Chapter 1

EFFICIENT RETRIEVAL OF SIMILAR TIME SERIES

Davood Rafiei
Department of Computer Science
University of Toronto
drafiei@cs.toronto.edu

Alberto O. Mendelzon
Department of Computer Science
University of Toronto
mendel@cs.toronto.edu

Abstract We propose an improvement of the known DFT-based indexing technique for fast retrieval of similar time series. We use the last few Fourier coefficients in the distance computation without storing them in the index since every coefficient at the end is the complex conjugate of a coefficient at the beginning and as strong as its counterpart. We show analytically that this observation can accelerate the search time of the index by more than a factor of two. This result was confirmed by our experiments, which were carried out on real stock prices and synthetic data.

Keywords: Indexing Time Series, Time Series Retrieval, Similarity Queries, Fourier Transform

Introduction

Time series constitute a large amount of data stored in computers. Examples include stock prices, exchange rates, weather data and biomedical measurements. We are often interested in similarity queries on time-series data [3, 2]. For example, we may want to find stocks that behave in approximately the same way; or years when the temperature patterns in two regions of the world were similar.

There have been several efforts to develop access methods for efficient retrieval of similar time series [1, 6, 12, 15]. Agrawal et al. [1] propose an efficient index structure to retrieve similar time series stored in a database. They map time series into the frequency domain using the Discrete Fourier Transform (DFT) and keep the first few coefficients in the index. Two series are considered similar if their Euclidean distance is less than a user-defined threshold.

In this paper, we propose using the last few Fourier coefficients of a time series in the distance computation, the main observation being that, every coefficient at the end is the complex conjugate of a coefficient at the beginning and as strong as its counterpart. This observation reduces the search time of the index by more than 50% in most cases.

The rest of the paper is organized as follows. In the next section we review some background material on related work and on the discrete Fourier transform. Our proposal on the efficient use of DFT in retrieving similar time series is discussed in Section 2.. In the same section, we present analytical results on the search time improvements of our proposed method. Section 3. discusses the performance results obtained from experiments on real and synthetic data. Section 4. is the conclusion.

1. BACKGROUND

In this section, we briefly review background material on past related work and on the discrete Fourier transform.

1.1 RELATED WORK

There has been follow-up work on the indexing technique proposed by Agrawal et al. [1]. In our earlier work [12, 11], we use this indexing method and propose techniques for retrieving similar time sequences whose differences can be removed by a linear transformation such as moving average, time scaling and inverting. More follow-up work includes the work of Faloutsos et al. [6] on subsequence matching and that of Goldin et al. [7] on normalizing sequences before storing them in the index.

In this paper, we use the indexing technique proposed by Agrawal et al.[1], but in addition to the first few coefficients we also take the last few coefficients into account. Both our analytical results and our experiments show that this observation accelerates the retrieval speed of the index by more than a factor of 2. All follow-up works described earlier benefit from this performance improvement.

There are other related works on time series data. A domain-independent framework for posing similarity queries on a database is developed by Jagadish et al. [8]. The framework has three components: a pattern language, a transformation rule language, and a query language. The framework can be tuned to the needs of time series domain. Yi et al. [15] use time warping as a distance function and present algorithms for retrieving similar time sequences under this function. Finally, Agrawal et al. [3] describe a pattern language called SDL to encode queries about "shapes" found in time sequences.

