Efficient Similarity Search In Sequence Databases

Rakesh Agrawal *Christos Faloutsos** *Arun Swami*

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Abstract. We propose an indexing method for time sequences for processing similarity queries. We use the Discrete Fourier Transform (DFT) to map time sequences to the frequency domain, the crucial observation being that, for most sequences of practical interest, only the first few frequencies are strong. Another important observation is Parseval's theorem, which specifies that the Fourier transform preserves the Euclidean distance in the time or frequency domain. Having thus mapped sequences to a lower-dimensionality space by using only the first few Fourier coefficients, we use R^*-trees to index the sequences and efficiently answer similarity queries. We provide experimental results which show that our method is superior to search based on sequential scanning. Our experiments show that a few coefficients (1-3) are adequate to provide good performance. The performance gain of our method increases with the number and length of sequences.

*On sabbatical from the Dept. of Computer Science, University of Maryland, College Park. This research was partially funded by the Systems Research Center (SRC) at the University of Maryland, and by the National Science Foundation under Grant IRI-8958546 (PYI), with matching funds from EMPRESS Software Inc. and Thinking Machines Inc.

1 Introduction

Sequences constitute a large portion of data stored in computers. There have been several efforts to model time-sequenced data, to design languages to query such data, and to develop access structures to efficiently process such queries (see [25] for a bibliography). Most of the work, however, has focussed on "exact" queries. New emerging applications, particularly database mining applications [2], require that databases be enhanced with the capability to process "similarity" queries. The following are some examples of the similarity queries over sequence databases:

- Identify companies with similar pattern of growth.
- Determine products with similar selling patterns.
- Discover stocks with similar movement in stock prices.
- Find if a musical score is similar to one of the copyrighted scores.

Similarity queries can be classified into two categories:

a. *Whole Matching.* The sequences to be compared have the same length

n.

b. *Subsequence Matching.* The query sequence is smaller; we look for a subsequence in the large sequence that best matches the query sequence.

We concentrate on whole matching, and present an indexing technique that can be used to efficiently process such queries. Within the whole matching case, we consider the following problems:

a1. *Range Query.* Given a query sequence, find sequences that are similar within distance ϵ.

a2. *All-Pairs Query* (or 'spatial join'). Given N sequences, find the pairs of sequences that are within ϵ of each other.

The parameter ϵ is a distance parameter that controls when two sequences should be considered similar. It could be either user-defined, or determined automatically (eg., ϵ=10% of the 'energy' of the query sequence; see Eq. 3 for the definition of 'energy').

Approximate matching has been attracting increasing interest lately. Motro described a user interface for vague queries [18]. Shasha and Wang [24] proposed an indexing method that uses the triangular inequality and some precomputed distances to prune the search. However, the space overhead of the method seems quadratic on the number of objects, which may make it prohibitive for large databases. Aurenhammer [5] surveyed recent research on Voronoi diagrams, along with their use for nearest neighbor queries. Although Voronoi diagrams work well for approximate matches in 2-dimensional spaces, they need intricate transformations to work for a 3-d space, and they do not work at all for higher dimensionalities. Jagadish [15] suggested using a few minimum bounding rectangles to extract features from shapes and subsequently managing the resulting vectors using a spatial access method, like k-d-B-trees, grid files, etc.

For numerical sequences, we propose extracting k features from every sequence, mapping it to k-dimensional space, and then using a multidimensional index to store and search these points. The multidimensional indexing methods currently in use are R^*-trees [6] and the rest of the R-tree and k-d-Btree family [12, 14, 16]; linear quadtrees [22]; and grid-files [19]. There are two subtle problems with this approach that must be addressed:

- *Completeness of feature extraction*: How to extract features, and how to guarantee that we do not miss any qualifying object (time sequence, in our case). To guarantee no "false dismissal", objects should be mapped to points in k-dimensional space such that the Euclidean distance in the k-dimensional space is *less than or equal* to the real distance between the two objects.
- *Dimensionality "curse"*: Most multidimensional indexing methods scale exponentially for high dimensionalities, eventually reducing to sequential scanning. For linear quadtrees, the effort is proportional to the hyper surface of the query region [13]; the hyper surface grows exponentially with the dimensionality. Grid files face similar problems, since they require a directory that grows exponentially with the dimensional-

ity. The R-tree based methods seem to be most robust for higher dimensions, provided that the fanout of the R-tree nodes remains > 2. Experiments [21] indicate that R^*-trees work well for up to 20 dimensions. The feature extraction method should therefore be such that a few features are sufficient to differentiate between objects.

