Finding Similar Time Series

Gautam Das[1] and Dimitrios Gunopulos[2] and Heikki Mannila[3]

[1] Univ. of Memphis, Memphis TN, dasg@next1.msci.memphis.edu. Research done while the author was visiting the Univ. of Helsinki.
[2] IBM Almaden RC K55/B1, 650 Harry Rd., San Jose CA 95120, USA, gunopulo@almaden.ibm.com. Research done while the author was visiting the Univ. of Helsinki.
[3] Univ. of Helsinki, Dept. of Computer Science, FIN-00014 Helsinki, Finland, Heikki.Mannila@cs.helsinki.fi.

Abstract. Similarity of objects is one of the crucial concepts in several applications, including data mining. For complex objects, similarity is nontrivial to define. In this paper we present an intuitive model for measuring the similarity between two time series. The model takes into account outliers, different scaling functions, and variable sampling rates. Using methods from computational geometry, we show that this notion of similarity can be computed in polynomial time. Using statistical approximation techniques, the algorithms can be speeded up considerably. We give preliminary experimental results that show the naturalness of the notion.

1 Introduction

Being able to measure the similarity or dissimilarity between objects is a crucial point in many data retrieval and data mining applications; see [9] for a general discussion on similarity queries. For complex objects, defining similarity is by no means trivial. In this paper we consider the problem of defining the similarity between time series.

Time series are an important class of complex data objects. They arise in several financial and scientific applications; examples include stock price indices, the volume of product sales, telecommunications data, one-dimensional medical signals, audio data, and environmental measurement sequences.

In many cases it is necessary to search within a time series database for those series that are similar to a given query sequence. This primitive is needed, for example, for prediction and clustering purposes. While the statistical literature on time-series is vast, it has not studied similarity notions that would be appropriate for, e.g., data mining applications.

Intuitively, we consider two sequences similar if they exhibit similar behavior for a large subset of their length. We assume that the sequences to be compared can have

- outliers, i.e., values that are measurement errors and should be omitted when comparing the sequence against others;
- different scaling factors and baselines: the sequences can, e.g., be measurements done using different devices, and the scaling and baseline values can be different.

Our goal is to obtain a measure of similarity that would be resistant[4] with respect

[4] See [8], pages 1–6 for a discussion of resistance and robustness of statistical procedures.

to such changes. That is, if we have a sequence X and modify it to sequence X' by introducing outliers, by scaling and translation, and by adding or removing some observations, the sequences X and X' should still be considered reasonably similar.

We give a definition of similarity that fulfills these conditions and study algorithms that can be used to compute the similarity between sequences. We also discuss some generalizations and specializations of the similarity concept. In detail, the paper is organized as follows. In Section 2 we present the similarity model. In Sections 3 and 4 we give an exact algorithm that finds the similarity of two time sequences, where similarity is defined as above. In Section 5 we give a faster approximation algorithm. This algorithm was implemented and in Section 6 we present some experimental results. Finally, Section 7 is a short conclusion.

The results we present here represent preliminary work, to demonstrate the validity of the approach. A rigorous comparison with different methods will follow. In this extended abstract some of the proofs of lemmas and theorems appear in the Appendix.

2 The similarity model

A time series is a finite sequence X of integers: $(x_1, \ldots, x_n)$.