1.2 DISCRETE FOURIER TRANSFORM

Let a time series be a finite duration signal $\vec{x} = [x_t]$ for $t = 0, 1, \cdots, n-1$. The DFT of $\vec{x}$, denoted by $\vec{X}$, is given by

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t e^{\frac{-j2\pi t f}{n}} \quad f = 0, 1, \cdots, n-1 \tag{1.1}$$

where $j = \sqrt{-1}$ is the imaginary unit. Throughout this paper, unless it is stated otherwise, we use small letters for sequences in the time domain and capital letters for sequences in the frequency domain. The energy of signal $\vec{x}$ is given by the expression

$$E(\vec{x}) = \sum_{t=0}^{n-1} |x_t|^2. \tag{1.2}$$

A fundamental observation that guarantees the correctness of the indexing method for time series data is Parseval's rule [10], which states for a given signal $\vec{x}$ its energy remains the same after DFT, i.e. $E(\vec{x}) = E(\vec{X})$ where $\vec{X}$ is the DFT of $\vec{x}$. Using Parseval's rule and the linearity property of DFT (for example, see Oppenheim and Schafer [10] for details), it is easy to show that the Euclidean distance between two signals in the time domain is the same as their distance in the frequency domain.

$$D^2(\vec{x}, \vec{y}) = E(\vec{x} - \vec{y}) = E(\vec{X} - \vec{Y}) = D^2(\vec{X}, \vec{Y}) \tag{1.3}$$

2. STORAGE AND RETRIEVAL OF SEQUENCES

Given a set of time series data, we can construct an index [1] as follows: find the DFT of each sequence and keep the first few DFT coefficients as the sequence features. Let's assume that we keep the first *k* coefficients. Since all DFT coefficients except the first one are complex numbers, keeping the first k DFT coefficients maps every time series into a point in a $(2k-1)$-dimensional space. These points can be organized in a multidimensional index such as R*-tree[4]. Keeping only the first *k* Fourier coefficients in the index does not affect the correctness because the Euclidean distance between any two points in the feature space is less than or equal to their real distance due to Parseval's rule and the monotonic property of the Euclidean distance. Thus, the index always returns a superset of the answer set. However, the performance of the index mainly depends on the energy concentration within the first k Fourier coefficients of sequences. It turns out that a large class of real world sequences concentrate the energy within the first few coefficients, i.e. they have a skewed energy spectrum of the form $O(F^{-2b})$ for $b \geq 0.5$ where F denotes the frequency. For example, classical music and jazz fall in the class of *pink noise* whose energy spectrum is $O(F^{-1})$ ([14, 13]), stock prices and exchange rates fall in the class of *brown noise* whose energy spectrum is $O(F^{-2})$ ([9, 5]), and the water level of rivers falls in the class of *black noise* for which $b > 1$ ([9, 13]).

To retrieve similar time series stored in the index we may invoke one of the similarity (range, nearest-neighbour, or all-pair) queries. For example, suppose we want to answer a range query using the index, i.e., to find all sequences $\vec{X}$ that are within distance ϵ of a query sequence $\vec{Q}$, or equivalently $D(\vec{X}, \vec{Q}) < \epsilon$. A common approach to answer this query is to build a multidimensional rectangle of side 2ϵ (or a multidimensional circle of radius ϵ) around $\vec{Q}$ and check for an overlap between the query rectangle (circle) and every rectangle in the index. That is, instead of checking $D^2(\vec{X}, \vec{Q}) < \epsilon^2$, we check $|X_f - Q_f|^2 < \epsilon^2$ for $f = 0, \ldots, k-1$. The latter is a necessary (but not sufficient) condition for the former.

The size of the query rectangle has a strong effect on the number of directory nodes accessed during the search process and the number of candidates which includes all qualifying data items plus some false positives (data items whose full database records do not intersect the query region). Our goal here is to reduce the size of the query region, using the inherent properties of DFT, without sacrificing the correctness.

2.1 OUR PROPOSAL

The following lemma is central to our proposal.

Lemma 1 *The DFT coefficients of a real-valued sequence of duration n satisfy $X_{n-f} = X_f^*$ for $f = 1, \ldots, n-1$ where the asterisk denotes complex conjugation*[1].

Proof: See Oppenheim and Schafer [10, page 25]. ∎

This means the Fourier transform of every real-valued sequence is symmetric with respect to its middle. A simple implication of this lemma is $|X_{n-f}| = |X_f|$, i.e. every amplitude at the beginning except the first one appears at the end.

