Time-Series Similarity Problems and Well-Separated Geometric Sets

Béla Bollobás* Gautam Das† Dimitrios Gunopulos‡ Heikki Mannila§

Abstract

Given a pair of nonidentical complex objects, defining (and determining) how similar they are to each other is a nontrivial problem. In data mining applications, one frequently needs to determine the similarity between two time series. We analyze a model of time-series similarity that allows outliers, different scaling functions, and variable sampling rates. We present several deterministic and randomized algorithms for computing this notion of similarity. The algorithms are based on nontrivial tools and methods from computational geometry. In particular, we use properties of families of well-separated geometric sets. The randomized algorithm has provably good performance and also works extremely efficiently in practice.

1 Introduction

Being able to measure the *similarity* between objects is a crucial issue in many data retrieval and data mining applications; see [10] for a general discussion on similarity queries. Typically, the task is to define a function $Sim(X, Y)$, where X and Y are two objects of a certain class, and the function value represents how "similar" they are to each other. For complex objects, designing such functions, and algorithms to compute them, is by no means trivial.

Time series are an important class of complex data objects: they arise in many applications. Examples of time series databases are stock price indices, volume of product sales, telecommunication data, one-dimensional medical signals, audio data, and environmental measurement sequences.

In *data mining* applications, it is often necessary to search within a series database for those series that are similar to a given query series. 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.

*Institute for Advanced Study, Princeton, N.J. 08540 AND University of Memphis, Department of Mathematical Sciences, Memphis, TN 38152, USA, bollobas@IAS.EDU.

†University of Memphis, Department of Mathematical Sciences, Memphis, TN 38152, USA, dasg@mathsci.msci.memphis.edu.

‡IBM Almaden RC K55/B1, 650 Harry Rd., San Jose CA 95120, USA, gunopulo@almaden.ibm.com.

§University of Helsinki, Department of Computer Science, P.O. Box 26, FIN-00014 Helsinki, Finland, Heikki.Mannila@cs.helsinki.fi.

In this paper we present and analyze some similarity functions for time series. The main contributions of our paper are, we present new deterministic and randomized algorithms that are both practical *and have* provable performance bounds. Interestingly, these algorithms are based on several subtle geometric properties of the problems. In particular, we investigate properties of certain *well-separated* geometric sets, which are interesting in their own right. In addition to our theoretical analysis, we present several implementation results.

Next we describe the similarity notion used here;[1] related work is considered at the end of the introduction.

Suppose we are given two sequences $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$. A simple starting point would be to measure the similarity of X and Y by using their L_p-distance as points of R^n. For time series, this way of measuring similarity or distance is not appropriate, since the sequences can have outliers, different scaling factors, baselines, and sampling factors.

Outliers are values that are measurement errors and should be omitted when comparing the sequence against others. A grossly outlying value can cause two otherwise identical series to have large distance. A more reasonably similarity notion is based on the *longest common subsequence* concept, where intuitively X and Y are considered similar if they exhibit similar behavior for a large part of their length. More formally, let $X' = x_{i_1}, \ldots, x_{i_l}$ and $Y' = y_{j_1}, \ldots, y_{j_l}$ be the longest subsequences in X and Y respectively, where (a) for $1 \leq k \leq l-1$, $i_k < i_{k+1}$ and $j_k < j_{k+1}$, and (b) for $1 \leq k \leq l$, $x_{i_k} = y_{j_k}$. We define $Sim(X, Y)$ to be l/n. Note that it is not necessary for the two given sequences to have the same length, because the shorter sequence can always be padded with dummy numbers.

There are still several shortcomings in the above model. The two sequences may have different *scaling factors* and *baselines.* For example, two stock indices could be essentially similar (i.e. they react similarly to changing market conditions) even though one fluctuates near \$30 while the other fluctuates near \$100. Secondly, the two sequences could have *variable sampling rates*, i.e. the rate of measurement of one sequence could (for some time) be different from the other. Thirdly, in practice there should be some allowable *tolerance* when comparing elements from both sequences (even after one sequence has been appropriately scaled or otherwise transformed to resemble the other sequence).

The following similarity function overcomes these problems.

The similarity function $Sim_{\epsilon,\delta}(X, Y)$:

Let $\delta > 0$ be an integer constant, $0 < \epsilon < 1$ a real constant, and f a linear function (of the form $f : y = ax + b$) belonging to the (infinite) family of linear functions $\mathcal{L}$. Given two sequences $X = x_1, \ldots, x_n$ and $Y = y_1, \ldots, y_n$, let $X' = (x_{i_1}, \ldots, x_{i_l})$ and $Y' = (y_{j_1}, \ldots, y_{j_l})$ be the longest subsequences in X and Y respectively such that

1. for $1 \leq k \leq l-1$, $i_k < i_{k+1}$ and $j_k < j_{k+1}$,
2. for $1 \leq k \leq l$, $|i_k - j_k| \leq \delta$, and
3. for $1 \leq k \leq l$, $y_{j_k}/(1+\epsilon) \leq f(x_{i_k}) \leq y_{j_k}(1+\epsilon)$.