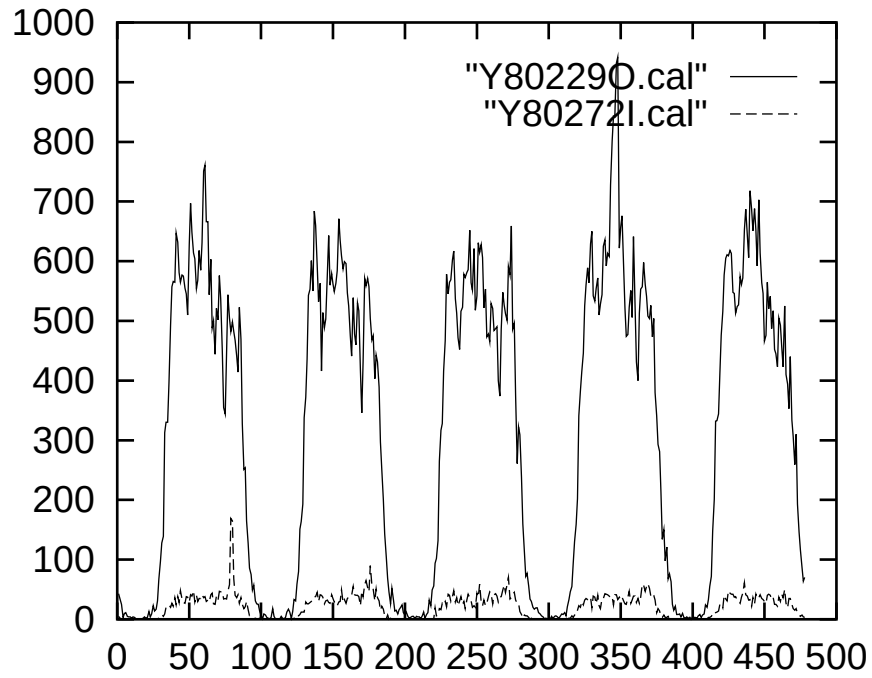

Fig. 1. Two telecommunication sequences: Each sequence represents the number of connections on a given physical line over time.

Fix a set $\mathcal{F}$ of *transformation functions* for mapping integers to integers. The set $\mathcal{F}$ could consist of, say, all linear functions $x \mapsto ax + b$, all scaling functions

$x \mapsto ax$, all quadratic functions, all monotone functions, or just the identity function.

Intuitively, we say that that two sequences $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ are $\mathcal{F}$-similar, if there is a function $f \in \mathcal{F}$ such that a long subsequence X' of X can be approximately mapped to a long subsequence Y' of Y using f.

It is important to note here that X' or Y' does not consist of consecutive points of X (respectively Y). Rather, the points of X' (Y'), appear in the same relative order in X (Y). This means that the matched subsequences allow for a number of holes (outliers) in the original sequences. Clearly, if X and Y are similar, the number of outliers will be small, and X' and Y' will approximate them in length.

Definition 1. Given two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, and numbers $0 < \gamma, \varepsilon \leq 1$, the sequences X and Y are $(\mathcal{F}, \gamma, \varepsilon)$-**similar** if and only if there exist a function $f \in \mathcal{F}$ and subsequences $X_f = (x_{i_1}, \ldots x_{i_{\gamma n}})$ and $Y_f = (y_{j_1}, \ldots y_{j_{\gamma n}})$, where $i_k \leq i_{k+1}$ and $j_k \leq j_{k+1}$ for all $k = 1, \ldots, \gamma n - 1$, such that $\forall k, 1 \leq k \leq \gamma n$,

$$y_{j_k}/(1+\varepsilon) \;\leq\; a\, x_{i_k} + b \;\leq\; y_{j_k}(1+\varepsilon)$$

The parameter γ ($0 < \gamma \leq 1$) is used to control the length of the subsequence of X that can be mapped to Y. The parameter ε controls how close we want the sequences to match. When y_{j_k} and x_{i_k} satisfy the above condition, we say that they are ε-close.

Definition 2. For given $X, Y, \mathcal{F}, \varepsilon$, the similarity of X and Y is

$$\mathbf{Sim}_{\mathcal{F},\varepsilon}(X, Y) = \{\max \gamma \mid X, Y \text{ are } (\mathcal{F}, \gamma, \epsilon)\text{-similar}\}$$

$Sim_{\mathcal{F},\varepsilon}(X, Y)$ is therefore a number between 0 and 1, with numbers closer to 1 signifying greater similarity. In this paper we mostly consider the collection of functions $\mathcal{F}_{lin}$ consisting of linear functions:

$$\mathcal{F}_{lin} = \{x \mapsto ax + b \mid a, b \in \Re\}.$$

This set of functions is reasonably simple, but allows us to find similarities between sequences with different scaling factors and base values. We call $(\mathcal{F}_{lin}, \gamma, \varepsilon)$-similarity simply (γ, ε)-similarity.

