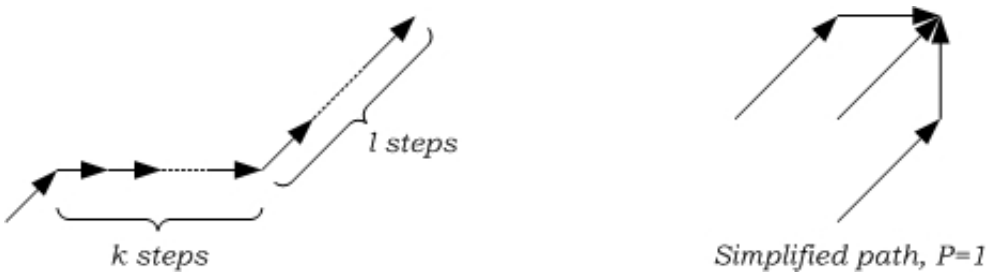

Figure 5: Step function constraints.

3.2 Weighting

Previously we defined the measure of distance between time series as the cost function (3), which essentially is a summation of pairwise distances between corresponding points at time-series X and Y. By adding the weights to the each of the distances based on the step direction we could penalize or favor certain types of point-to point correspondence. While Sakoe and Chiba in [3] report superiority of the symmetric weighting for the Japanese speech recognition, the idea behind weighting seems to be valid and interesting and we plan to evaluate the performance of asymmetric weighting when applied to the software development telemetry time-series.

3.3 Global path constraints

As we shown before, the computational cost of DTW algorithm is $O(NM)$ and algorithm requires a storage for two matrices of the size $N \times M$. In order to improve the computational cost and optimize the DTW sensitivity similarly to the step function constraints **global constraints** were introduced. The "Sakoe-Chiba band" appeared in [3] and "Itakura parallelogram" in [20], both global path constraints shown at Figure 7 and define the set of points available for DTW alignment only from the non-shaded regions.

The Sakoe-Chiba band runs along the main diagonal, $i = j$, and constraints the range of warping while specified by the fixed width $R \in \mathbb{N}$. The

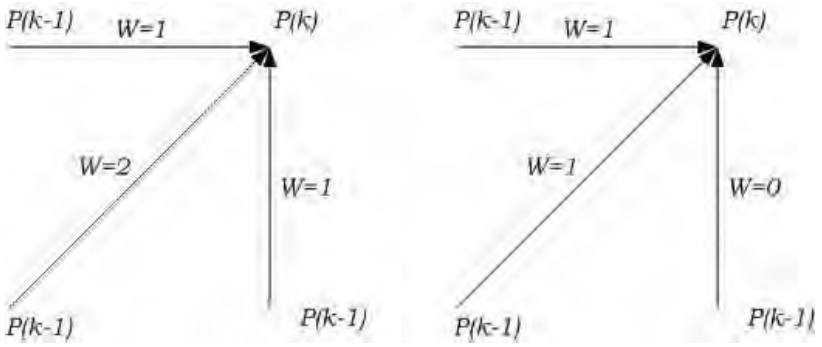

Figure 6: Weighting coefficients $w(k)$ for symmetric (left) and asymmetric (right) forms.

parameter R specifies the length of the vertical vector from the main diagonal to the upper boundary of the band and symmetrically the length of the horizontal vector from the main diagonal to the lower boundary of the band:

$$|i(k) - m| \leq R, \ |j(k) - m| \leq R, \tag{5}$$

Where m is the corresponding coordinate at the main diagonal and k indicates the k-th point of the warping path.

The Itakura parallelogram also constraints the warping range and specified by the $S \in \mathbb{R}_{>1}$ parameter.

4 DTW Optimization

While we have shown the reduction of the DTW algorithm computational cost in the section 3.3, another popular method of the DTW speed-up is the time-series approximation through the scaling as shown in [18] [21].