We propose to use the *Discrete Fourier Transform* [20] for feature extraction. Given a sequence, we transform it from the time domain to the frequency domain. We then index only on the first few frequencies, dropping all other frequencies. This approach addresses the two problems cited above as follows:

- *Completeness of feature extraction:* Parseval's theorem [20], discussed in Section 2, guarantees that the distance between two sequences in the frequency domain is the same as the distance between them in the time domain.
- *Dimensionality curse:* As we discuss in subsection 3.3, a large family of interesting sequences exhibit strong amplitudes for the first few frequencies. Using the first few frequencies then avoids the dimensionality problem, while still introducing few *false hits*. The false hits are removed in a post-processing step.

The organization of the rest of the paper is as follows. Section 2 gives some background material on the Discrete Fourier Transform, and introduces Parseval's theorem that provides the basis for the indexing technique we propose. A resume of our indexing technique is given in Section 3. We also justify our choice of similarity measure and the selection of DFT for feature extraction in this section. Section 4 contains performance experiments that empirically show the effectiveness of our technique. We conclude with a summary in Section 5.

2 Discrete Fourier Transform

We start with a brief overview of the Discrete Fourier Transform (DFT). The importance of the DFT is the existence of a fast algorithm, the *Fast Fourier Transform (FFT)*, that can calculate the DFT coefficients in $O(n \log n)$ time. Further information on the Fourier transform can be found in any digital signal processing textbook, for example, [20].

The n-point *Discrete Fourier Transform* [20] of a signal $\vec{x} = [x_t]$, $t = 0, \ldots, n-1$ is defined to be a sequence $\vec{X}$ of n complex numbers X_f, $f = 0, \ldots, n-1$, given by

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t \exp\left(-j 2\pi f t / n\right) \quad f = 0, 1, \ldots, n-1 \quad (1)$$

where j is the imaginary unit $j = \sqrt{-1}$. The signal $\vec{x}$ can be recovered by the inverse transform:

$$x_t = 1/\sqrt{n} \sum_{f=0}^{n-1} X_f \exp\left(j 2\pi f t / n\right) \quad t = 0, 1, \ldots, n-1 \quad (2)$$

X_f is a complex number (with the exception of X_0, which is a real, if the signal $\vec{x}$ is real). There are some minor discrepancies among books: some define $X_f = 1/n \sum_{t=0}^{n-1} \ldots$ or $X_f = \sum_{t=0}^{n-1} \ldots$. We have followed the definition in (Eq 1), for it simplifies the upcoming Parseval's theorem (Eq 4).

Definitions: For a complex number $c = a + jb = A\exp(j\phi)$

- $A \equiv |c|$ is said to be the *amplitude* and ϕ to be the *phase* of the number c.
- The *conjugate* c^* of c is defined as $a - jb$.
- The *energy* $E(c)$ of c is defined as the square of the amplitude ($E(c) \equiv |c|^2 \equiv c\, c^*$).
- The *energy* $E(\vec{x})$ of a sequence $\vec{x}$ is defined as the sum of energies at every point of the sequence:

$$E(\vec{x}) \equiv \| \vec{x} \|^2 \equiv \sum_{t=0}^{n-1} |x_t|^2 \tag{3}$$

A fundamental observation for this paper is Parseval's theorem [20]:

Theorem 1 (Parseval) *Let $\vec{X}$ be the Discrete Fourier Transform of the sequence $\vec{x}$. Then we have*

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2 \tag{4}$$

That is, the energy in the time domain is the same as the energy in the frequency domain.

The Discrete Fourier Transform inherits the following properties from the continuous Fourier transform. Let '$\Longleftrightarrow$' indicate Fourier pairs, i.e.,

$$[x_t] \Longleftrightarrow [X_f] \tag{5}$$

means that $[X_f]$ is the Discrete Fourier Transform of $[x_t]$. The Discrete Fourier Transform is a *linear transformation*: If

$$[x_t] \Longleftrightarrow [X_f];\, [y_t] \Longleftrightarrow [Y_f] \tag{6}$$

then

$$[x_t + y_t] \Longleftrightarrow [X_f + Y_f] \tag{7}$$

$$[a x_t] \Longleftrightarrow [a X_f] \tag{8}$$

Also, a shift in the time domain changes only the phase of the Fourier coefficients, but not the amplitude.

$$[x_{t-t_0}] \Longleftrightarrow [X_f \ \exp(2\pi f t_0 j/n)] \tag{9}$$

Given the above, Parseval's theorem gives

$$\| \vec{x} - \vec{y} \|^2 \equiv \| \vec{X} - \vec{Y} \|^2 \tag{10}$$

The latter implies that the Euclidean distance between two signals $\vec{x}$ and $\vec{y}$ in the time domain is the same as their Euclidean distance in the frequency domain.

We believe that for a large number of time sequences of practical interest, there will be a few frequencies with high amplitudes. Thus, if we index only on the first few frequencies, we shall have few *false hits*. This is *a key observation* for our proposed method.