Next we very briefly mention some related work; for lack of space, this section is strongly abbreviated. The problem of searching for similar sequences was brought to the database community perhaps mostly by the papers [1, 6, 7, 4]. The idea of using longest common subsequence in measuring similarity between sequences of objects has been proposed in [13]. The similarity model presented here however does not account for different scaling factors and different baseline values. A similar model has been proposed by Agrawal et. al. [2]. The main difference is that, this model does not allow outliers within windows of a specified length W, and the linear function can vary slightly in the length of the matched common subsequence. See also [10] for a collection of material on sequence comparisons.

3 Longest common subsequences

If we know the function $f \in \mathcal{F}$ that is to be used, determining $(\mathcal{F}, \gamma, \varepsilon)$-similarity between X and Y is easy: we form the sequence $f(X)$ by applying f to each element of X, and then locate the longest common subsequence between $f(X)$ and Y. Two numbers are considered equal if they are ϵ-close. The longest common subsequence can be found by simple dynamic programming in $O(n^2)$ time (Theorem 8, in the Appendix); time $O(hn)$ can also be obtained, when the length of the longest common subsequence is at least $n - h$ ([3], Theorem 8 in the Appendix gives a sketch of the algorithm.) We refer to this algorithm as the lcss algorithm, and will use it as a subroutine in the next sections.

The lcss algorithm is able to solve in a simple fashion a seemingly complex problem of determining which elements of two complex objects correspond to each other in the maximal pairing between the objects. Note that the sequence aspect is crucial here: finding maximal pairing between two *sets* is NP-complete.

4 A polynomial algorithm for (γ, ε)-similarity

In this section we present a correct algorithm that, given a pair of sequences X and Y, finds the linear transformation f that maximizes the length of the longest common subsequence of $f(A)$, B (within ε). The algorithm is based on the use of methods from computational geometry.

The main idea of the algorithm is to locate all fundamentally different linear transformations and check each one. Given two linear transformations f_1 and f_2, specified by pairs (a_1, b_1) and (a_2, b_2), we say that they are *equivalent*, denoted $f_1 \equiv_{lin} f_2$, if for all $1 \leq i, j \leq n$ we have: f_1 maps x_i ε-close to y_j if and only if f_2 maps x_i ε-close to y_j.

Lemma 3. *There are at most $O(n^4)$ equivalence classes of $\equiv_{lin}$.*

Algorithm 1 *Find if sequences A, B are (γ, ε)-similar.*

1. *For all equivalence classes of $\equiv_{lin}$, find a representative (a, b).*
2. *For each pair of (a, b), run the lcss algorithm for the sequences $aX + b$ and Y, and test whether the length of the longest common subsequence is at least γn.*

The following theorem is a corollary of the Algorithm 1 and Lemma 1.

Theorem 4. *Given two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, and numbers $0 < \gamma, \varepsilon < 1$, we can compute if X and Y are (γ, ε)-similar in $O(n^6)$ time.*

If we consider the smaller family of scaling functions $\mathcal{F}_{SC} = \{x \mapsto ax \mid a \in \Re\}$, then we obtain the following result.

Theorem 5. *Given two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, and $0 < \gamma, \varepsilon < 1$, we can compute if X and Y are $(\mathcal{F}_{\mathrm{SC}}, \gamma, \varepsilon)$-similar in $O(n^4)$ time.*

5 An Approximate Algorithm

The algorithm presented in the previous section shows that the problem of deciding similarity is solvable in polynomial time, but it is of no practical use. In this section we show how to obtain a faster approximation algorithm.

The main idea is to reduce the number of candidate pairs of (a, b). To do so we use some statistical arguments to compute bounds for possible values of a and b.

Let X', Y' be two matched subsequences of length γn. Then $y'_i/(1+\varepsilon) \leq ax'_i + b \leq y'_i(1+\varepsilon)$, and after summing for all i we obtain:

$$E(Y')/(1+\varepsilon) \leq aE(X') + b \leq E(Y')(1+\varepsilon)$$

Let X_{min}, Y_{min} be the subsequences of X and Y of length γn that minimize the average, and X_{max}, Y_{max} be the subsequences that maximize the average. These subsequences can be found easily, given the value of γ, after sorting the points in the input sequences. After we compute $E(X_{max})$, $E(X_{min})$, $E(Y_{min})$, $E(Y_{max})$, we can bound the values of $E(Y')$ an $E(X')$ in the above inequality from above and below. Thus we obtain the following inequalities:

1. $b \geq -E(X_{max})\, a + E(Y_{min})/(1+\varepsilon)$
2. $b \leq -E(X_{min})\, a + E(Y_{max})(1+\varepsilon)$

These two inequalities define an infinite wedge in (a, b) space (see Figure 2). To get a finite convex area we need at least another inequality. In order to do that, we use the deviation of the sequences. The deviation of a sequence $X = (x_1, \ldots, x_n)$ is:

$$D(X) = \sum_i |x_i - E(X)|$$

Note that $D(a\,X + b) = a\,D(X)$.