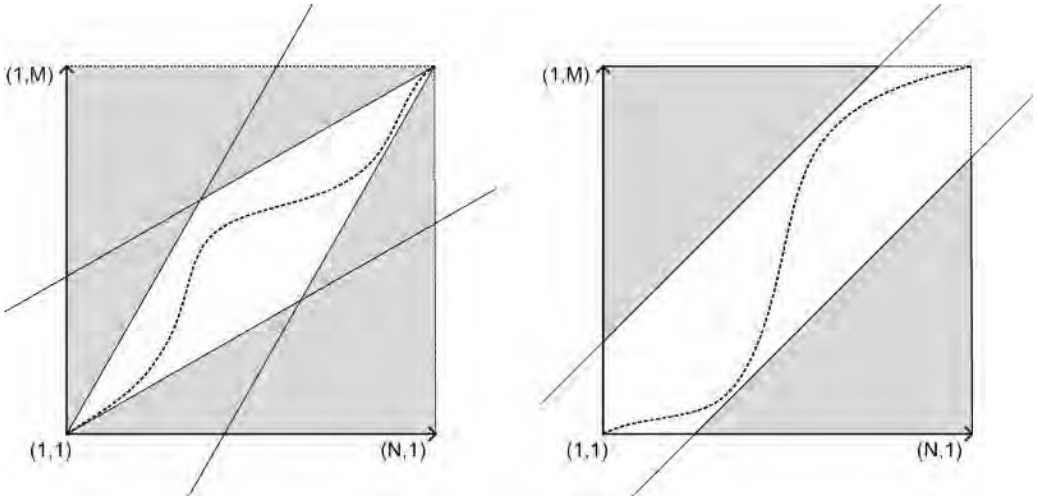

Figure 7: Global path constraints, left: Itakura parallelogram, right: Sakoe-Chiba band.

4.1 Scaling

The scaling is also known as the dimensionality reduction and aiming the reduction of the lengths N and M of the input time-series through coarsening. This could be accomplished through the downsampling of the low-pass filtered series, or by linear approximation, [21]. The only backdraw of such approach is that after reaching of the certain level of coarsening the alignment becomes completely useless [18].

5 Query-By-Example

So far we have discussed the DTW algorithm application for the time-series alignment and DTW-based similarity measure. In this section we will focus on the **"query-by-example"** (QBE) [22] or "query-by-humming" [23] concept and specifically **"time-boxes"** [24] [25] and **"sketching"** [26] approaches which we are considering as the primary candidates for the future Hackystat module user-interaction model.

