TRCS00-08

A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases

Yi-Leh Wu Divyakant Agrawal Amr El Abbadi

Department of Computer Science
University of California, Santa Barbara
Email:{ywu, agrawal, amr}@cs.ucsb.edu

Abstract

Similarity search in time-series databases has received significant attention lately. Popular techniques for efficient retrieval of time sequences in time-series databases has been to use Discrete Fourier Transform (DFT). Recently, the Discrete Wavelet Transform (DWT) has gained popular interest in database domain and several proposals have been made to replace DFT by DWT for similarity search over time-series databases. In this paper, we explore the feasibility of replacing DFT by DWT with a comprehensive analysis of the DFT and DWT as matching functions in time-series databases. Our results show that although the DWT based technique has several advantages, e.g., the DWT has complexity of $O(N)$ whereas DFT is $O(N \log N)$, DWT does not reduce relative matching error and does not increase query precision in similarity search as suggested by previous works [1]. We conclude that, by exploring the conjugate property of DFT in real domain, the DFT-based and DWT-based techniques yield comparable results on similarity search in time-series databases.

Key words. time-series analysis, fourier transform, wavelet transform, time-series database, smoothing, time-series matching

1 Introduction

Time-series data constitute a large portion of the data stored in computers. A time-series is a sequence of real numbers, each number represents a data value at a point in time. Examples of time-series data include stock prices, weather data, exchange rates, history of product sales, medical information, etc. Many applications with temporal data require the capability of searching, especially based on similarity, over the data. For example, we may want to find the stocks that have correlation with Microsoft stock; or the time-period during which two commodities have similar price patterns.

There have been several efforts to develop efficient similarity search mechanisms in time-series databases. In [2], Discrete Fourier Transform (DFT) was employed to map time-series data from the time domain to the frequency domain. After dropping all but the first few frequency coefficients, the remaining ones are indexed through a multidimensional index structure such as R^*-Tree. But in [2], one of the main constraints of this approach is that is it assumed that the data sequence and the query sequence have the same length. This problem was tackled in [3] which allowed subsequence matching by using a sliding window over the data sequence, map each window to the frequency domain using DFT and keep only the first few coefficients. A data sequence is thus mapped into a trail in the feature space and the trail is further divided into sub-trails that can be represented by their minimum bounding rectangles (MBR) and stored in a R^*-Tree for indexing. Rafiei and Mendelzon [4] extend previous works to handle time scaling, i.e., stretching or shrinking the time axis. In [5], they proposed an improvement of DFT-based indexing techniques for time-series data by using the last few Fourier coefficients in the distance computation to speed up similarity search without storing them.

All the above approaches assume Euclidean distance as the measurement of similarity. Agrawal et al. [6] propose a new distance measure to capture the notions that two sequences should be considered similar if they have enough non-overlapping ordered similar subsequences. In [7], Yi et al. introduce "time warping" distance as the similarity measurement and techniques to speed up the similarity query processing. Park et al. [8] use time warping distance with a disk-based suffix tree indexing method for retrieval similar subsequences without false dismissals. Perng et al. [9] propose the *Landmark Similarity* measurement that is invariant under six transformations (e.g., shifting, uniform amplitude scaling, etc.), where the landmarks are filtered local maxima and minima in the time sequences.

Most of the previous works employ DFT to map time-series data from the time domain to the frequency domain. By taking only the first few Fourier coefficients for indexing they effectively reduce the search space and speed-up the similarity query. Chan and Fu [1] first proposed to use the Discrete Wavelet Transform (DWT) to replace the well-accepted DFT for various reasons; e.g., the computation of DWT is more efficient than DFT in general. They state that the DWT as a matching function has more discrimination power than DFT.

In this paper, we present a comprehensive comparison between DFT and DWT as matching functions in time-series databases. Our experiment results show that there is marginal difference for discrimination power with DWT or DFT if we consider the conjugate property of DFT as proposed in [5].