3 Proposed Technique

We propose using the square root of the *sum of squared differences* as the distance function between two sequences. Specifically, the distance $\mathcal{D}(\vec{x}, \vec{y})$ between two sequences $\vec{x}$ and $\vec{y}$ is the square root of the energy of the difference:

$$\mathcal{D}(\vec{x}, \vec{y}) \ \equiv \ (\sum_{t=0}^{n-1} |x_t - y_t|^2)^{1/2} \ \equiv \ (E(\vec{x} - \vec{y}))^{1/2} \tag{11}$$

If this distance is below a user-defined threshold ϵ, we say that the two sequences are similar.

The importance of Parseval's theorem (Eq 4) is that it allows to translate the query from the time domain to the frequency domain. Coupled with the conjecture that few Fourier coefficients are enough, it allows us to build an effective index with a low dimensionality.

The following is a resume of our proposed technique:

1. Obtain the coefficients of the Discrete Fourier Transforms of every sequence in the database.

2. Build a multidimensional index using the first f_c Fourier coefficients, where f_c stands for 'cut-off frequency'. Thus, each sequence becomes a point in a $2f_c$-dimensional space (recall that the Fourier coefficients are complex numbers). We discuss in subsection 3.3 why f_c can be taken to be small (< 5). As discussed earlier, we recommend the R^*-trees as the indexing structure, since it has been shown to work well for at least up to 20 dimensions [21]. This index will be called 'F-index' henceforth.

3. For a range query, obtain the first f_c Fourier coefficients of the query sequence. Use the F-index to retrieve the set of matching sequences that are at most ϵ distance away from the query sequence.

4. For an all-pairs query, we do a spatial join using the F-index. The result of the join will be a superset of the answer set.

5. The actual answer set is obtained in a post-processing step in which the actual distance between two sequences is computed in the time domain and only those within ϵ distance are accepted.

The 'completeness' of this method is based on the following lemma:

Lemma 1 *The F-index introduces no false dismissals.*

We only give the proof for range queries; the proof for 'all-pairs' queries is very similar. Suppose we want all sequences $\vec{x}$ that are similar to a query sequence $\vec{q}$, within distance ϵ, *i.e.*:

$$\mathcal{D}(\vec{x}, \vec{q}) \leq \epsilon \tag{12}$$

or, equivalently:

$$\| \vec{x} - \vec{q} \|^2 = \sum_{t=0}^{n-1} |x_t - q_t|^2 \leq \epsilon^2 \tag{13}$$

Using Parseval's theorem (Eqs. 4, 10), we want all $\vec{X}$ such that

$$\| \vec{X} - \vec{Q} \|^2 = \sum_{f=0}^{n-1} |X_f - Q_f|^2 \leq \epsilon^2 \tag{14}$$

Keeping only the first $f_c < n$ coefficients, we have

$$\sum_{f=0}^{f_c-1} |X_f - Q_f|^2 \leq \sum_{f=0}^{n-1} |X_f - Q_f|^2 \leq \epsilon^2 \tag{15}$$

Thus, equation (14) implies the following condition

$$\sum_{f=0}^{f_c-1} |X_f - Q_f|^2 \leq \epsilon^2 \tag{16}$$

In other terms, the condition of (Eq. 16) will retrieve all $\vec{X}$ that are in the answer, plus some false hits. Thus, our index acts as a filter that returns a superset of the answer set.

∎

3.1 Choice of Similarity Measure

The similarity measure is clearly application-dependent. Several similarity measures have been proposed, for

1-d and 2-d signals. In a recent survey for images (2-d signals), Brown [7](p. 367, sect. 4.2) mentions that one of the typical similarity measures is the cross-correlation (which reduces to the Euclidean distance, plus some additive and multiplicative constants). We have chosen the Euclidean distance, because (a) it is useful in many cases, as is (b) it can be used with *any* other type of similarity measure, as long as this measure can be expressed as the Euclidean distance between feature vectors in some feature space.

In fact, the Euclidean distance is the optimal distance measure for estimation [11], if signals are corrupted by Gaussian, additive noise. Thus, if $\vec{q}$ is our query and $\vec{x}$ is a corrupted version of it in the database, a searching method using the Euclidean distance should produce good results.

A valuable feature of the Euclidean distance is that it is preserved under orthonormal transforms. Other distance functions, like the L_p norms

$$L_p(\vec{x}, \vec{y}) = (\sum |x_t - y_t|^p)^{1/p} \tag{17}$$

do not have this property, unless $p = 2$ (because $L_2 \equiv$Euclidean distance).