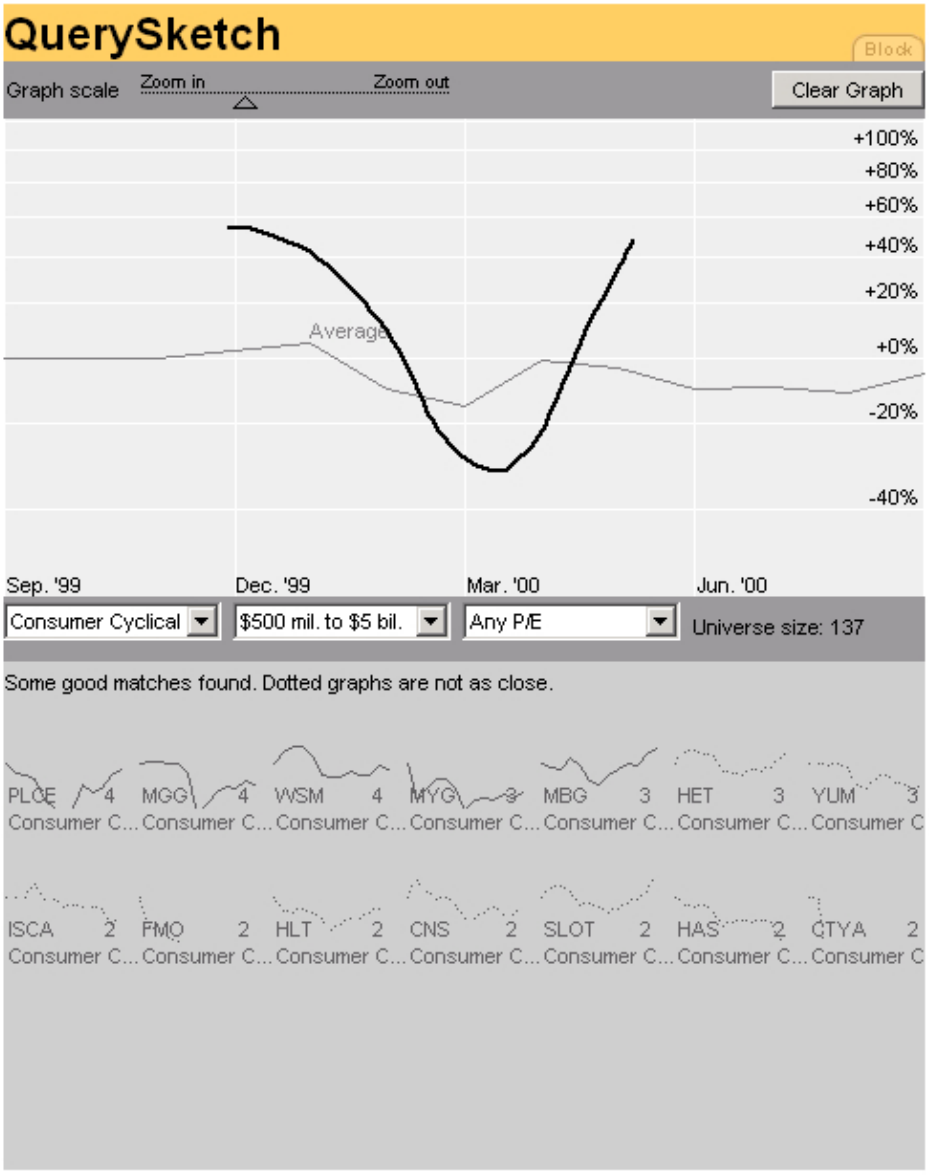

Figure 8: The QuerySketch applet screenshot showing the sketched graph and matching stocks.

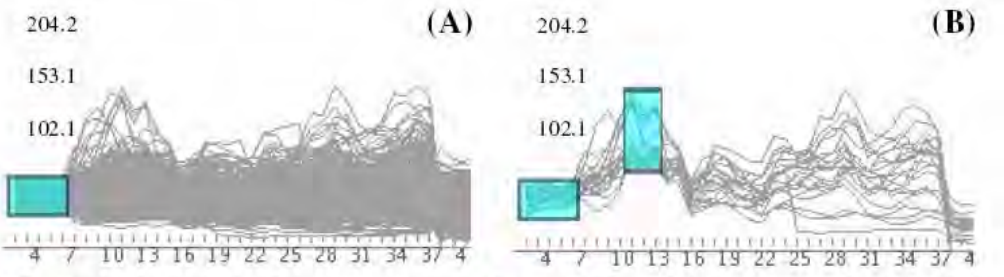

Fig. 2. (A) A single timebox query, for items between $28 and $64 during weeks 1-5. (B) A refinement of the query in (A) reduces the number of matching time series

Figure 9: The figure from E. Keogh, H. Hochheiser, and B. Shneiderman, "An augmented visual query mechanism for finding patterns in time series data".

Query by Example or QBE approach was first introduced as the database query language for relational databases. It was published by IBM Research during 70s in parallel to the development of SQL. It is the first query language where user would enter example elements, commands and conditions while using graphical tables representation. QBE was developed targeting non-programmer user working with databases and currently adopted by many graphical front-end database utilities.

Wattenberg adopted QBE approach for the querying the time-series databases using a graphical applet named QuerySketch [26]. Within the QuerySketch applet environment user allowed to perform a time-series database query by sketching own pattern of interest as shown at the screenshot of the online version of QuerySketch, Figure 8.

Different approach based on the interactive filtering of existing time-series by selecting acceptable "box" ranges was implemented by Keogh et al. [24], see Figure 9 from the original paper explaining the TimeSearcher UI design.

6 Time-series indexing

It is well understood that the direct search for similarity in the time-series databases is computationally expensive. As the remedy for the fast turn-around of database queries "time-series indexing" technique was introduced.