[1]and in a preliminary form in [7]

Let $S_{f,\epsilon,\delta}(X,Y)$ be defined as l/n. Then $Sim_{\epsilon,\delta}(X,Y)$ is defined as $\max_{f \in \mathcal{L}}\{S_{f,\epsilon,\delta}(X,Y)\}$.

Thus, when $Sim_{\epsilon,\delta}(X,Y)$ is close to 1, the two sequences are considered to be very similar. The constant δ ensures that the positions of each matched pair of elements are not too far apart. In practice this is a reasonable assumption, and it also helps in designing very efficient algorithms.

The linear function (or transformation) f allows us to detect similarity between two sequences with different base values and scaling factors. Note than an algorithm trying to compute the similarity will have to find out *which* linear function to use (i.e. the values of a and b) that maximizes l.

Depending on the application, we could also define similarity using other interesting function families such as, scaling functions (i.e. of the form $y = ax$), quadratic functions, various monotone functions, etc.

The tolerance ϵ allows us to "approximately" match an element in X (after transformation) with an element in Y. Depending on the application, other reasonable tolerance models worth considering would be, *absolute tolerance* (i.e. $y_{j_k} - \epsilon \leq f(x_{i_k}) \leq y_{j_k} + \epsilon$), or even a *combination* of absolute and relative tolerance (i.e., $y_{j_k}/(1+\epsilon_1) - \epsilon_2 \leq f(x_{i_k}) \leq y_{j_k}(1+\epsilon_1) + \epsilon_2$, for two given constants ϵ_1 and ϵ_2), etc.

We observe that our similarity function is not necessarily symmetric; in general $Sim_{\epsilon,\delta}(X,Y)$ may not be the same as $Sim_{\epsilon,\delta}(Y,X)$. This is because, even though $f(x_{i_k})$ may be within ϵ tolerance of y_{i_k}, $f^{-1}(y_{i_k})$ may not be within the same tolerance of x_{i_k}. A more symmetric definition of similarity would be to use $\max\{Sim_{\epsilon,\delta}(X,Y),\ Sim_{\epsilon,\delta}(Y,X)\}$.

Given two sequences of length n, the longest common subsequence can be found in $O(n^2)$ time by a well known dynamic programming algorithm [3, 6]; This algorithm can easily be modified to compute $S_{f,\epsilon,\delta}(X,Y)$ in $O(n\delta)$ time where f is a given linear function. Note that this is essentially linear time. We refer to this algorithm as the LCSS algorithm.

To design algorithms to compute $Sim_{\epsilon,\delta}(X,Y)$, the main task is to locate a finite set of all fundamentally different linear transformations and run LCSS on each one. In this paper we concentrate primarily on this problem. Wherever appropriate, we mention the modifications necessary to our algorithms to handle other variations such as different function families (e.g. scaling functions), absolute tolerance, etc.

In the present paper we describe algorithms that are based on a thorough analysis of the geometric properties of the problem. These algorithms have provable performance bounds, and some of them are very efficient in practice. The geometric properties that we discover are of independent interest and represent a major contribution of the paper. We summarize our results below.

1. **Cubic-time exact algorithm**: $Sim_{\epsilon,\delta}(X,Y)$ can be computed in $O(n^3\delta^3)$ time.

2. **Quadratic-time approximation algorithm**: Let $0 < \beta < 1$ be any desired small constant. A linear transformation f and the corresponding $S_{f,\epsilon,\delta}(X,Y)$ can be computed in $O(n^2\delta^2 + n\delta^3/\beta^2)$ time such that $Sim_{\epsilon,\delta}(X,Y) - S_{f,\epsilon,\delta}(X,Y) \leq \beta$.

3. **Linear-time randomized approximation algorithm**: Let $0 < \beta < 1$ be any desired small constant. A linear transformation f and the corresponding $S_{f,\epsilon,\delta}(X,Y)$ can be computed in $O(n\delta^3/\beta^2)$ expected-time such that $Sim_{\epsilon,\delta}(X,Y) - S_{f,\epsilon,\delta}(X,Y) \leq \beta$.

4. **Experimental results**: The randomized approximation algorithm is easy to implement, is much faster than the exact algorithm, and finds very good approximations to the similarity measure. The absolute difference in the length of the longest common subsequence found is typically 1 or 2 for a very small number of random trials.

While developing these algorithms we discovered several geometric results which are of independent interest. We describe these results below.