3.2 Using DFT

Having decided on the Euclidean distance as the distance measure, we would like a transform that (a) preserves the distance (b) is easy to compute and (c) concentrates the energy of the signal in few coefficients.

The distance-preservation requirement is met by any *orthonormal transform* [10], DFT being one of them. Orthonormal transforms form two classes: (1) the data-dependent ones, like the Karhunen-Loeve (K-L) transform, which need all the data signals to determine the transformation matrix and (2) the data-independent ones, like the DFT, Discrete Cosine (DCT), Harr, or wavelet transform, where the transformation matrix is determined a-priori.

The data-dependent transforms can be fine-tuned to the specific data set, and therefore they can achieve better performance, concentrating the energy into fewer features in the feature vector. Their drawback is that, if the data set evolves over time, e.g., a recomputation of the transformation matrix may be required to avoid performance degradation, requiring expensive data reorganization. We, therefore, favor data-independent transforms.

Among them, we have chosen the DFT because it is the most well known, its code is readily available and it does a good job of concentrating the energy in the first few coefficients, as we shall see next. In addition, the DFT has the attractive property that the *amplitude* of the Fourier coefficients is invariant under shifts (Eq. 9). Thus, using Fourier transforms for feature extraction has the potential that our technique can be extended to finding similar sequences ignoring shifts.

Note that our approach can be applied with *any* orthonormal transform. In fact, our response time will improve with the ability of the transform to concentrate the energy: the fewer the coefficients that contain most of the energy, the faster our response time. Thus, the performance results presented next are just pessimistic bounds; better transforms will achieve even better response times.

3.3 Using Few Fourier Coefficients for Indexing

Using a small value for the number of Fourier coefficients retained f_c does not affect the correctness — the F-index is a filter that returns a superset of the answer set. However, our proposed technique will not be very effective if the choice of a small f_c results in a large number of false hits.

The worst-case signal for our method is *white noise*, where each value x_t is completely independent of its neighbors x_{t-1}, x_{t+1}. The energy spectrum of white noise follows $O(f^0)$ [23], that is, it has the same energy in every frequency. This is bad for the F-index, because it implies that all the frequencies are equally important. However, we have strong reasons to believe that real signals have a skewed energy spectrum. For example, *random walks* (also known as *brown noise* or *brownian walks*) exhibit an energy spectrum of $O(f^{-2})$ [23], and therefore an amplitude spectrum of $O(f^{-1})$. Stock movements and exchange rates have been successfully modeled as random walks (e.g., [8, 17]). Using the data set available through ftp from *sfi.santafe.edu*, we show in [1] that the Fourier transform of the movement of the exchange rate between the Swiss franc and the US dollar follows closely the same $1/f$ behavior as for a random walk.

Our mathematical argument for keeping the first few Fourier coefficients agrees with the intuitive argument of the Dow Jones theory for stock price movement (see, for example, [9]). This theory tries to detect *primary* and *secondary* trends in the stock market movement, and ignores *minor* trends. Primary trends are defined as changes that are larger than 20%, typically lasting more than a year; secondary trends show 1/3-2/3 relative change over primary trends, with a typical duration of a few months; minor trends last roughly a week. From the above definitions, we conclude that primary and secondary trends correspond to strong, low frequency signals while minor trends correspond to weak, high frequency signals. Thus, the primary and secondary trends are exactly the ones that our method will automatically choose for indexing.

In addition to stock movements and exchange rates, it is believed that several families of real signals are not white noise. For example, 2-d signals, like photographs, are far from white noise, exhibiting a few strong coefficients in the lower spatial frequencies. The JPEG image compression standard [26] exactly exploits this phenomenon, effectively ignoring the high-frequency components of the Discrete Cosine Transform, which is closely related to the Fourier transform. If the image consisted of white noise, no compression would be possible at all. Birkhoff's theory [23] claims that *'interesting'* signals, such as musical scores and other works of art, consist of *pink noise*, whose energy spectrum follows $O(f^{-1})$. The argument of the theory is that white noise with $O(f^0)$ energy spectrum is completely unpredictable, while brown noise with $O(f^{-2})$ energy spectrum is too predictable and therefore boring. The energy spectrum of pink noise lies in-between. Signals with pink noise also have their energy concentrated in the first few frequencies (but not as few as in the random walk). In addition to the above, there is another group of signals, called *black noise* [23]. Their energy spectrum follow $O(f^{-b})$, $b > 2$, which is even more skewed than the spectrum of the brown noise. Such signals model successfully, for example, the water level of rivers as they vary over time [17].