Essentially any of the time-series of length n can be considered as a tuple in an n-dimensional space. Once such a space is indexed, the search usually takes a linear time by performing a simple comparisons. While "dimensionality curse" makes the direct indexing of this space inefficient, the idea is to improve indexing through the application of some dimensionality reduction technique that transforms the n item long time series into a lower dimensional space with k dimensions where $k << n$. The SVD decomposition, the Fourier transform (and the similar Discrete Cosine transform), the Wavelet decomposition, Multidimensional Scaling, random projection techniques, FastMap (and variants) are the methods usually used, [27].

7 Software metrics application

As the immediate application of the "naive" algorithm discussed above we developed the Hackystat ProjectBrowser plugin during the Fall 2008 semester. This plugin called "Trajectory" and the screenshot shown at the Figure 10. Currently, by employing other Hackystat modules such as Sensorbase engine, Dailyprojectdata and Telemetry the Trajectory software runs within the ProjectBrowser infrastructure and allows user to apply the DTW algorithm aligning the set of two arbitrary selected telemetry streams from independent projects and date ranges. Once DTW is computed, Trajectory render two additional to the standard telemetry plots - first is the normalized time-series plot and the second one is the DTW alignment plot. The Euclidean distance is computed for both. Currently Trajectory implements the various step constraints following Sakoe&Chiba [3] but the windowing and the open-end alignment are still in the development.

The primarily goal for the Trajectory plugin was a proof of the concept and the evaluation of the DTW applicability for the software development telemetry time-series. Secondary goal was the engineering of the DTW implementation itself. Due to the limited functionality of the implemented DTW it is impossible to evaluate applicability immediately, but during the literature review and the iterative development process we found enough evidence suggesting that the probability of the application to be useful is relatively high. As per engineering the DTW plugin, two abstraction layers were created - first is the data transformation layer which communicates with the Sensorbase and Telemetry services retrieving the user-defined raw Telemetry data, validating and curing the time-series by adding missing data and finally performing the time series normalization according to [28]. The second layer is the DTW engine itself which takes two abstract time-series and align those by applying the user-specified DTW algorithm.

8 Future work

During the oncoming Spring 2009 semester we are planning to finish the current DTW implementation by adding the windowing constraints and extend the functionality further by the drafting the query-by-example implementation as discussed in 5. Another idea discussed and sketched on paper is the implementation of the telemetry database indexing in order to improve the turn-around time while querying the database. Further development over the telemetry indexing could be an autonomous engine, "telemetry crawler" which will be finding and reporting similar pattern among the data within the Sensorbase storage. The main idea behind such an indexing and autonomous search lies in the splitting each of the time series into the set of overlapping subseries (subsequences) and indexing such a pool of short sequences by building the **library (index) of patterns**. Once the index is built, we are planning to explore the not yet explored in the literature similarity measure