Observation 1 *In the class of (real-valued) time series that have an energy spectrum of the form $O(F^{-2b})$ for $b \geq 0.5$, the DFT coefficients are not only strong at the beginning but also strong at the end.*

This means if we do our distance computations based on only the first k Fourier coefficients, we will miss all the information carried by the last k Fourier coefficients which are as important as the former. However, the next observation shows that the first k Fourier coefficients are the only features that we need to store in the index.

Observation 2 *The first $\lceil (n+1)/2 \rceil$ DFT coefficients of every (real-valued) time series contain the whole information about the sequence.*

The point left to describe now is how we can take advantage of the last k Fourier coefficients without storing them in the index. We can write the Euclidean distance between two time series $\vec{x}$ and $\vec{q}$, using equations 1.3 and 1.2, as follows:

$$D^2(\vec{x}, \vec{q}) = D^2(\vec{X}, \vec{Q}) = \sum_{f=0}^{n-1} |X_f - Q_f|^2 \tag{1.4}$$

where $\vec{X}$ and $\vec{Q}$ are respectively DFTs of $\vec{x}$ and $\vec{q}$. Since $|X_{n-f}| = |X_f|$ and $|Q_{n-f}| = |Q_f|$ for $f = 1, \ldots, n-1$, we can write $D^2(\vec{X}, \vec{Q})$ as follows:

$$D^2(\vec{X}, \vec{Q}) = |X_0 - Q_0|^2 + \begin{cases} \sum_{f=1}^{n/2-1} 2|X_f - Q_f|^2 + |X_{n/2} - Q_{n/2}|^2 & even\ n \\ \\ \sum_{f=1}^{(n-1)/2} 2|X_f - Q_f|^2 & odd\ n \end{cases} \tag{1.5}$$

A necessary condition for the left side to be less than ϵ^2 is that every magnitude on the right side be less than ϵ^2. For the time being and just for the purpose of presentation, we assume time series are normalized [2] before being stored in the index. In general, time series may be normalized because of efficiency reasons [7] or other useful properties [11]. Since the first Fourier coefficient is zero for normalized sequences, there is no need to store it in the index. In addition, since k is usually a small number, much smaller than n, we can assume that the $(n/2)th$ coefficient is also not stored in the index. Now the condition left to be checked on the index is

$$2|X_f - Q_f|^2 < \epsilon^2 \quad for\ f = 1, \ldots, k$$

or, equivalently

$$|X_f - Q_f| < \frac{\epsilon}{\sqrt{2}} \quad for\ f = 1, \ldots, k$$

A common approach to check this condition is to build a search rectangle of side $\frac{2\epsilon}{\sqrt{2}} = \sqrt{2}\epsilon$ (or a circle of diameter $\sqrt{2}\epsilon$) around $\vec{Q}$ and check for an overlap between this rectangle (circle) and every rectangle in the index. The search rectangle is still guaranteed to include all points within the Euclidean distance ϵ from $\vec{Q}$, but there is a major drop in the number of false positives. The effect of reducing the size of the search rectangle on the search time of a range query is analytically discussed in the next section.

The symmetry property can be similarly used to reduce the size of the search rectangle even if sequences are not normalized. The only difference is that one side of the search rectangle (the one representing the first DFT coefficient) is 2ϵ and all other sides are $\sqrt{2}\epsilon$.

We can show that all-pair queries also benefit from the symmetry property of DFT. Suppose we want to answer an all-pair query using two R-tree indices, i.e., to find all pairs of sequences that are within distance ϵ form each other. A common approach for processing this query is to take pairs of (minimum bounding) rectangles, one rectangle from each index, extend the sides of one by 2ϵ and check for a possible overlap with the other. However, the symmetry property implies that if we extend every side by $\sqrt{2}\epsilon$, the result is still guaranteed to include all qualifying pairs though the number of false positives is reduced.