Let $S_i \Delta S_j$ represent the *symmetric-difference* between two sets. Let V be a finite set and 2^V be its power set. Let $k > 0$ be an integer. A family of finite sets $\mathcal{S} \subseteq 2^V$ is *k-separated* if for all $S_i, S_j \in \mathcal{S}$, $|S_i \Delta S_j| > k$. It is known that there exist a k-separated family $\mathcal{S}$ where $|\mathcal{S}| = 2^{\alpha n}$ where α depends on k/n [4]. However, we can get much better bounds if we only consider certain kinds of geometric sets.

Consider the following set system. Let R be a fixed set of n line segments on the plane. Given any infinite line L, let R_L be the set of line segments of R intersected (or *stabbed*) by L. (If $R_L = R$, then L is known as a *transversal* of R; refer to [8] for interesting combinatorial and algorithmic results on transversals). Let the family $\mathcal{S}$ consist of the distinct stabbed sets R_L of R, for all possible infinite lines L. The following result and its corollaries are crucial for our algorithms.

4. **k-separated stabbed sets**: The maximum size of a k-separated family $\mathcal{S}' \subseteq \mathcal{S}$ is $\Theta(n^2/k^2)$.

In particular, if $k = \beta n$ for some constant $0 < \beta < 1$, the size of any k-separated family is $O(1)$. It is this property that is used in our approximation algorithms.

While not directly relevant to our time-series algorithms, our next results concern two other interesting and natural geometric sets. Let V be a fixed set of n points on the plane. A *half-plane* is the unbounded region to one side of an infinite line (i.e. any line, and not necessarily one that passes through a point in V). Thus a half-plane can be considered as a subset of V that it contains. Let the family $\mathcal{H}$ consist of the distinct subsets of V corresponding to all possible half-planes.

5. **k-separated half-planes**: The maximum size of a k-separated family $\mathcal{H}' \subseteq \mathcal{H}$ is $\Theta(n^2/k^2)$.

Let V again be a fixed set of n points on the plane. A *triangle* is the intersection of three half-planes (the vertices of the triangle do not have to be points in V). Thus a triangle can be considered as a subset of V that it contains. Let the family $\mathcal{T}$ consist of the distinct subsets of V corresponding to all possible triangles.

6. **k-separated triangles**: The maximum size of a k-separated family $\mathcal{T}' \subseteq \mathcal{T}$ is $\Theta(n^6/k^6)$.

Some of our results extend to higher dimensions as well as to other kinds of geometric sets (such as p-topes). Details will appear in the full paper.

The rest of the paper is organized as follows. In Section 2 we discuss the results concerning k-separated geometric sets. We then describe in Section 3 the exact algorithm, in Section 4 the approximate algorithm, and in Section 5 the randomized algorithm. In Section 6 we discuss our implementation results. We conclude with some future research directions.

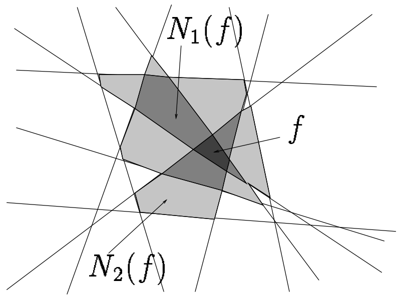
Related work: There has been some recent work on the problem of defining similarity between time series; due to lack of space we only give some references. The problem was introduced to the data mining community by papers [1, 9]. The similarity measures considered in [2, 7, 13] use the concept of *longest common subsequence*. In [1], a *fingerprint* method is used, where the discrete Fourier transform is employed to reduce the dimensions of each sequence, and the resulting fingerprints are compared. In [11], *feature extraction* techniques are used; see [12] for a general discussion on fingerprinting techniques.

Essentially the same similarity function as above was investigated in [7] (except that δ was not considered). In that paper, each linear function $y = ax + b$ is treated as a point (a, b) in the dual ab-plane. Using statistical measures, a bounded polygonal region in the ab-plane is computed which is guaranteed to contain all potential transformations. This region is then sampled by overlaying a grid on it, and running LCSS on the functions represented by the grid vertices. While the algorithm works reasonably well in practice, its efficiency is sensitive to the values of the input numbers (as opposed to *input size*), and furthermore, the computed similarity may not be the optimal. Our experiments show that the randomized algorithm presented in this paper produces more accurate results and is far faster.

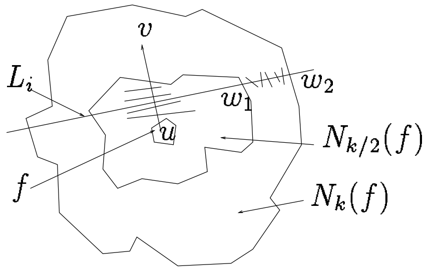
2 k-Separated Geometric Sets

In this section we discuss k-separated geometric sets. We start by proving an useful lemma concerning *arrangements* of infinite lines.