Lemma 6. *Let X', Y' be two matched subsequences of length γn. Then:*

$$|D(Y') - D(aX' + b)| \leq 2\,\gamma\,n\,\varepsilon\,|E(Y')|$$

Let X_{dmin}, Y_{dmin} be the subsequences of length γn that minimize the deviation, and X_{dmax}, Y_{dmax} be the subsequences that maximize the deviation. Then, from the previous inequality, we have:

1. $a\,D(X_{dmax}) \geq D(Y_{dmin}) - 2\,\gamma\,n\,\varepsilon\,E(Y_{max})$
2. $a\,D(X_{dmin}) \leq D(Y_{dmax}) + 2\,\gamma\,n\,\varepsilon\,E(Y_{max})$

These two inequalities together with the other two ones define a bounded quadrilateral in (a, b) space.

To use these two inequalities however we have to find the subsequences that minimize or maximize the deviation.

Lemma 7. *Given a sequence $X = (x_1, \ldots, x_n)$ and $\gamma \in [0, 1]$, assume that the subsequence of length γn of X that minimizes the deviation is $X_{min} = (x'_1, \ldots, x'_{\gamma n})$. If $x'_{min} = \min(x'_1, \ldots, x'_{\gamma n})$ and $x'_{max} = \max(x'_1, \ldots, x'_{\gamma n})$, then for all $x_i \in X \setminus X_{min}$, either $x_i \leq x'_{min}$ or $x_i \geq x'_{max}$.*

The previous lemma shows that the following $O(n \log n)$ running time algorithm computes the subsequence of length γn that minimizes the deviation.

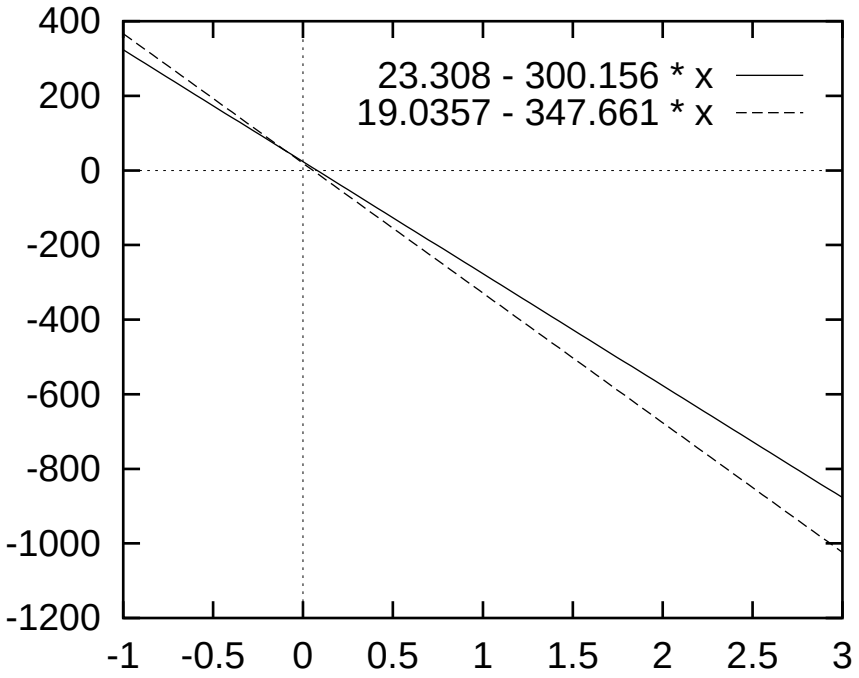

Fig. 2. These two lines represent the two inequalities that were obtained for a specific pair of sequences, using the subsequences that minimized or maximized the average.