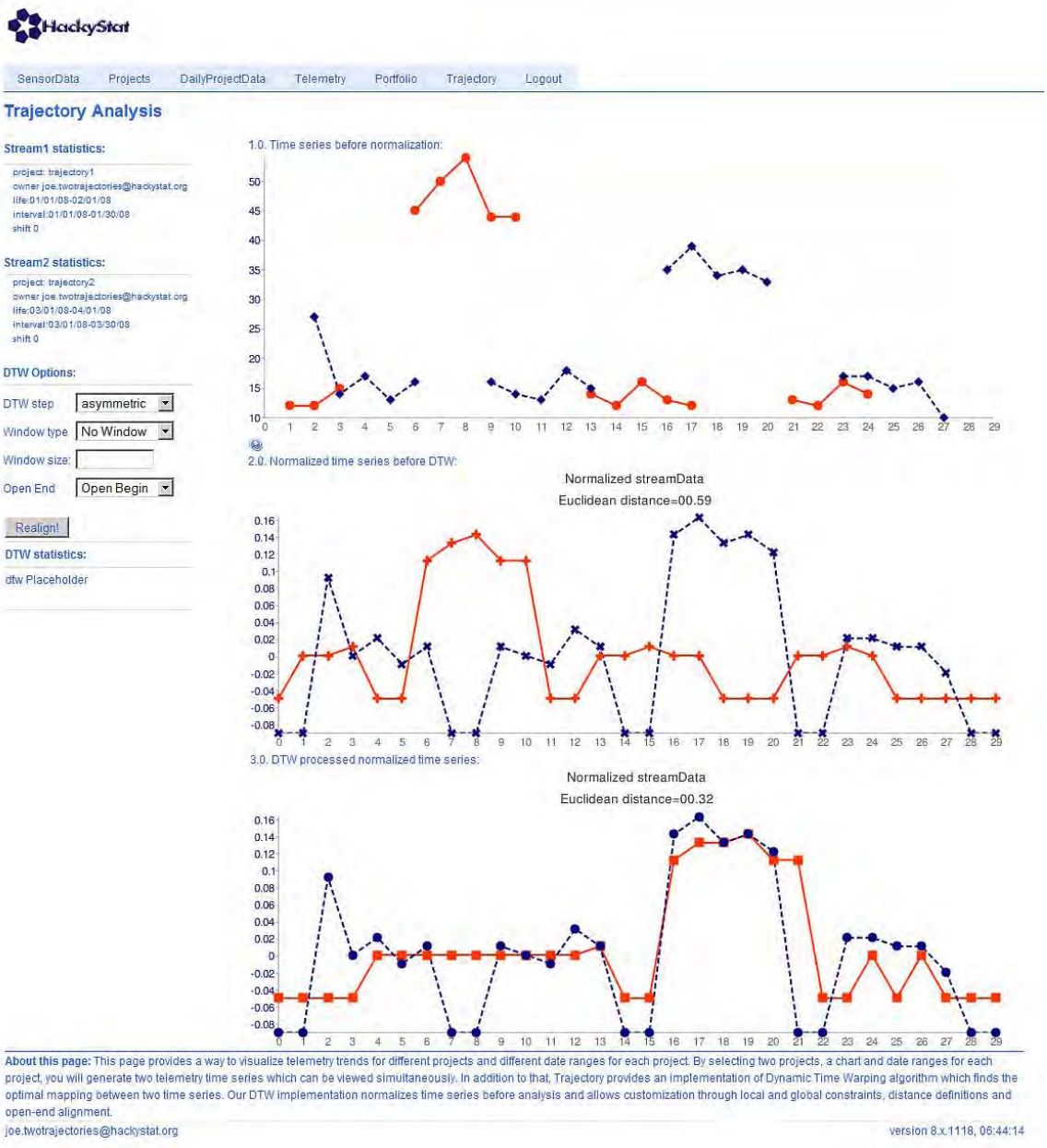

Figure 10: Trajectory page screenshot.

based on the frequency of the occurrence of the shared patterns.

9 Notes

Some of the papers I have seen while working on my report show that one of the DTW successors, Cluster Generative Statistical Dynamic Time Warping (CSDTW) [10], based on dynamic time warping (DTW) and hidden Markov modeling (HMM), demonstrates superior performance over the plain DTW or HMM and combines cluster analysis and generative statistical sequence modeling. It might be interesting to further review the CSDTW and HMM-based methods in particular aiming performance comparison for our needs.

Also I found that DTW has another limitation: it does not obey the triangular inequality [29] which could be a problem while indexing time series.

References

[1] R. Bellman and R. Kalaba, "On adaptive control processes," *Automatic Control, IRE Transactions on*, vol. 4, no. 2, pp. 1–9, 1959. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1104847

[2] C. Myers, L. Rabiner, and A. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, vol. 28, no. 6, pp. 623–635, 1980. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1163491

[3] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing,*

IEEE Transactions on, vol. 26, no. 1, pp. 43–49, 1978. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1163055

[4] A. Efrat, Q. Fan, and S. Venkatasubramanian, "Curve matching, time warping, and light fields: New algorithms for computing similarity between curves," *J. Math. Imaging Vis.*, vol. 27, no. 3, pp. 203–216, April 2007. [Online]. Available: http://dx.doi.org/10.1007/s10851-006-0647-0

[5] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 8, pp. 787–808, 1990. [Online]. Available: http://dx.doi.org/10.1109/34.57669