4 Performance Experiments

To determine the effectiveness of our proposed method (the *F-index* method), we compared it to a *sequential scanning* method. We used the R^*-tree for the index. For range queries, the sequential scanning method computes the distance between the query sequence and each data sequence. In our effort to do the best possible implementation for the sequential scanning, we stop the test as soon as the square of the distance exceeds ϵ^2, and we declare the two sequences to be dissimilar. Thus, a data sequence is fully scanned only if it is similar to the query sequence. For 'all-pairs' queries, each sequence in the database is tested against every other sequence, for a total of $N(N-1)/2$ tests.

We investigated the following questions in these experiments:

- How to choose the number of Fourier coefficients to be retained (cut-off frequency f_c) in the F-index method. A larger f_c reduces the false hits but at the same time increases the dimensionality of the R^*-tree, and hence the search time.
- How does the search time grow as a function of number of sequences in the database?
- How does the length n of the sequences affect the performance?

4.1 Experimental setup

We generated synthetic sequences for the experiments. Each sequence $\vec{x} = [x_t]$ was a random walk:

$$x_t = x_{t-1} + z_t \tag{18}$$

where z_t $(t = 1, 2, \ldots)$ are independent, identically distributed (IID) random variables. For implementation convenience, each z_t variable is uniformly distributed in the range (-500, 500). The probability distribution of each z_t is immaterial; the results would be the same had we chosen a gaussian distribution, or a fair, random coin. For each set $\mathcal{S}$ of N sequences, queries were generated by creating a distorted copy $[\tilde{x}_t]$ of each sequence $[x_t]$ in $\mathcal{S}$. This was accomplished by adding a small amount of noise to every x_t, i.e.,

$$\tilde{x}_t = x_t + p\ w_t \tag{19}$$

where p=0.05 and w_t $(t = 1, 2, \ldots)$ are IID random variables, each following a uniform distribution in the range (-500,500).

Let $\mathcal{Q}$ be the set of distorted sequences, which we shall use as queries. For range queries, we search $\mathcal{S}$ for sequences within distance ϵ for every distorted sequence in $\mathcal{Q}$. For all-pairs queries, we concatenate $\mathcal{S}$ and $\mathcal{Q}$, and ask for all sequence pairs within ϵ distance. The execution time for the F-index method includes both the search time in the R^*-tree and the post-processing time.

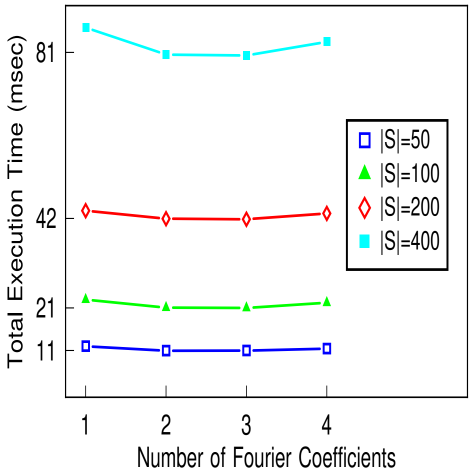

Figure 1: Time per query vs. # Fourier coefficients f_c, for range queries

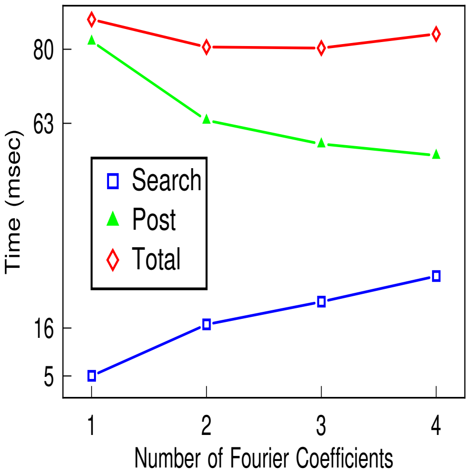

Figure 3: Breakup of the execution time, for range query ($|\mathcal{S}| = 400$)

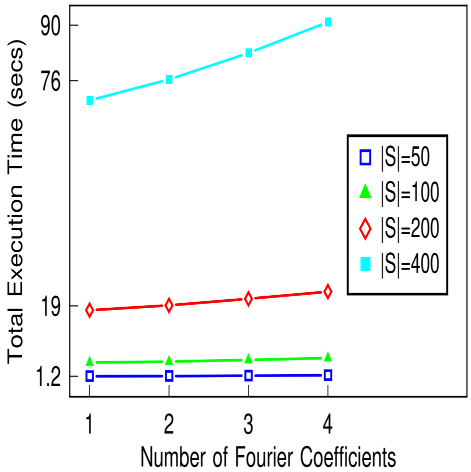

Figure 2: Time per query vs. # Fourier coefficients f_c, for all-pairs queries

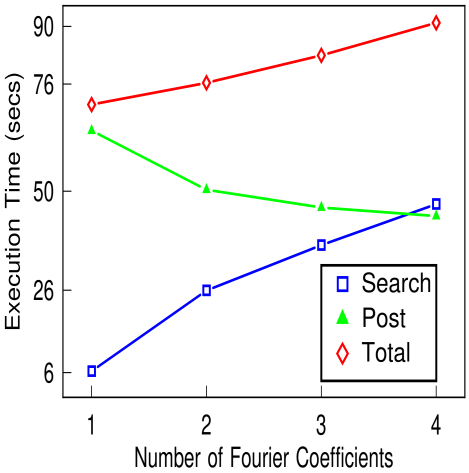

Figure 4: Breakup of the execution time, for all-pairs query ($|\mathcal{S}| = 400$)