The rest of the paper is organized as follows. In Section 2, we gives an overview of the Discrete Fourier Transform (DFT) and Discrete Wavelet Transform (DWT). In Section 3, we exam the hypothesis proposed by Chan and Fu [1], and analyze DFT and DWT based matching for time-series datasets. In Section 4, we compare the relative matching error and query precision by using the DFT and DWT in real-world time-series database. We conclude in Section 5.

2 Background

In this section, we start by giving a brief introduction of how to process similarity search in time-series database followed by a brief description of DFT and DWT.

2.1 Query Processing in Time-series Databases

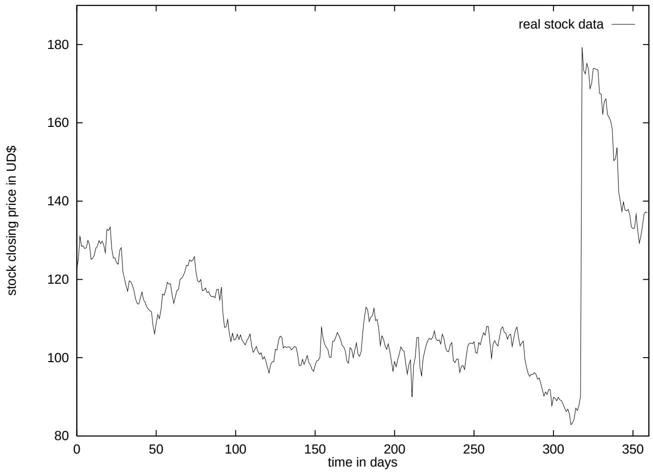

Figure 1: Real Stock Data for 360 days

A time-series chart of the stock closing prices of IBM for 360 consecutive transaction days is shown in Figure 1. A time-series database consists of thousands of such time-series sequences. A common operation/query against such a database would be: Given a query sequence, e.g., Microsoft stock closing prices for the past 3 months, find the stocks in the database that have a close correlation with its behavior. Although the data-sequences used in this paper are from financial applications, such data-sequences arise in a variety of applications. For example, observations of physical phenomena (temperature, rainfall, etc.), for scientific applications over a period of time, seismic activity in southern California, EKG observations/samples of patients in the last few years. etc. In the above applications, correlation or similarity based queries play an important role. More formally, a time-series database is a set denoted $DB = \{X_1, X_2, ..., X_i, ..., X_N\}$, where $X_i = [x_0^i, x_1^i, ..., x_n^i]$ and a query sequence is a sequence of data points $Q = [q_0, q_1, ..., q_n]$. Given a query Q, the result set R from the database is $R = \{X_1, X_2, ..., X_j, ..., X_m\}$, such that $D(X_j, Q) < d$. The distance function $D(X, Y)$ is the Euclidean distance between X and Y, that is

$$D(X,Y) = (\sum_j |x_j - y_j|^2)^{1/2} \tag{1}$$

which is the aggregation of the point to point distance of two sequences.

One way to solve this problem is as follows. Given a query sequence Q, we compare all sequences stored in the database with Q using the distance function in Equation 1 and put all sequences within distance d into the result set R. Although this approach is correct, it is not practical for two reasons. First, the number of sequences in the database may be large and a sequential scan of all such sequence for every query will result in severe performance penalty. Second, the number of data points in a query may need to be matched against each sequence not once but N (the database size) times so that all possible subsequences of that sequence are evaluated.

Another approach is to use mathematical transforms to capture the essence of time-series sequences. One such transform is the discrete Fourier transform (DFT). The DFT takes the original signals in time/space domain and transforms them into the frequency domain. The significance of the DFT is that there exists a fast algorithm that can compute the DFT coefficients in $O(n \log n)$ time.

Many previous proposals [2, 4, 5, 3] in time-series databases use the notion of DFT and separate the query processing phase from the indexing phase. The indexing phase will usually take the following steps,

1. Take the original sequence data and chop it into fixed sized subsequence samples using a sliding window.
2. Normalize all the subsequence samples so that all the sample data fall into a certain range.
3. Use the DFT to transform the subsequence samples into the frequency domain.
4. Use only the first few DFT coefficients to represent the original subsequence.
5. Use any multidimensional index structure such as R-trees to index the resulting few coefficients.