Let H be a set of n infinite lines on the plane in general positions (i.e no three line intersect at a point). It is well known that the arrangement $A(H)$ is a planar graph with $\Theta(n^2)$ vertices, edges and faces [8]. Let $k > 0$ be any integer. For any face f in the arrangement, define a region $N_k(f)$ as the union of all faces g such that a line segment connecting a point within f with a point within g intersects at most k lines of H. Figure 1(a) illustrates these notions.



(a)



(b)

Figure 1: The $N_k(f)$ regions

Lemma 2.1 *The number of vertices of $A(H)$ contained within $N_k(f)$ is $\Omega(k^2)$.*

Proof : (Sketch) Let u be a point in f and v a "target" point beyond the boundary of $N_{k/2}(f)$ (see Figure 1(b)). If we start from u and start walking straight towards v, we will intersect $k/2$ lines of H, say $L_1, L_2, \ldots, L_{k/2}$, before we reach the boundary of $N_{k/2}(f)$. Consider one such line, say L_i. Since it is an infinite line, it has to intersect the boundary of $N_{k/2}(f)$ as well as the boundary of $N_k(f)$. Let (w_1, w_2) be a portion of L_i within $N_k(f) \setminus N_{k/2}(f)$ such that w_1 (resp. w_2) is on the boundary of $N_{k/2}(f)$ (resp. $N_k(f)$). If we walk along L_i from w_1 to w_2, it should be clear that we will encounter at least $k/2$ vertices of $A(H)$.

Thus, there are $k/2$ lines of H such that each line contributes at least $k/2$ vertices of $A(H)$ inside $N_k(f)$. The same vertex can be contributed by at most two lines. Thus the number of vertices contained within $N_k(f)$ is $\Omega(k^2)$. □

We are now ready to prove our k-separated results. We start with stabbed sets. Consider a well known *duality transformation*, where a (non-vertical) line $y = ax + b$ in the xy-plane is mapped to the point (a, b) in the ab-plane, and a point (x, y) is mapped to the line $b = xa - y$ [5, 8]. By this transformation, a line segment (u, v) in the xy-plane (Figure 2(a)) corresponds to a *double-wedge* in the ab-plane, where the duals of u and v form the boundaries of the double-wedge (Figure 2(b)). Furthermore, if an infinite line L intersects (u, v), then the dual of L becomes a point inside the double-wedge.

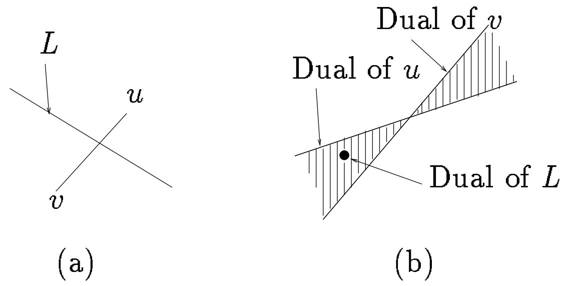


Figure 2: The dual of a line segment is a double-wedge

Let R be a fixed set of n line segments in the xy-plane, and let the family $\mathcal{S}$ consist of the stabbed sets of all possible infinite lines. Let $k > 0$ be any constant.

Theorem 2.2 *The maximum size of a k-separated family $\mathcal{S}' \subseteq \mathcal{S}$ is $\Theta(n^2/k^2)$.*

Proof : (Sketch) We shall only prove the upper bound in this version of the paper. Compute the dual transformations of all segments in R. We thus get a collection of double-wedges in the ab-plane. Compute the arrangement of these double-wedges. The number of vertices (also edges and faces) of this arrangement is $\Theta(n^2)$. In this arrangement, each face f is essentially the intersection of a subset of the double-wedges. Any point inside f represents the dual of an infinite line that stabs precisely the corresponding subset of R in the xy-plane. It is also easy to see that for all $g \in N_k(f)$, the symmetric difference between the subsets of R corresponding to f and g is at most k.

Let $\mathcal{S}' \subseteq \mathcal{S}$ be the largest k-separated family. Consider any pair of subsets $S'_i, S'_j \in \mathcal{S}'$. Let f'_i and f'_j be their corresponding faces in the arrangement of double-wedges. Clearly the region $N_{k/2}(f_i')$ is completely disjoint from the region $N_{k/2}(f_j')$. But by Lemma 2.1 each such region contains at least $\Omega(k^2)$ vertices of the arrangement. Thus, $|\mathcal{S}'| = O(n^2/k^2)$. $\square$

Our next result is on half-planes. Let V be a fixed set of n points on the plane. Let the family $\mathcal{H}$ consist of the distinct subsets of V corresponding to all possible half-planes.

Theorem 2.3 *The maximum size of a k-separated family $\mathcal{H}' \subseteq \mathcal{H}$ of half-planes is $\Theta(n^2/k^2)$.*

The proof is structurally similar to the proof of Theorem 2.2, and we omit it from this version of the paper.