Algorithm 2 *Find the subsequence that minimizes the deviation.*

1. *Sort the points.*
2. *Slide a window of size γn, and compute the deviation of the subsequence inside the window.*

The deviation of the new sequences can be computed incrementally, so step two of this algorithm can be performed in linear time. To find the subsequence that maximizes the deviation we use a similar argument. In this case we slide a window of length $(1-\gamma)n$. Now it is the points outside this window that are the points in the subsequence.

We can now give the outline of the approximation algorithm.

Algorithm 3 *Find if sequences X, Y are (γ, ε)-similar.*

1. *Compute bounds for a, b.*
   These define a convex area in (a, b) space.
2. *Use a grid to sample the area defined by the bounds.*
3. *For each grid point (a, b), apply the linear transformation $x'_i = ax_i + b$ and run the lcss algorithm for Y, X'.*

The running time of the algorithm is $O(M(1-\gamma)n^2)$, where M is the number of sampling points. We are trying to find a longest common subsequence of length at least γn, so the running time of the lcss algorithm is $O(1-\gamma)n^2)$. The accuracy of the algorithm depends on the size of the sampling grid. In our experiments, we use a sampling interval of $(\epsilon/2)a_j$ for a, and a constant value for b.

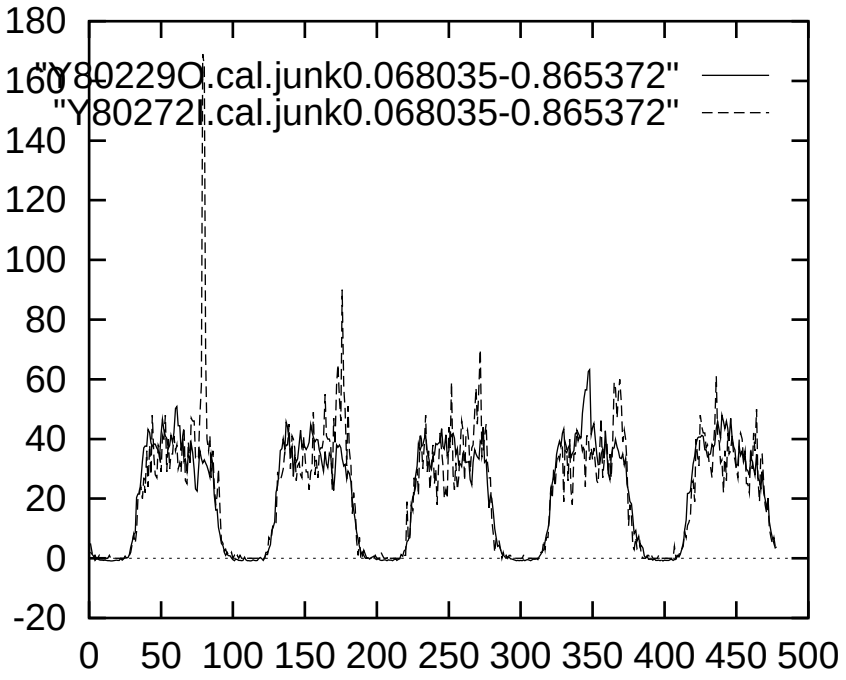

Fig. 3. This figure shows the same sequences shown in Figure 1. The linear transformation 0.068 x - 0.865 maximizes the length of the longest common subsequence for these sequences.

6 Experimental Results

The approximate algorithm was implemented in C, and we used this implementation to find similar sequences among a set of telecommunication sequences. Each sequence represented the number of telephone connections that went through a given physical telephone line over time. The measurements were obtained by sampling the line every 15 minutes. Some sequences represented the number of connections that were established during this time, and some represented the number of connections that were on (but might have been established earlier). Each sequence was 480 points long (5 days.)

We used a set of 34 sequences, and ran the algorithm for each pair. We used large values for ϵ, between 0.2 and 0.3, but we observed little variation on the final results for different values of ϵ. For each pair, 480 minus the length of longest common subsequence found was used as the distance between two sequences. Thus we obtained a 34×34 distance matrix, which was fed to the SAS clustering software package.