2.2 ANALYTICAL RESULTS ON THE SEARCH TIME IMPROVEMENTS

There are two factors that affect the search time of a range query, if we assume the CPU time to be negligible; one is the number of index nodes touched by the query rectangle and the other is the number of data points inside the search rectangle (or candidates). Both factors can be approximated by the area of the search rectangle, if we assume data points are uniformly distributed over a unit square, and the search rectangle is a rectangle within this square (we relax this assumption later in this section). Thus, to compare the search time of a rectangle of side $\sqrt{2}\epsilon$ to that of a one of side 2ϵ, we compare their areas.

Since a search rectangle has $2k$ sides, the area (or the volume) of a search rectangle of side $\sqrt{2}\epsilon$ is $(\sqrt{2}\epsilon)^{2k} = 2^k\epsilon^{2k}$. This is one 2^kth of the area (or the volume) of a rectangle of side 2ϵ which is $(2\epsilon)^{2k} = 2^{2k}\epsilon^{2k}$. Thus under the assumptions we have made, using a search rectangle of side $\sqrt{2}\epsilon$ instead of a one of side 2ϵ should reduce the search time by $(1-1/2^k)*100\%$. For example, using a rectangle of side $\sqrt{2}\epsilon$ on an index built on the first two non-zero DFT coefficients should reduce the search time by 75%.

However, for the class of time series that have an *energy spectrum* of the form $O(F^{-2b})$, the *amplitude spectrum* follows $O(F^{-b})$. In particular for $b > 0$, the amplitude reduces as a factor of frequency and points get denser in higher frequencies. If we assume that the first non-zero DFT coefficient (for every data or query sequence) is uniformly distributed within a unit square, the ith DFT coefficient (for $i = 1, \ldots, k$) must be distributed uniformly within a square of side i^{-b}. Thus keeping the first k Fourier coefficients maps sequences into points which are uniformly distributed within rectangle $R = (< 0, 1 >, < 0, 1 >, < 0, 2^{-b} >, < 0, 2^{-b} >, \ldots, < 0, k^{-b} >, < 0, k^{-b} >)$.

In addition, a search rectangle built on an arbitrarily chosen query point $\vec{Q}$ (inside or on R) is not necessarily contained fully within R. If $\vec{Q}$ happens to be a central point of R, as is shown in Figure 1.1, the overlap between the two rectangles reaches its maximum. We refer to this query as 'the worst case query' since it requires the largest number of disk accesses. On the other hand, if $\vec{Q}$ happens to be a corner point of R, the overlap between the two rectangles reaches its minimum. We call this query 'the best case query'. Thus the area of the overlap between the search rectangle and R, and as a result the search time, is not only a factor of ϵ but also a factor of $\vec{Q}$.

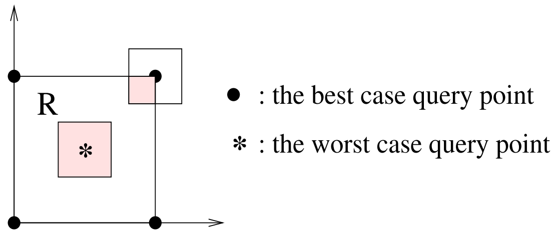

Figure 1.1 Two extreme query points

To compare the search time of a query rectangle of side $\sqrt{2}\epsilon$ to that of one of side 2ϵ, we can compare their area of overlap with R. The projection of the overlap between a search rectangle of side 2ϵ and R to the ith DFT coefficient plane is a square of side $min(i^{-b}, 2\epsilon)$ for the worst case query and a square of side $min(i^{-b}, \epsilon)$ for the best case query. Thus the area of the overlap between the search rectangle and R for the worst case query is $\prod_{i=1}^{k}(min(i^{-b}, 2\epsilon))^2$ and that for the best case query is $\prod_{i=1}^{k}(min(i^{-b}, \epsilon))^2$.