Our final result in this section is on triangles, and is somewhat more involved. We first introduce some useful concepts. Given a family of sets $\mathcal{S}$, a subfamily $\mathcal{S}'$ is a *k-cover* if for all $S \in \mathcal{S}$, there is $S' \in \mathcal{S}'$ such that $|S \Delta S'| \leq k$. The following simple greedy algorithm constructs a k-cover for $\mathcal{S}$, such that the cover *is also k-separated.* Initialize $\mathcal{S}'$ to empty. While $\mathcal{S}$ is not empty, repeat the following steps: delete any $S' \in \mathcal{S}$ from $\mathcal{S}$, add S' to $\mathcal{S}'$, and delete from $\mathcal{S}$ all S such that $|S \Delta S'| \leq k$.

We next introduce an useful set-theoretic lemma, whose proof is omitted from this version of the paper.

Lemma 2.4 *Let R, G, B, R', G' and B' be finite sets. Then $(R \cap G \cap B) \Delta (R' \cap G' \cap B') \subseteq (R \Delta R') \cup (G \Delta G') \cup (B \Delta B')$.*

We are now ready to prove our result on triangles. Let V be a fixed set of n points on the plane. Let the family $\mathcal{T}$ consist of the distinct subsets of V corresponding to all possible triangles.

Theorem 2.5 *The maximum size of a k-separated family $\mathcal{T}' \subseteq \mathcal{T}$ of triangles is $\Theta(n^6/k^6)$.*

Proof : (Sketch) We shall only prove the upper bound. Let $\mathcal{T}' \subseteq \mathcal{T}$ be the largest k-separated family. Let the set of triangles that realize this family be $T_1', T_2', \ldots, T_m'$. Each triangle is the intersection of three half-planes; color the three half-planes as red, green and blue. Thus each T_i' becomes a triple of colored half-planes (R_i', G_i', B_i'). Note that each half-plane in the triple represents a subset of V, but this subset might contain many points not in T_i.

Consider the three families of half-planes, $\mathcal{R}' = \{R_1', \ldots, R_m'\}$, $\mathcal{G}' = \{G_1', \ldots, G_m'\}$ and $\mathcal{B}' = \{B_1', \ldots, B_m'\}$. Note that these three families may not be k-separated by themselves. Let us apply the greedy algorithm to each of $\mathcal{R}'$, $\mathcal{G}'$ and $\mathcal{B}'$ to get corresponding $k/6$-separated $k/6$-covers $\mathcal{R}''$, $\mathcal{G}''$ and $\mathcal{B}''$. ¿From Theorem 2.3 we know that each cover is $O(n^2/k^2)$ in size. Let $\mathcal{R}'' \times \mathcal{G}'' \times \mathcal{B}''$ represent the Cartesian product of these three families. Thus $|\mathcal{R}'' \times \mathcal{G}'' \times \mathcal{B}''| = O(n^6/k^6)$. For every triangle $T' = (R', G', B')$ in $\mathcal{T}'$, there exists a triple (R'', G'', B'') in $\mathcal{R}'' \times \mathcal{G}'' \times \mathcal{B}''$, such that $|R' \Delta R''| \leq k/3$, $|G' \Delta G''| \leq k/3$, and $|B' \Delta B''| \leq k/3$. Let one such triple (R'', G'', B'') be defined as the *representative* of T'.

We claim that every T' in $\mathcal{T}'$ has a unique representative. To prove this, consider any pair of subsets $T_i' = (R_i', G_i', B_i')$ and $T_j' = (R_j', G_j', B_j')$ in $\mathcal{T}'$. We know that $|T_i' \Delta T_j'| > k$. If T_i' and T_j' have the same representative, say (R'', G'', B''), then by applying Lemma 2.4 we get $|T_i' \Delta (R'' \cap G'' \cap B'')| \leq k/2$, and $|T_j' \Delta (R'' \cap G'' \cap B'')| \leq k/2$. Thus we get $|T_i' \Delta T_j'| \leq k$, which is a contradiction.

Thus, $|\mathcal{T}'| \leq |\mathcal{R}'' \times \mathcal{G}'' \times \mathcal{B}''| = O(n^6/k^6)$. $\square$

3 An $O(n^3)$-Time Exact Algorithm for Time-Series Similarity

Let $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$ be two sequences. Let $0 < \epsilon < 1$ be a real constant and $\delta > 0$ an integer constant. In this section we shall describe an exact algorithm to compute $Sim_{\epsilon,\delta}(X, Y)$.