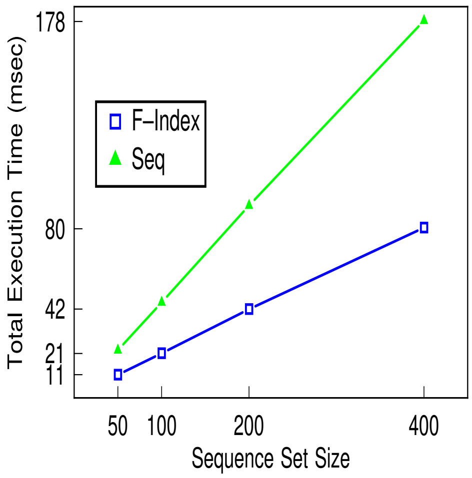

Figure 5: Time per query varying # sequences $|\mathcal{S}|$, for range queries

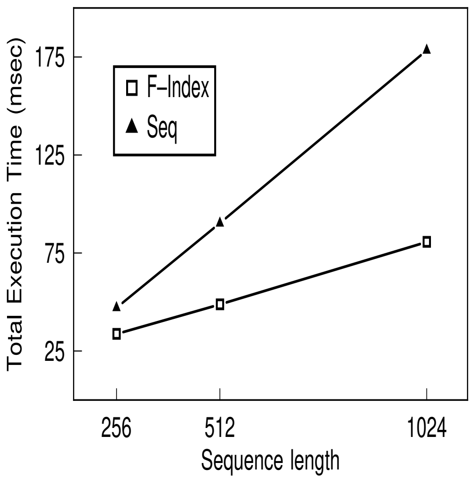

Figure 7: Time per query varying the sequence length n, for range queries

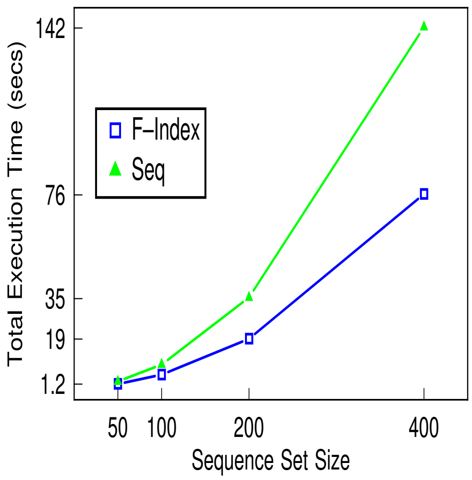

Figure 6: Time per query varying # sequences $|\mathcal{S}|$, for all-pairs queries

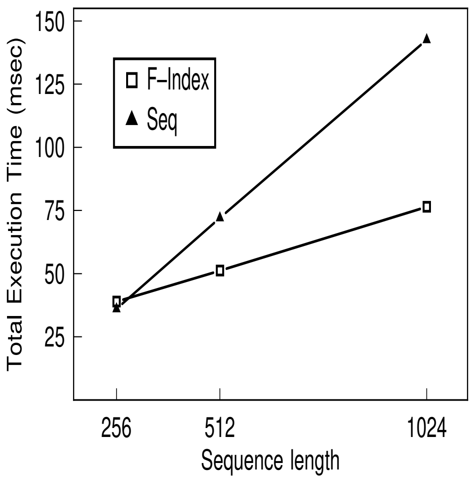

Figure 8: Time per query varying the sequence length n, for all-pairs queries

Parameter	Symbol	Values	Default value
# Fourier coefficients kept	f_c	1, 2, 3, 4	2
# sequences in $\mathcal{S}$	$\lvert\mathcal{S}\rvert$	50, 100, 200, 400	400
Length of each sequence	n	256, 512, 1024, 2048	1024
Distance (tolerance)	ϵ	sqrt($1000 \times n$)	

Table 1: Summary of experimental settings

We repeated each experiment 10 times by generating 10 sequence sets with different seeds, and averaged the execution times from these repetitions. Table 1 summarizes the parameters of the experiments.