The results of the clustering were compared to the results of visual classification of the sequences. The two different kinds of sequences were in different clusters. Office phone lines, which present a distinct pattern were clustered together. Pairs of sequences that appeared similar to a human observer were clustered together. In addition, similarities that we didn't notice before, mainly due to different scale, were brought forward. For example Figure 3 shows the best match for the two sequences of Figure 1. Figure 4 gives an example of the matchings obtained.

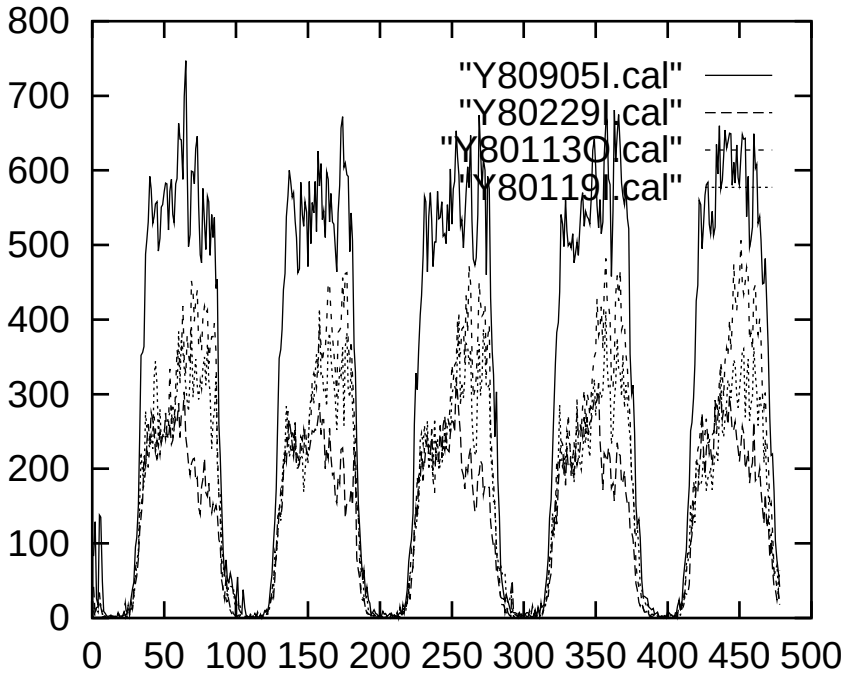

Fig. 4. Of the sequences shown, Y80905I and Y80229I were clustered together, and so were Y80113O and Y80119I, but the two pairs were in different clusters ($\epsilon = 0.2$.)

7 Discussion

We present an intuitive model to capture the similarity of two time series, and a practical approximate algorithm to measure the similarity under this model. The algorithm has been implemented, and has been applied to a set of telecommunication data with encouraging results.

These results represent preliminary work. More experimental work has to be done to properly evaluate the behavior of the model and to compare this approach with existing different ones.

The model can be modified in several ways. One of the most interesting possible changes is the *bounded-offset* restriction, which means that each element x_i may only be mapped to an element y_j of sequence Y such that $|i - j| < K$, where K is a constant ([5].) This restriction has the role of forbidding very large timing differences between the sequences, and it seems to be quite reasonable in several application domains. In the bounded offset model, the complexities of the algorithms are typically a factor of n lower than in the general model, as the longest common subsequence computation can be speeded up.

The approximation techniques presented in Section 5 can be sharpened by noting that the linear transformations preserve the distributional properties of the sequences very well. For example, if there are m of repeated values in sequence X, then in order for Y to be (γ, ε)-similar there must be in Y approximately $m - \gamma n$ values that are within a factor of ε from each other.

An important future research direction is to consider the database problem: Given a set of time series, and a query sequence, find the ones that are similar to the query. In order to avoid comparing the query sequence with each sequence

in the database, we have to use some approximation (fingerprint) scheme that reduces the dimension of the sequences, such as the wavelet transformation, or the deviation of subsequences. To compute fingerprints of small dimension Agrawal et al [1] use the Discrete Fourier Transform, Shatkay et al [11] use feature extraction, and other methods have been suggested; see [12] for some general discussions on fingerprinting.