Theorem 3.1 *$Sim_{\epsilon,\delta}(X, Y)$ can be computed in $O(n^3\delta^3)$ time.*

Proof : (Sketch) Consider any linear transformation $f : y = ax + b$. When plotted on the xy-plane, f is a straight line which meets the x-axis at point $p = (-b/a, 0)$. Consider two other linear functions, $y = (ax + b)(1 + \epsilon)$ and $y = (ax + b)/(1 + \epsilon)$. These are also straight lines, and both meet the x-axis at p. Let us define the region between these two lines as the *double-wedge of f*, denoted as DW_f (note: these double-wedges are not to be confused with the double-wedges of the previous section). We observe that DW_f contains the line f.

Let $V = \{(x_i, y_j) | x_i \in X, y_j \in Y, |i - j| \leq \delta\}$ be a set of points in the xy-plane. Clearly $|V| = O(n\delta)$. Let $V_f \subseteq V$ be those points inside DW_f.

Suppose $f' : y = a'x + b'$ is a different linear transformation. As was done with f, we can correspondingly define $DW_{f'}$ and $V_{f'}$. We make an important observation: if $V_f = V_{f'}$, then $S_{f,\epsilon,\delta}(X, Y) = S_{f',\epsilon,\delta}(X, Y)$. In other words, even if both are different functions (i.e different straight lines), the LCSS algorithm will produce the same output when run on both. Our task then, is to determine a finite family of linear functions that *cover* all linear functions, i.e. for every function there should be a corresponding *representative* function in the finite family such that their corresponding double-wedges contain identical subsets of V.

Instead of working with double-wedges, we will reformulate the problem using the equivalent concept of *stabbed sets*, which is easier to analyze. For every point $p = (x_i, y_j) \in V$, consider a vertical line segment $L_p = (p_1, p_2)$, where $p_1 = (x_i, y_j(1+\epsilon))$ and $p_2 = (x_i, y_j/(1+\epsilon))$. Let $R = \{L_p | p \in V\}$. Consider the linear transformation $f : y = ax + b$. Let R_f be the segments of R stabbed by the infinite line f. The following lemma establishes the equivalence between R_f and V_f. The proof is easy, and is omitted from this version of the paper.

Lemma 3.2 *$R_f = \{L_p | p \in V_f\}$.*

Let V_R be the set of end points of all vertical segments in R. Thus $|V_R| = 2|V| = O(n\delta)$. Consider any linear function f that does not pass through any point in V_R. Clearly if we perturb f slightly, R_f will not change. Recall that $\mathcal{L}$ is the (infinite) family of all linear functions. Let $\mathcal{L}'$ be the family of linear functions such that $f' \in \mathcal{L}'$ iff f' passes through two points of V_R. It is easy to see that for *any* linear function $f \in \mathcal{L}$, there is a function $f' \in \mathcal{L}'$ such that $V_f = V_{f'}$.

Thus, our algorithm first computes $\mathcal{L}'$ (whose size is $O(n^2\delta^2)$), then runs LCSS on each function, and finally outputs the normalized length of the longest sequence found. This gives a total running time of $O(n^3\delta^3)$. This concludes the proof of Theorem 3.1. $\square$

It is instructive to study the above algorithm from the *duality* point of view (see [5, 8]). Recall that the dual of a line segment is a double-wedge in the ab-plane. Imagine computing the duals of each segment in R, and computing the arrangement, A, of these double-wedges. Any linear function f corresponds to a point in the ab-plane. In particular, the duals of the functions in $\mathcal{L}'$ correspond to vertices of A. In a sense, our exact algorithm implicitly generates all the vertices of A.

The arrangement A will have a significant role in our other algorithms.

4 An $O(n^2)$-Time Approximation Algorithm

In this section we shall describe an approximation algorithm to compute $Sim_{\epsilon,\delta}(X,Y)$.

Theorem 4.1 *Let $0 < \beta < 1$ be any desired small constant. A linear transformation f and the corresponding $S_{f,\epsilon,\delta}(X,Y)$ can be computed in $O(n^2\delta^2 + n\delta^3/\beta^2)$ time such that $Sim_{\epsilon,\delta}(X,Y) - S_{f,\epsilon,\delta}(X,Y) \leq \beta$.*

Proof : (Sketch) Recall the definitions of the previous section, especially that of V, R, V_R, $\mathcal{L}'$, and A. Let f and f' be any two linear functions such that $|V_f \Delta V_{f'}| \leq \beta n$. It is not hard to see that $S_{f',\epsilon,\delta}(X,Y) \geq S_{f,\epsilon,\delta}(X,Y) - \beta$. Our approximation algorithm first computes a finite family of linear functions $\mathcal{L}''$, such that for *any* linear function f, there is $f'' \in \mathcal{L}''$ such that $|V_f \Delta V_{f''}| \leq \beta n$. It then runs LCSS on each function in $\mathcal{L}'$, and finally outputs the normalized length of the longest sequence found.

Compute the arrangement A as defined in the previous section (this can be done in $O(n^2\delta^2)$ time, see [5]).