To eliminate the effect of the size of R in our estimates, we divide the area of the overlap by the area of R, i.e. $\prod_{i=1}^{k}(i^{-b})^2$, to get what we call the *query selectivity*. The query selectivity for the worst case query using a search rectangle of side 2ϵ can be expressed as follows:

$$S(b, k, 2\epsilon) = \frac{\prod_{i=1}^{k}(min(i^{-b}, 2\epsilon))^2}{\prod_{i=1}^{k}(i^{-b})^2} = \prod_{i=1}^{k}(min(i^{-b}, 2\epsilon)i^b)^2. \tag{1.6}$$

The term $min(i^{-b}, 2\epsilon)i^b$ is 1 for $i^{-b} \leq 2\epsilon$ (or $i \geq (2\epsilon)^{-1/b}$) , and it is $2\epsilon i^b$ for $i^{-b} > 2\epsilon$ (or $i < (2\epsilon)^{-1/b}$). Thus the query selectivity can be expressed as

$$S(b, k, 2\epsilon) = \prod_{i=1}^{min(k, \lfloor (2\epsilon)^{-1/b} \rfloor)} (2\epsilon i^b)^2 \tag{1.7}$$

It can be easily shown that $S(b, k, \epsilon)$ gives the query selectivity for the best case query using the same search rectangle. If we employ the symmetry property of the DFT, i.e. use a search rectangle of side $\sqrt{2}\epsilon$, the query selectivities for the worst and the best case queries respectively would be $S(b, k, \sqrt{2}\epsilon)$ and $S(b, k, \frac{\epsilon}{\sqrt{2}})$.

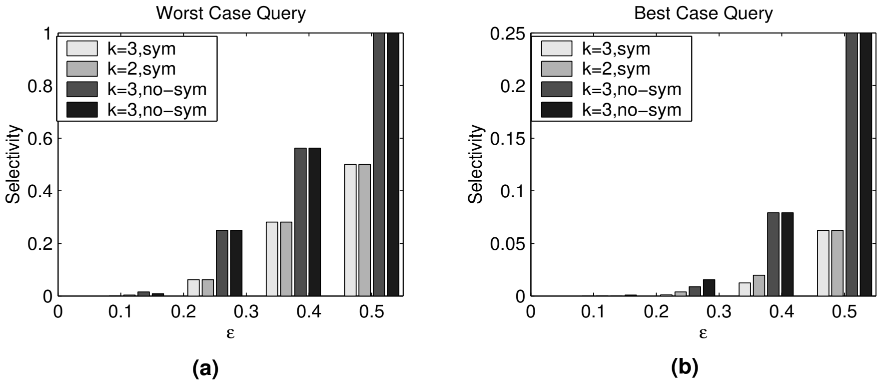

Figure 1.2 Query selectivity per search rectangle and k varying the threshold on brown noise data for: (a) the worst case query, (b) the best case query

Figure 1.2(a) shows the worst case query selectivity per search rectangle and k varying the query threshold for brown noise data ($b = 1$). As is shown, using the symmetry property reduces the query selectivity by 50% to 75% for $k = 2$ and $\epsilon \leq 0.5$. If we keep the first three non-zero DFT coefficients ($k = 3$), using the symmetry property reduces the selectivity by up to 87%. In general, taking the symmetry property into account reduces the selectivity and as a result the search time in the worst case by 50% to $(1 - 1/2^k) * 100\%$ for $k \geq 2$ and $\epsilon \leq 0.5$.

Figure 1.2(b) shows the best case query selectivity per search rectangle and k varying the query threshold again for the brown noise data. As is shown, taking the symmetry property into account reduces the selectivity by at least 75% for all values of $\epsilon \leq 0.5$, if we keep only the first two non-zero DFT coefficients. In general, taking the symmetry property into account reduces the selectivity and as a result the search time of the best case query by 75% to $(1 - 1/2^k) * 100\%$ for $k \geq 2$ and $\epsilon \leq 0.5$.