By using an appropriate transformation we can capture the approximate shape of a given long sequence and hence use fewer data points to describe the overall shape. Note that, the underlying shape of a sequence/signal, which is the slow changing part, is the low frequency part of the sequence/signal. Hence, in step 4, by keeping only the first few DFT coefficients, the approximate shape of the original subsequence can be captured.

In the querying phase, given a query sequence Q with the same number of data samples as all the subsequences in the database, find the most similar ones to Q within a distance threshold,

1. Normalize and DFT Q as in indexing steps 2 and 3.
2. Take the same number of coefficients as in indexing step 4 from the resulting DFT coefficients or Q'.
3. Search the index structure(e.g., R-tree) with the remaining coefficients. Find all sequences in the indexing structure that are within distance d to Q'. The resulting R' contains all sequences with estimated distance within d to Q'.
4. For each sequence in R', if the true distance to Q is within d, put it in the final result set R.

Since we only stored the approximate shapes of the original sample sequences in the database, the distance calculation in step 3 cannot be the exact distance between query sequence Q and the original sample sequences in time domain. Parseval's Theorem [2] guarantees that the distance between Q' and the approximated sample sequence X' is always smaller than the distance between Q and the original sample sequence X. This result implies that the result set R' in step 3 is always a superset of the actual result set R in step 4. In another word, by keeping only a few DFT coefficients, i.e., keeping the basic shape and dropping the detail information, we successfully reduce the cost of indexing and distance calculations. But at the same time, we introduce false hits in step 3 of the querying phase as a result of losing detailed information in the indexing phase. This is a design tradeoff between having fast query processing in step 3 at the expense of false hits in step 4 versus slow query processing in step 3 but with less false hits for step 4. This can be adjusted by varying the number of coefficients that are chosen in step 4 of the indexing phase.

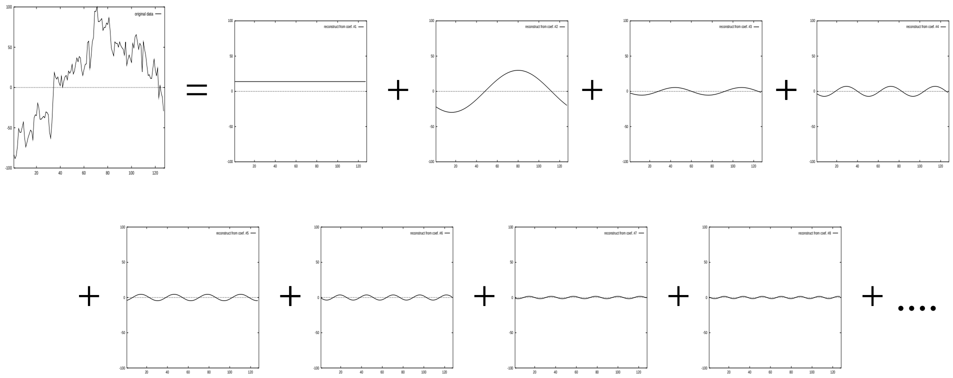

Figure 2: Discrete Fourier Transform (DFT)

Figure 2 shows how DFT decomposes a signal into different frequency parts. Each frequency response is represented by one coefficient of DFT. The signals on the right hand side of the equation represent portions of the original signal at specific frequencies.