4.2 Varying the cut-off frequency f_c

Figures 1 and 2 show the execution time per query for range and all-pairs queries respectively, for different number of sequences in $\mathcal{S}$. Figures 3 and 4 give the the total execution time by the F-index method for the two types of queries, broken into (a) search time in the R^*-tree and (b) post-processing time (where the 'false hits' are eliminated). The latter two graphs have been plotted for $|\mathcal{S}|$ = 400 sequences in $\mathcal{S}$.

As the number of Fourier coefficients ('cut-off frequency' f_c) increases, the dimensionality of the R^*-tree increases. Recall that each Fourier coefficient, being a complex number, increases the dimensionality of the R^*-tree by 2. The increase in dimensionality results in better index selectivity, which gives fewer false hits. This reduction in false hits is reflected in the post-processing time, which decreases with the cut-off frequency. However, the time to search the R^*-tree increases with the dimensionality, because the fanout is smaller, and the tree is taller. Figures 3 and 4 are in complete agreement with the above intuitive arguments.

Given the trade-off between the tree-search time and the post-processing time, it is natural to expect that there is an 'optimal' f_c. Indeed, the total execution time of our method shows such a minimum, as illustrated in Figures 1 and 2. Notice that this minimum is rather flat, and, more importantly, it occurs for small values of the cut-off frequency f_c. This experiment confirms our early conjecture that we can effectively use a small number of Fourier coefficients for indexing sequences.

For the rest of the experiments, we kept f_c=2 Fourier coefficients for indexing, resulting in a 4-dimensional R^*-tree.

4.3 Varying the number of sequences in the database

The next experiment compares the F-index method with the sequential scanning method for increasing number of sequences in the database. Figures 5 and 6 show the execution time per query for range and all-pairs queries respectively, for different values of the number of sequences in $|\mathcal{S}|$. Clearly, the F-index method outperforms the sequential scanning. As the number of sequences increases, the gain of the F-index method increases, making this method even more attractive for large databases.

4.4 Varying the length of sequences

First, we show the results for range queries. We varied the length of sequences, keeping the number sequences in $\mathcal{S}$ fixed to 400. The distance parameter ϵ was set to $(1000 \times n)^{1/2}$ (where n is the length of a sequence). Figure 7 shows the execution time per query for range queries for different sequence lengths. The gain of the F-index method increases with n.

Figure 8 shows the results of the experiments for all-pairs queries. The trends are similar with the ones for range queries.

4.5 Discussion

The major conclusions from our experiments are:

- The minimum in the execution time for both range and 'all-pairs' queries is achieved for a small number of Fourier coefficients ($f_c = 1 - 3$). Moreover, the minimum is rather flat, which implies that a sub-optimal choice for f_c will give search time that is close to the minimum.
- Increasing the number of sequences in the database results in higher gains for our method.
- Increasing the length of the sequences n also results in higher gains for our method.

Thus, the experiments show that the proposed F-index method achieves increasingly better performance, as the volume of the data increases.

Finally, we should mention that we also examined whether a 'naive' feature extraction method would work as well. For example, consider a method that keeps the first few values of each time sequence, and indexes on them. We carried out an experiment in which we indexed on the first 10 values of each time sequence. The performance of this method was very poor compared to the F-index method; there were many false hits, resulting in a large post-processing time. Judging that further details are of little interest, we omit the experimental results.

5 Summary

We proposed a method to index time sequences for similarity searching. The major highlights of this method are:

- The use of an orthonormal transform, and specifically, the Discrete Fourier Transform, to extract features from a sequence. The attractive property of the DFT is that the Euclidean distance in the time domain is preserved in the frequency domain, thanks to Parseval's theorem. Thus, the DFT fulfills the "completeness of feature extraction" criterion. In addition, the DFT is fast to compute ($O(n \log n)$).
- The recognition that a large family of sequences have only a few (f_c) strong Fourier coefficients. For example, random walks, stock price movements, exchange rates, exhibit an amplitude spectrum of O($1/f$). Ignoring the weak coefficients, we introduce a few false hits, but no false dismissals. The importance of this observation is that it avoids the "dimensionality curse" at the expense of a modest post-processing cost. Keeping the first f_c coefficients, each sequence becomes a point in a $2f_c$–dimensional space (recall that the Fourier coefficients are complex numbers).
- The use of spatial access methods, and specifically R^*-trees, to index those points. We believe that R^*-trees are more robust than their competitors, for medium dimensionalities.

Extensive empirical evaluation demonstrated the effectiveness of the proposed method. We generated random walks, which model well stock price movements. The conclusions from our experiments are the following: (a) the execution time of our method shows a rather flat minimum for a small cut-off frequency ($f_c \approx$ 1-3) (b) compared to sequential scanning, our method achieves better gains with increasing number of sequences and increasing length. Thus, our method will be more and more attractive, as the volume of the database increases to Gigabytes and Terabytes.