3. EXPERIMENTS

To show the performance gain of our proposed method, we implemented it using Norbert Beckmann's Version 2 implementation of the R*-tree[4] and compared it to the original indexing method proposed by Agrawal et al. [1].

All our experiments were conducted on a 168MHZ Ultrasparc station. We ran experiments on the following two data sets:

1. Real stock prices data obtained from the FTP site "ftp.ai.mit.edu/pub/stocks/results". The data set consisted of 1067 stocks and their daily closing prices. Every stock had at least 128 days of price recordings.

2. Synthetic random walk sequences each of the form $\vec{x} = [x_t]$ where $x_t = x_{t-1} + z_t$ and z_t is a uniformly distributed random number in the range $[-500, 500]$. The data set consisted of 20,000 sequences.

We first transformed every sequence to its normal form, and then found its DFT coefficients. We kept the first k DFT coefficients as the sequence features. Since a DFT coefficient was a complex number, a sequence became a point in a $2k$-dimensional space. But the first DFT coefficient was always zero for normalized sequences, and we did not need to store it in the index; instead, we stored the mean and the standard deviation of a sequence along with its $k - 1$ DFT coefficients. In our experiments we used the polar representation for complex numbers.

To do the performance comparison, we used both range and all-pair queries. For range queries, we ran each experiment 100 times and each time we chose a random query sequence from the data set and searched for all other sequences within distance ϵ of the query sequence. We averaged the execution times from these runnings. Our all-pair queries were spatial self-join queries where we searched the data set for all sequence pairs within distance ϵ of each other.

3.1 VARYING THE QUERY THRESHOLD

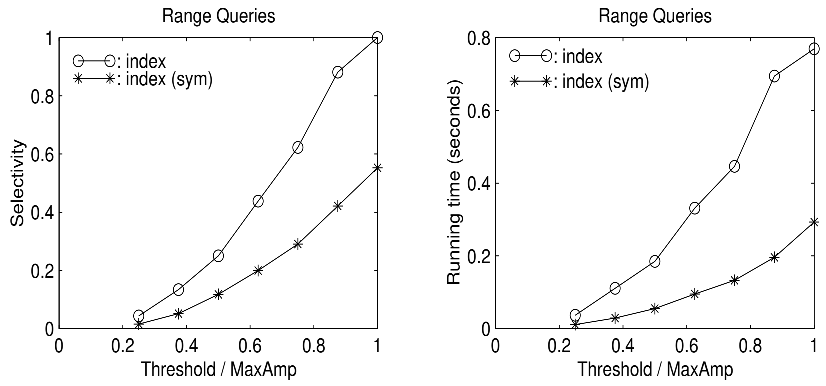

Figure 1.3 Both query selectivities and running times for range queries varying the query threshold

Our first experiment was on stock prices consisting of 1067 time series each of length 128. Our aim was to make a comparison between average case query selectivities obtained experimentally and the extreme case query selectivities computed analytically. We fixed the number of DFT coefficients to 2, but we varied the query threshold from $1 * MaxAmp$ to $0.24 * MaxAmp$ where $MaxAmp$ was the maximum amplitude of the first non-zero DFT coefficient over all sequences in the data set. Under this setting, a threshold $\epsilon * MaxAmp$ in our experiments was analogous to threshold ϵ in our analytical results. The average output size for $\epsilon = 1 * MaxAmp$ was 75 out of 1068 and that for $\epsilon = 0.24 * MaxAmp$ was zero, so we didn't try smaller thresholds. Since query points were chosen randomly, we expected the query selectivity for every threshold $\epsilon * MaxAmp$ to fall between the two extreme selectivities (the worst case and the best case) computed analytically for ϵ. As is shown in Figure 1.3, for $\epsilon/MaxAmp \leq 0.5$, using the symmetry property reduces the query selectivity by 53% to 64% and the search time by 70% to 74%. It is consistent with our analytical results. For $0.5 < \epsilon/MaxAmp \leq 1$, as the figure shows, using the symmetry property reduces the query selectivity by 45% to 64% and the running time by 62% to 74%.