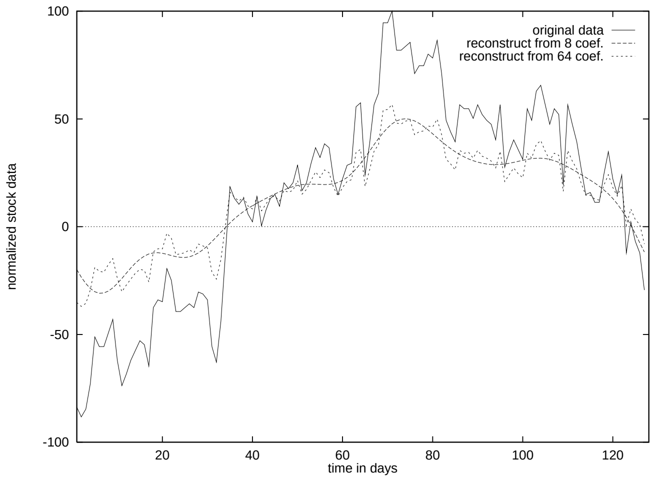

Figure 3: Reversed Discrete Fourier Transform (RDFT)

Figure 3 shows the effects of reconstructing the original sequence with only the low frequency parts of the original signal using RDFT. The solid line in Figure 3 shows the original signal. The dashed line in Figure 3 shows the reconstruction using only the first 8 DFT coefficients Note that the reconstructed signal captures the basic shape of the original sequence. Database researchers have proposed to explore this property for efficient processing of similarity based queries [2, 4, 3]. The dotted line in Figure 3 shows the reconstruction of using the first 64 DFT coefficients, which is half of the total coefficients. The details in the original sequence gradually appear as the number of coefficients increased.

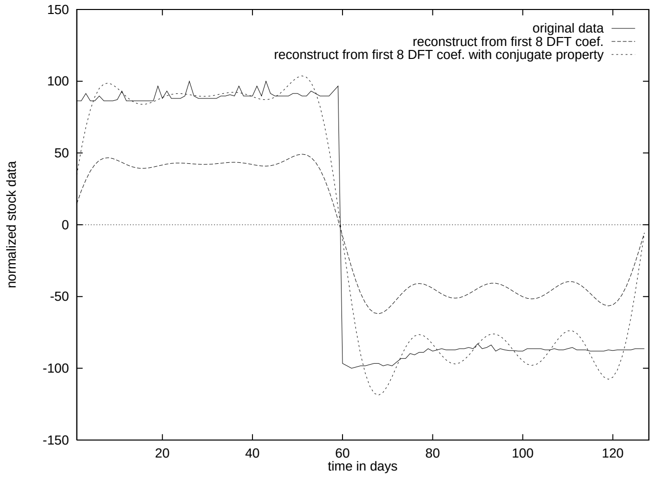

Figure 4: The conjugate property effect of DFT

An interesting property of the DFT is that if the input sequence is in the real domain, i.e. all signal samples are real numbers, then the resulting DFT coefficients have a mirror effect such that the coefficients in the rear are complex conjugates of the coefficients in the front. Rafiei and Mendelzon [5] suggested to use this property to increase the precision of distance calculations without increasing the number of coefficients stored for indexing. The idea is that although only the first few coefficients are used for indexing, say l coefficients are stored, due to the complex conjugate property we can have $2l$ coefficients for the distance calculation. This technique greatly reduces the error by the thresholding process. Figure 4 illustrates the differences of signal reconstruction using this approach. The significance of this technique is that it can greatly reduce the distance measuring errors and by doing so reduces the false hits in the result.

2.2 Discrete Wavelet Transform

We now introduce another family of transformations, the Discrete Wavelet Transform (DWT) [10], that performs similar properties as DFT in time-series signals. Whereas the basis function of the DFT is a sinusoid, the wavelet basis is a set of functions which are defined by a recursive function

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{2}$$

where 2^j is the scaling of t (j is the log_2 of the scale), $2^{-j}k$ is the translation in t, and $2^{j/2}$ maintains the L^2 (the space of square integrable functions) norm of the wavelet at different scales. So any signal in $L^2(R)$ can be represented by the series

$$f(t) = \sum_{j,k} a_{j,k} 2^{j/2}\psi(2^j t - k) \tag{3}$$

or, using Equation 2, as

$$f(t) = \sum_{j,k} a_{j,k} \psi_{j,k}(t) \tag{4}$$

where the two-dimensional set of coefficients $a_{j,k}$ is called the Discrete Wavelet Transform (DWT) of $f(t)$. A more specific form indicating how the $a_{i,j}$'s are calculated can be written using inner products as

$$f(t) = \sum_{j,k} \langle \psi_{j,k}(t), f(t) \rangle \psi_{j,k}(t) \tag{5}$$

if the $\psi_{j,k}(t)$ form an orthonormal basis for the space of signals of interest [10].