References

1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO'93)*, Chicago, 1993.
2. R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 490 – 501, Zurich, Swizerland, 1995.
3. A. V. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 255 – 400. Elsevier Science Publishers B.V (North-Holland), Amsterdam, 1990.
4. D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 229 –248. AAAI Press, Menlo Park, CA, 1996.
5. B. Bollobás, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated geometric sets. In *ACM Computational Geometry Conference*, 1997.
6. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD'94*, May 1994.
7. D. Goldin and P. Kanellakis. On similarity queries for time-series data: constraint specification and implementations. In *Int. Conf. on the Principles and Practice of Constraint Programming*, pages 137–153, 1995.
8. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, editors. *Understanding Robust and Exploratory Data Analysis.* Wiley, 1982.
9. H. Jagadish, A. O. Mendelzon, and T. Milo. Similarity-based queries. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'95)*, pages 36–45, 1995.
10. D. Sankoff and J. B. Kruskal. *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison Wesley, 1983.
11. H. Shatkay and S. Zdonik. Approximate queries and representations for large data sequences. In *ICDE'96*, 1996.
12. D. A. White and R. Jain. Algorithms and strategies for similarity retrieval. Technical Report VCL-96-101, Visual Computing Laboratory, University of California, San Diego, 9500 Gilman Drive, Mail Code 0407, La Jolla, CA 92093-0407, July 1996.
13. N. Yazdani and Z. M. Ozsoyoglu. Sequence matching of images. In *Proceedings of the 8th International Conference on Scientific and Statistical Database Management, Stockholm*, pages 53–62, 1996.

A Proofs

Theorem 8. *Given two time series* $X = (x_1, \ldots, x_n)$ *and* $Y = (y_1, \ldots, y_n)$, *and* $0 < \epsilon < 1$, *we can compute the longest subsequences* $X_{ab} = (x_1, \ldots x_s)$ *and*

$Y_{ab} = (y_1, \ldots y_s)$ *such that* $\forall i, 1 \leq i \leq s,\ y_i/(1+\epsilon) \leq x_i \leq y_i(1+\epsilon)$, *in* $O(n^2)$ *time.*

Proof: A dynamic programming algorithm is used. Let $D[i,j]$ be the length of the longest common subsequence of sequences $X = (x_1, x_2, \ldots, x_i)$ and $Y = (y_1, y_2, \ldots, y_j)$. Then $D[n,n]$ is the length of the longest common subsequence of X and Y. Is easy to see that if $|x_i - y_j| > \epsilon$, then $D[i,j] = \max(D[i-1,j], D[i,j-1])$. Otherwise $D[i,j] = \max(D[i-1,j], D[i,j-1], D[i-1,j-1]+1)$. □

Proof of Lemma 3: Let us assume that for a given pair of (a,b), the points x_i and y_j are in the longest common subsequence, and in fact are mapped to each other. Since the transformation $aX+b$ has been applied to X, the following inequality must hold: $y_i/(1+\varepsilon) \leq a\,x_i + b \leq y_i(1+\varepsilon)$

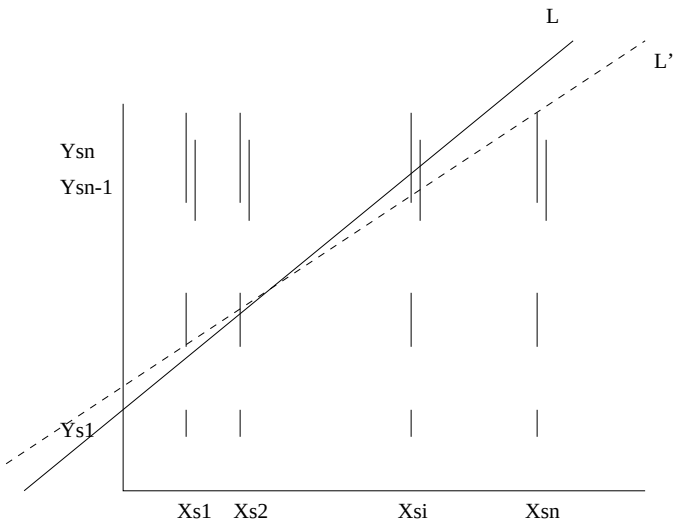

Fig. 5. The two lines intersect the same number of line segments, and therefore belong in the same equivalence class.