[6] A. Kuzmanic and V. Zanchi, "Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system," in *EUROCON, 2007. The International Conference on "Computer as a Tool"*, 2007, pp. 264–269. [Online]. Available: http://dx.doi.org/10.1109/EURCON.2007.4400350

[7] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01)*. Washington, DC, USA: IEEE Computer Society, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=882476.883586

[8] V. Niennattrakul and C. A. Ratanamahatana, "On clustering multimedia time series data using k-means and dynamic time warping," in *Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on*, 2007, pp. 733–738. [Online]. Available: http://dx.doi.org/10.1109/MUE.2007.165

[9] J. Gu and X. Jin, "A simple approximation for dynamic time warping search in large time series database," 2006, pp. 841–848. [Online]. Available: http://dx.doi.org/10.1007/11875581_101

[10] C. Bahlmann and H. Burkhardt, "The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 3, pp. 299–310, 2004. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2004.1262308

[11] T. Kahveci and A. Singh, "Variable length queries for time series data," in *Data Engineering, 2001. Proceedings. 17th International Conference on*, 2001, pp. 273–282. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2001.914838

[12] T. Kahveci, A. Singh, and A. Gurel, "Similarity searching for multi-attribute sequences," in *Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on*, 2002, pp. 175–184. [Online]. Available: http://dx.doi.org/10.1109/SSDM.2002.1029718

[13] W. Euachongprasit and C. Ratanamahatana, "Efficient multimedia time series data retrieval under uniform scaling and normalisation," 2008, pp. 506–513. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78646-7_49

[14] "Dtw-based motion comparison and retrieval," 2007, pp. 211–226. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74048-3_10

[15] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 1135–1138. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2006.392

[16] J. Vial, H. Nocairi, P. Sassiat, S. Mallipatu, G. Cognon, D. Thiebaut, B. Teillet, and D. Rutledge, "Combination of dynamic time warping and multivariate analysis for the comparison of comprehensive two-dimensional gas chromatogramsapplication to plant extracts," *Journal of Chromatography A*, September 2008. [Online]. Available: http://dx.doi.org/10.1016/j.chroma.2008.09.027

[17] M. Muller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization," pp. 192–197, 2006.

[18] "Dynamic time warping," 2007, pp. 69–84. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74048-3_4

[19] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1367993

[20] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 23, no. 1, pp. 67–72, 1975. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1162641

[21] S. Chu, E. Keogh, D. Hart, and M. Pazzani, "Iterative deepening dynamic time warping for time series," 2002. [Online]. Available: http://www.siam.org/meetings/sdm02/proceedings/sdm02-12.pdf

[22] M. M. Zloof, "Query-by-example: the invocation and definition of tables and forms," in *VLDB '75: Proceedings of the 1st International Conference on Very Large Data Bases*. New York, NY, USA: ACM, 1975, pp. 1–24. [Online]. Available: http://dx.doi.org/10.1145/1282480.1282482

[23] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2003, pp. 181–192. [Online]. Available: http://dx.doi.org/10.1145/872757.872780

[24] E. Keogh, H. Hochheiser, and B. Shneiderman, "An augmented visual query mechanism for finding patterns in time series data," 2002, pp. 240–250. [Online]. Available: http://dx.doi.org/10.1007/3-540-36109-X_19

[25] H. Hochheiser and B. Shneiderman, "Interactive exploration of time series data," 2001, pp. 441–446. [Online]. Available: http://dx.doi.org/10.1007/3-540-45650-3_38

[26] M. Wattenberg, "Sketching a graph to query a time-series database," in *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2001, pp. 381–382. [Online]. Available: http://dx.doi.org/10.1145/634067.634292

[27] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 517–580, December 2003. [Online]. Available: http://dx.doi.org/10.1145/958942.958948

[28] D. Goldin and P. Kanellakis, "On similarity queries for time-series data: Constraint specification and implementation," 1995, pp. 137–153. [Online]. Available: http://dx.doi.org/10.1007/3-540-60299-2_9

[29] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Data Engineering, 1998. Proceedings., 14th International Conference on*, 1998, pp. 201–208. [Online]. Available: http://dx.doi.org/10.1109/ICDE.1998.655778