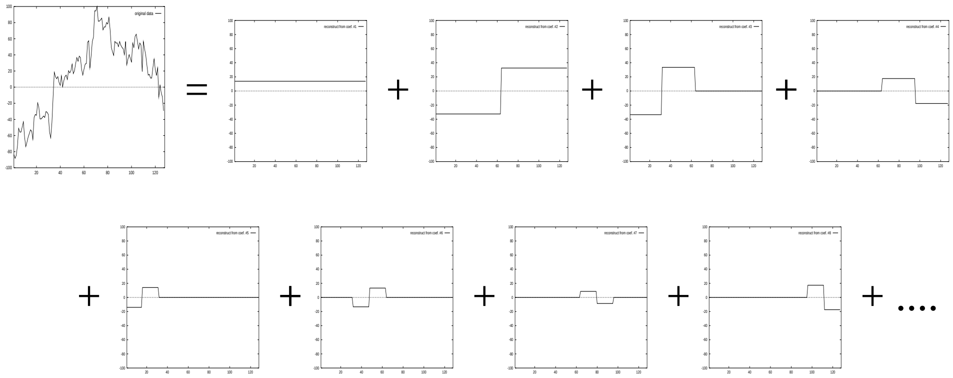

Figure 5: Discrete Wavelet Transform (using Haar wavelet)

Unlike the DFT that takes the original signals in time/space domain and transforms them into frequency domain, the Wavelet transform takes the original signals in time/space domain and transforms them into time/frequency or space/frequency domain. In Figure 5, we show how DWT decomposed the original signal into different frequency components, which is similar to DFT. Several things have to be noted. First, the wavelet we used is not sinusoid. We use a special class of wavelets called Haar wavelet [10] throughout this paper because of its simplicity. One can use any other orthonormal basis wavelets and achieve similar results as we present here. Second, the decomposed signals differ not only in the frequency but also in the position/time of the signal responses. Each time/frequency response is represented by one coefficient of DWT. The signals on the right hand side of the equation represent portions of the original signal at specific time/frequencies. The frequencies they represent increase from left to right and top to bottom. And the time responses spreads through the entire time domain at different frequency levels. Since the wavelet transform gives a time-frequency *localization* of the signal, it means most of the energy of the signal can be represented by only a few DWT coefficients.

In Figure 6, we show the effects of reconstructing the original sequence with only some of the time/frequency parts of the original signal using reversed discrete wavelet transform. The solid line in Figure 6 shows the original signal. The dashed line shows the reconstruction of using only the first 8 DWT coefficients, i.e., the first 8 time/frequency parts of the original sequence. And the dotted line shows the reconstruction using the first 64 DWT coefficients. Note how the basic shapes of the original sequence are captured by using

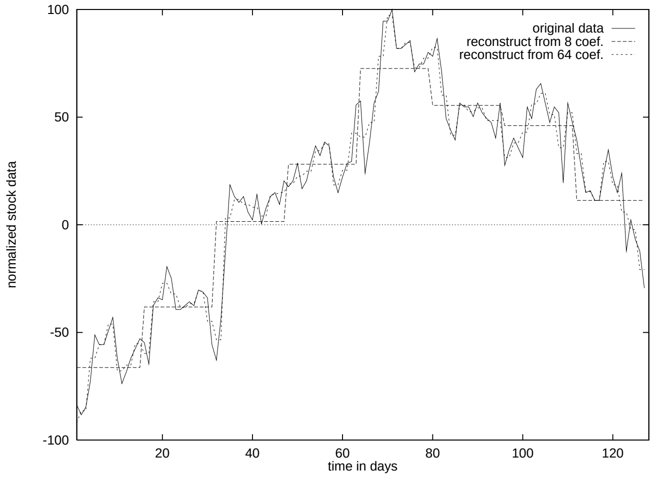

Figure 6: Reversed Discrete Wavelet Transform

DWT. Comparing the results by using DFT in Figure 3 and by using DWT in Figure 6, we note that DWT appears to be superior DFT in capturing the underlying shape of the original signal.