3.2 VARYING THE NUMBER OF DFT COEFFICIENTS

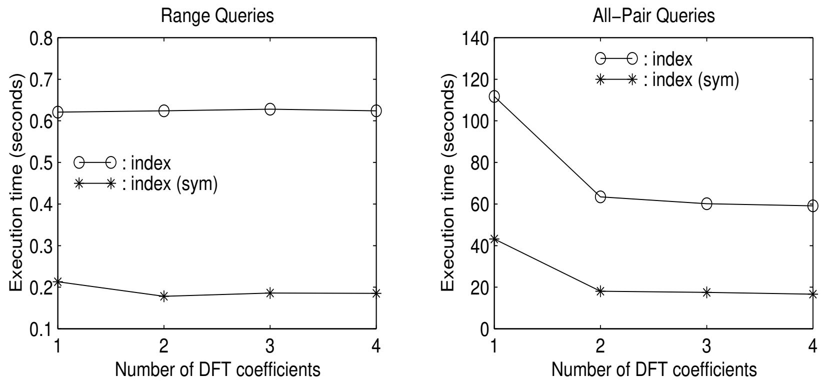

Figure 1.4 Running times for range and all-pair queries varying the number of DFT coefficients

Our next experiment was again on stock prices data, but this time we fixed the query threshold for range queries to $0.95 * MaxAmp$ and that for all-pair queries to $0.32 * MaxAmp$. This setting gave us average output sizes of 30 and 203 respectively for range and all-pair queries. We varied the number of DFT coefficients kept in the index from 1 to 4. Figure 1.4 shows the running times per query for range and all-pair queries. Taking our observations into account reduces the search time of the index by 66% to 72% for range queries and by 61% to 72% for all-pair queries.

3.3 VARYING THE NUMBER OF SEQUENCES

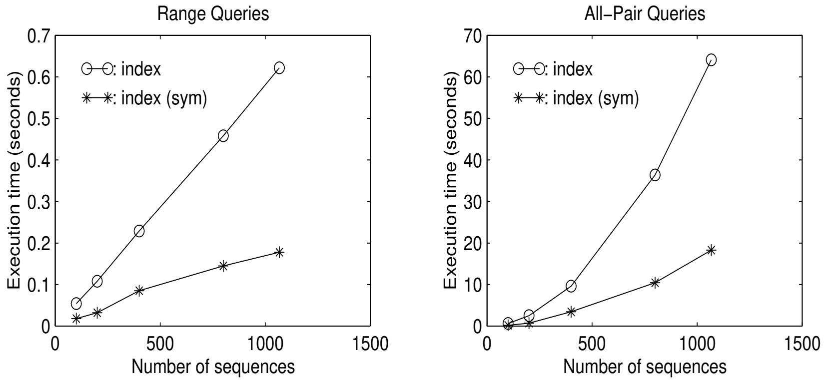

Figure 1.5 Running times for range and all-pair queries varying the number of sequences

In our next experiment, we fixed the number of DFT coefficient to 2 and the sequence length to 128, but we varied the number of sequences from 100 to 1067. The experiment conducted on stock prices data set. We again fixed the query threshold for range queries to $0.95 * MaxAmp$ and that for all-pair queries to $0.32 * MaxAmp$. Figure 1.5 shows the running times per query for range and all-pair queries. Our observation reduces the search time of the index by 63% to 71% for range queries and by 64% to 72% for all-pair queries.