We shall construct the family $\mathcal{L}''$ as follows. Within each face of A, select a representative point. Build the planar dual graph of A, say $Dual(A)$, where these representative points become vertices. Let P be an initially empty set of points. While $Dual(A)$ is not empty, repeat the following steps: add any vertex p in $Dual(A)$ to P, then for all $p' \in Dual(A)$ such that $dist_{Dual(A)}(p,p') \leq k$, remove p' and associated edges from $Dual(A)$. (The function $Dist$ computes the number of edges on the shortest path in $Dual(A)$ between p and p'). Let $\mathcal{L}''$ be the family of functions whose duals are the points in P. It should be clear that for *any* linear function f, there is $f'' \in \mathcal{L}''$ such that $|V_f \Delta V_{f''}| \leq \beta n$. The above steps can be implemented in time linear in the size of $Dual(A)$ (i.e. $O(n^2\delta^2)$), essentially by making breadth-first searches from each p added to P.

All that remains is to estimate the size of $\mathcal{L}''$. Let $\mathcal{S}'' = \{R_{f'} | f' \in \mathcal{L}''\}$, i.e. each set in $\mathcal{S}''$ represents the segments of R stabbed by a function in $\mathcal{L}''$. It is not hard to see that $\mathcal{S}''$ is k-separated. By invoking Theorem 2.2, we get $|\mathcal{S}''| = |\mathcal{L}''| = O((n^2\delta^2)/(n^2\beta^2)) = O(\delta^2/\beta^2))$. We have to run LCSS on each of these functions. This phase takes time $O(n\delta^3/\beta^2)$. When combined with the time taken to compute $A(H)$, we get an overall running time of $O(n^2\delta^2 + n\delta^3/\beta^2)$.

This concludes the proof of Theorem 4.1. □

5 An $O(n)$-Time Randomized Approximation Algorithm

In this section we give a simple randomized approximation algorithm to compute $Sim_{\epsilon,\delta}(X,Y)$.

Theorem 5.1 *Let $0 < \beta < 1$ be any desired small constant. A linear transformation f and the corresponding $S_{f,\epsilon,\delta}(X,Y)$ can be computed in $O(n\delta^3/\beta^2)$ expected-time such that $Sim_{\epsilon,\delta}(X,Y) - S_{f,\epsilon,\delta}(X,Y) \leq \beta$*

Proof : (Sketch) Consider a procedure which works as follows. It selects two points at random from V_R, and defines the function f that passes through these two points. It then computes $S_{f,\epsilon,\delta}(X,Y)$. We shall show that $Prob(Sim_{\epsilon,\delta}(X,Y) - S_{f,\epsilon,\delta}(X,Y) \leq \beta) = \Omega(\beta^2/\delta^2)$. Let M be an optimal function, i.e. let $S_{M,\epsilon,\delta}(X,Y) = Sim_{\epsilon,\delta}(X,Y))$. Let the dual of M belong to face r in the arrangement A. Using Lemma 2.1, we know that the number of vertices of A within $N_{\beta n}(r)$ is $\Omega(n^2\beta^2)$. The procedure is essentially selecting at random one of the $O(n^2\delta^2)$ vertices of A. The probability that the vertex selected is within $N_{\beta n}(r)$ is therefore $\Omega(n^2\beta^2/n^2\delta^2) = O(\beta^2/\delta^2)$.

Our randomized algorithm iterates the above procedure a certain number of times, and outputs the normalized length of the longest subsequence computed thus far. To get the claimed approximation bounds, the expected number of iterations should be $O(\delta^2/\beta^2)$. Each iteration requires running the LCSS algorithm which takes $O(n\delta)$ time. Thus the expected running time of the randomized algorithm is $O(n\delta^3/\beta^2)$.

This concludes the proof of Theorem 5.1. $\square$

6 Experimental results

We experimented with the exact algorithm and the randomized approximation algorithm using three collections of sequences. One consisted of quaterly indicators of the status of the Finnish economy (67 sequences, 85 points), another of measurements of traffic data, error counts and call counts at 15 minute intervals (17 different phone lines, i.e., 51 sequences, 478 points each), and the third one about stock prices at the NYSE. Especially in the phone line data outliers are truly a problem: there are values in the sequence that differ by a factor of 2–5 from all the other values. In most cases these are outliers, but in some cases not; removing them permanently from the data is not possibly.

Overall, the algorithms behaved as predicted by the theoretical analysis. The exact $O(n^3)$ algorithm was far too slow to use for any but the smallest sequences and displacements. The randomized algorithm proved to be very efficient. Moreover, it produced approximations to the true similarity that are very close to the correct values. For example, in Table 1 we see that for varying ϵ, the randomized algorithm got to within 1 from the true optimum already after 500 randomly chosen wedges. Table 2 shows the time needed for this analysis. We see that for displacement of 0, the exact algorithm takes about 360 seconds, and for a displacement of 1, about 1 hour. (All timings are on a Pentium machine with 32 MB of main memory.) The randomized algorithm works extremely fast, checking 500 random wedges in about 6.5 seconds, and this time is almost independent of the displacement. The results for the other sequence types were similar, and they are omitted from this version of the paper.