Consider the plane where X is mapped to the horizontal axis and Y to the vertical (Figure 5). We will refer to this as the (x,y) plane. (x_i, y_j) represents a point on this plane. The linear transformation $aX+b$ represents the line $y = ax+b$. Notice that x_i and y_j can be mapped to each other after the linear function $ax+b$ has been applied to X if and only if $y_i/(1+\varepsilon) \leq a\,x_i + b \leq y_i(1+\varepsilon)$

As a result, the line $y = a\,x + b$ must intersect the line segment $[(x_i, y_j/(1+\varepsilon)), (x_i, y_j(1+\varepsilon))]$. Conversely, every line $y = a_1 x + b_1$ in the (x,y) plane corresponds to a linear transformation on X. If such a line intersects a line segment $[(x_i, y_j/(1+\varepsilon)), (x_i, y_j(1+\varepsilon))]$, then x_i can be matched to y_j in the longest common subsequence of sequences $a_1X + b_1$ and Y.

There is a total of n^2 such line segments, one for each pair $(x_i, y_j), 1 \leq i \leq n, 1 \leq j \leq n$. If two different lines intersect the same set of line segments, then the longest common subsequences obtained for the corresponding two linear functions are the same; the two linear functions are indistinguishable within ε.

Consider a line $l : y = ax + b$. Let S be the set of line segments that l intersects. If l does not pass through two line segment endpoints, then we can

first slide l vertically until the first endpoint is encountered, and then rotate l around this endpoint until it touches a second endpoint. Let l' be the new line. l' also intersects the line segments in S, and no additional one. Since we can use this procedure to reduce any line to an equivalent line that passes through endpoints, we only have to consider the lines that pass through two endpoints. There are $2n^2$ endpoints, and these define $O(n^4)$ lines. □

Proof of Theorem 4: Lemma 3 shows that we can find $O(n^4)$ lines that represent all the equivalence classes of $\equiv_{lin}$ simply by pairing all line segment endpoints. However not all of these lines are significant. If a line intersects less than γn line segments then the resulting longest common subsequence will have length less than γn. To enumerate the lines that intersect at least γn line segments, for each endpoint we sort all other end points in counter-clock-wise order around it, and then we sweep a line around the point. At each new endpoint we update the number of line segments crossed by the line, and this is a constant time operation since at each new endpoint this number either increases or decreases by one. Whenever this number is at least γn we run the lcss algorithm for this linear transformation. The running time is then $O(n^4(n^2 + \log n)) = O(n^6)$. □

Proof of Lemma 6: Let's assume that $x_i, y_i \geq 0$.

$$\begin{aligned} |D(Y') \ - D(aX' + b)| &= |\sum_i |y_i - E(Y')| - \sum_i |ax_i + b - aE(X') - b|| \\ &= |\sum_i (|y_i - E(Y')| - |ax_i + b - aE(X') - b|)| \end{aligned}$$

But we have that $|y_i - ax_i - b| \leq \varepsilon |y_i|$ and $|E(Y') - aE(X') - b| \leq \varepsilon E(Y')$. Taking all cases we have that $|y_i - E(Y')| - |ax_i - aE(X')| \leq \varepsilon E(Y') + \varepsilon y_i$, and therefore

$$|\sum_i (|y_i - E(Y')| - |ax_i + b - aE(X') - b|)| \leq 2\gamma \, n \, \varepsilon \, E(Y')$$

□

Proof of Lemma 7: Let us assume there is an $x_i \in X \setminus X_{min}$, such that $x'_{min} < x_i < x'_{max}$. Let us also assume $x_i < E(X_{min})$; the other case is symmetric. By replacing x_{min} with x_i in the subsequence X_{min} we get a new subsequence X' of length γn.

Since $x'_{min} < x_i$, $E(X_{min}) + (x_i - x'_{min})/(\gamma n) = E(X')$. Since x_i is smaller than $E(X_{min})$, it is also smaller than $E(X')$. Then the deviation of X' compared to X_{min} decreases by $(x_i - x'_{min})(\gamma n - 1)/(\gamma n)$ because of the replacement of x_{min} with x_i. The maximum increase of the deviation is $(\gamma n - 3)(x_i - x'_{min})/(\gamma n)$. This occurs if the average is larger than all points except one (the average cannot be larger than the largest point) and after the replacement the new average moves away from $\gamma n - 2$ points and closer to one point. So X' has a smaller deviation than X_{min}, a contradiction. □