Although we have made certain choices (Euclidean distance between sequences in time domain for similarity measure, DFT for feature extraction, and R^* tree for maintaining indexes), our technique can be trivially adapted for

- any similarity measure that can be expressed as the Euclidean distance between feature vectors in some feature space
- any distance-preserving (eg., orthonormal) transform (the more the energy concentrated on few coefficients, the faster our response time)
- any multi-dimensional index that performs well for the number of features used for indexing.

Future work could examine the following issues

- Examination of other orthonormal transformations, in addition to the Discrete Fourier Transform.
- Extensions of our approach to 2-d and higher-dimensionality signals (e.g., images), in addition to 1-d signals (time sequences) that we have examined.

The work reported in this paper has been done in the context of the Quest project [2] at the IBM Almaden Research Center. In Quest, we are exploring the various aspects of the database mining problem. Besides the problem of queries over large sequences, some other problems that we have looked into include the enhancement of the database capability with the classification queries [3] and with "what goes together" kinds of association queries [4]. The eventual goal is to build an experimental system that can be used for mining rules embedded in massive databases. We believe that database mining is an important application area, combining commercial interest with intriguing theoretical questions.

Acknowledgements: We thank Myron Flickner for several constructive comments and for his help with Parseval's theorem. We also thank Harpreet Sawhney for pointing out the optimality of the Euclidean distance as the similarity measure under Gaussian, additive noise.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search In Sequence Databases", Research Report, IBM Almaden Research Center, San Jose, California, 1993.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Database Mining: A Performance Perspective", *IEEE Transactions on Knowledge and Data Engineering*, Special issue on Learning and Discovery in Knowledge-Based Databases, (to appear).

[3] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami, "An Interval Classifier for Database Mining Applications", *VLDB 92*, Vancouver, August 1992.

[4] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *ACM SIGMOD*, Washington D.C., May 1993.

[5] F. Aurenhammer, "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure" *ACM Computing Surveys* 23(3):345-405, Sept. 1991.

[6] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles", *ACM SIGMOD*, pages 322–331, May 1990.

[7] L. G. Brown, "A Survey of Image Registration Techniques", *ACM Computing Surveys*, 24(4), pages 325–376, December 1992.

[8] C. Chatfield, *The Analysis of Time Series: an Introduction*, Chapman and Hall, London & New York, 1984, Third Edition.

[9] R. D. Edwards and J. Magee, *Technical Analysis of Stock Trends*, John Magee, Springfield, Massachusetts, 1966, 5th Edition, second printing.

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990, 2nd Edition.

[11] A. Gelb, *Applied Optimal Estimation*, MIT Press, 1986.

[12] A. Guttman, "R-trees: a dynamic index structure for spatial searching", *Proc. ACM SIGMOD*, pages 47–57, June 1984.

[13] G. M. Hunter and K. Steiglitz, "Operations on images using quad trees", *IEEE Trans. on PAMI*, PAMI-1(2):145–153, April 1979.

[14] H. V. Jagadish, "Spatial search with polyhedra", *Proc. Sixth IEEE Int'l Conf. on Data Engineering*, February 1990.

[15] H. V. Jagadish, "A retrieval technique for similar shapes", *Proc. ACM SIGMOD Conf.*, pages 208–217, May 1991.

[16] D. Lomet and B. Salzberg, "The Hb-Tree: a Multiattribute Indexing Method with Good Guaranteed Performance", *ACM TODS*, 15(4), pages 625–658, December 1990.

[17] B. Mandelbrot. *Fractal Geometry of Nature*, W.H. Freeman, New York, 1977.

[18] A. Motro, "VAGUE: A User Interface to Relational Databases that Permits Vague Queries," *ACM Trans. on Information Systems (TOIS)*, 6(3), pages 187–214, July 1988.

[19] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The grid file: an adaptable, symmetric multikey file structure", *ACM TODS*, 9(1):38–71, March 1984.

[20] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

[21] M. Otterman, "Approximate Matching with High Dimensionality R-trees", M.Sc. scholarly paper, Dept. of Computer Science, Univ. of Maryland, College Park, MD, 1992.

[22] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1989.

[23] M. Schroeder, *Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise*, W.H. Freeman and Company, New York, 1991.

[24] D. Shasha and T-L. Wang, "New techniques for best-match retrieval", *ACM TOIS*, 8(2):140–158, April 1990.

[25] R. Stam and R. Snodgrass, "A Bibliography on Temporal Databases", *IEEE Bulletin on Data Engineering*, 11(4), Dec. 1988.

[26] G. K. Wallace "The JPEG Still Picture Compression Standard", *CACM*, 34(4):31–44, April 1991.