			K						
ϵ	δ	LCSS	10	25	50	100	200	500	1000
0.10	0	85	80.6	82.0	83.1	83.4	83.2	84.0	84.0
0.10	1	85	79.2	81.6	83.2	83.7	83.9	84.7	84.9
0.05	0	81	77.7	79.5	80.1	80.2	80.2	80.5	80.9
0.05	1	81	78.1	79.3	80.3	80.3	80.6	81.0	80.9
0.01	0	56	49.5	53.3	53.4	53.8	54.7	55.3	55.3
0.01	1	57	50.2	51.8	52.5	53.6	54.4	54.8	55.6

Table 1: True similarity between sequences and the length of longest common subsequence found by using K randomly chosen linear functions; averages over 10 trials. Data: two series of 85 points about the Finnish national economy.

				K						
ϵ	δ	LCSS	exact	10	25	50	100	200	500	1000
0.10	0	85	361	0.16	0.36	0.67	1.3	2.6	6.5	13.0
0.10	1	85	3211	0.15	0.36	0.68	1.3	2.7	6.5	13.0
0.05	0	81	358	0.15	0.35	0.67	1.4	2.6	6.5	13.0
0.05	1	81	3236	0.14	0.30	0.57	1.1	2.6	5.6	11.3
0.01	0	56	364	0.15	0.36	0.67	1.3	2.6	6.5	13.0
0.01	1	57	3269	0.11	0.30	0.58	1.1	2.3	5.6	11.2

Table 2: Time used by the exact algortithm and the randomized approximate algorithm (for K randomly chosen linear functions; averages over 10 trials). Data: two series of 85 points about the Finnish national economy.

7 Conclusions

We have addressed the problem of defining (and determining) the similarity between two time series. The algorithms we present are based on a careful investigation of the geometric properties of the problem, which are interesting in their own right. At the same time, the algorithms are very efficient in practice. We intend to focus our future efforts in the directions summarized below.

In the full paper we will discuss how the algorithms can be extended to allow variations in our similarity model, such as considering different function families, tolerance models, etc.

We plan to investigate further some of the practical issues of the implementations. For example, preliminary experiments indicate that a combination of our statistical methods (see [7]) with the methods of this paper result in hybrid algorithms that seem to work very well.

Interesting fundamental questions still remain. It is not clear whether any of the algorithms are optimal w.r.t. running time. For example, an $o(n^3)$ time exact algorithm would be a remarkable breakthrough. Similarly, in case of the approximation algorithm, is it necessary to first compute the entire arrangement A, and then look for the $O(1)$-sized family $\mathcal{L}''$? Another theoretical question is, what is the maximum size of a k-separated family of convex sets?

Finally, an interesting related data retrieval problem is the *sequence database query problem*: given a database of N sequences and a query sequence, find out which sequences in the database are similar to the query sequence. It is desirable to have algorithms that preprocess the database so that a linear scan can be avoided at query time. Several solutions exist (see [1, 2]), and it is intriguing whether our techniques can be extended to this problem.

References

[1] R. Agrawal, C. Faloutsos and A. Swami. Efficient Similarity Search in Sequence Databases. In *Proc. of the 4th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO'93)*, 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 *Proc. of the 21st Intl. Conf. on Very Large Data Bases (VLDB'95)*, pp 490–501.

[3] A. V. Aho. Algorithms for Finding Patterns in Strings. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, Elsevier, 1990, pp. 255–400.

[4] B. Bollobas. Combinatorics. Cambridge University Press, 1986.

[5] B. Chazelle, L. J. Guibas and D. T. Lee. The Power of Geometric Duality. In *Proc. of IEEE FOCS*, 1983, pp. 217–225.

[6] T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms. The MIT Press, 1990, pp. 314–319.

[7] G. Das, D. Gunopulos and H. Mannila. Finding Similar Time Series. Manuscript, 1996.

[8] H. Edelsbrunner. Algorithms in Combinatorial Geometry. Springer-Verlag, 1987.

[9] C. Faloutsos, M. Ranganathan and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In *SIGMOD'94*, 1994.

[10] H.V. Jagadish, A. O. Mendelzon and T. Milo. Similarity-Based Queries. In *Proc. of 14th Symp. on Principles of Database Systems (PODS'95)*, 1995, pp. 36–45.

[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, UC Davis, 1996.

[13] N. Yazdani and Z. M. Ozsoyoglu. Sequence Matching of Images. In *Proc. of the 8th Intl. Conf. on Scientific and Statistical Database Management*, 1996, pp. 53–62.