3.4 VARYING THE LENGTH OF SEQUENCES

Our last experiment was on synthetic data where we fixed the number of DFT coefficients to 2 and the number of sequences to 20,000, but we varied the sequence length from 128 to 512. The size of the data file was in the range of 40 Mbytes (for sequences of length 128) to 160 Mbytes (for sequences of length 512). We fixed the query threshold to $0.44 * MaxAmp$ and, based on our analytical results, we expected using the symmetry property to reduce the search time by 50% to 75%. Figure 1.6 shows the running times per query for range queries. Our proposed method reduces the search time of the index by 73% to 77%. The search time improvement is slightly more than our analytical estimates mainly because of the CPU time reduction for distance computations which is not accounted for in our analytical estimates. Because of the high volume of data, experiments on all-pair queries were very time consuming. For example, doing a self-join on sequences of length 512 did not finish after 12 hours of overnight running. For this reason, we did not report them.

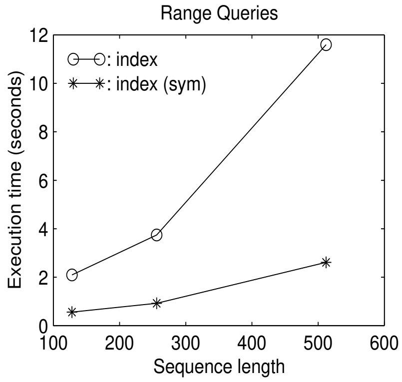

Figure 1.6 Running times for range queries varying the length of sequences

4. CONCLUSIONS

We have proposed using the last few Fourier coefficients of time sequences in the distance computation, the main observation being that every coefficient at the end is the complex conjugate of a coefficient at the beginning and as strong as its counterpart. Our analytical observation shows that using the last few Fourier coefficients in the distance computation accelerates the search time of the index by more than a factor of two for a large range of thresholds. We also evaluated our proposed method over real and synthetic data. Our experimental results were consistent with our analytical observation; in all our experiments the proposed method reduced the search time of the index by 61% to 77% for both range and all-pair queries.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario.

Notes

1. $(a + bj)^* = (a - bj)$
2. A sequence is in normal form if its mean is 0 and its standard deviation is 1.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proceedings of the FODO Conference*, pages 69–84, Chicago, October 1993.

[2] R. Agrawal, K. 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 VLDB Conference*, pages 490–501, Zurich, September 1995.

[3] R. Agrawal, G. Psaila, E.L. Wimmers, and M. Zait. Querying shapes of histories. In *Proceedings of the VLDB Conference*, pages 502–514, Zurich, September 1995.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R* tree: an efficient and robust index method for points and rectangles. In *Proceedings of the ACM SIGMOD Conference*, pages 322–331, Atlantic City, May 1990.

[5] C. Chatfield. *The Analysis of Time Series: an Introduction*. Chapman and Hall, fourth edition, 1984.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the ACM SIGMOD Conference*, pages 419–429, Minneapolis, May 1994.

[7] D.Q. Goldin and P.C. Kanellakis. On similarity queries for time-series data: constraint specification and implementation. In *1st Intl. Conf. on the Principles and Practice of Constraint Programming*, pages 137–153. LNCS 976, Sept. 1995.

[8] H.V. Jagadish, A.O. Mendelzon, and T. Milo. Similarity-based queries. In *Proceedings of the ACM PODS Conference*, pages 36–45, San Jose, May 1995.

[9] B. Mandelbrot. *Fractal Geometry of Nature*. W.H. Freeman, 1983.

[10] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1975.

[11] D. Rafiei. On similarity-based queries for time series data. In *Proceedings of the ICDE Conference*, Sydney, March 1999.

[12] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proceedings of the ACM SIGMOD Conference*, pages 13–24, Tucson, May 1997.

[13] M. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W.H. Freeman, 1991.

[14] B.J. West and M. Shlesinger. The noise in natural phenomena. *American Scientist*, 78:40–45, Jan-Feb 1990.

[15] B. Yi, H.V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the ICDE Conference*, pages 201–208, Orlando, February 1998.