The difference between DFT and DWT is that DFT maps a one dimensional time domain discrete function into a representation in frequency domain while the wavelet transform maps it into a representation that allows localization in both time and frequency domains. This special property of DWT supports the hypothesis proposed in [1] that the DWT is more suitable than the DFT in time series database applications because it reduce the error of distance estimates on the transformed domains. In the next section, we verify this hypothesis by a comprehensive analysis on real data sets.

3 Why wavelets?

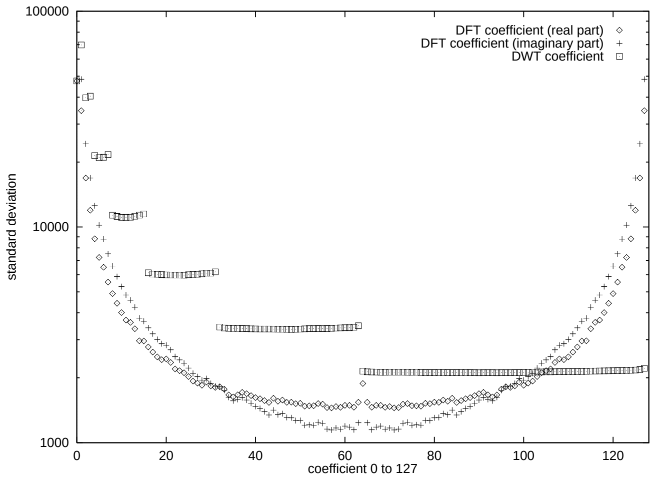

Figure 7: Standard deviation of individual DFT/DWT coefficients

In this section we verify the hypothesis that DWT is superior than DFT in time-series database domain as described in Section 2. As discussed in Section 2.1, the main purpose of using any type of transformation on the original time sequences is to extract only a few representative transformed coefficients for identifying and indexing the original time sequences. That is, we try to select the coefficients that can most differentiate the transformed sequence from others; in most cases, the coefficients that reserve the most energy of the original sequence.

The data set we used throughout this section contains 36000 stock price sequences as described in Section 4. Each sequence consists of the stock closing price of one company for 128 days. We applied the DFT and DWT on all 36000 sequences to construct the coefficient dataset for analysis. In Figure 7, we show how the individual DFT and DWT coefficients are actually distributed. Each point in Figure 7 represents the standard deviation of individual transformed coefficients of the 36000 sample sequences in the test data set. For example, Figure 7 shows that the 20th DWT coefficient in the dataset has a standard deviation of 6012. Note that the DFT coefficients are complex numbers. Hence there are two separate distributions for the real and imaginary parts of each coefficients, as shown in Figure 7. In Figure 7, the first few DFT coefficients have higher standard deviations; this implies that by incorporating the first few coefficients we can differentiate the sequences from each others better. In other words, the sample sequences in database tend to have more differences in the first few coefficients than the rest. This supports the general intuition of keeping only the first few coefficients for similarity comparisons because they have more *differentiate* power than the rest. The distributions of the DWT coefficients have the same property as the DFT coefficients. Note that the distributions of the DFT coefficients have the symmetric spread as discussed in Section 2.1, the coefficients in the front are complex conjugates of the coefficients in the rear, as shown in Figure 7. Another observation is that the standard deviations of DWT coefficients decrease in a hierarchical manner as a result of the multiresolution property of the DWT. This special multiresolution property suggests that hierarchical indexing structures are possible for the DWT coefficients.

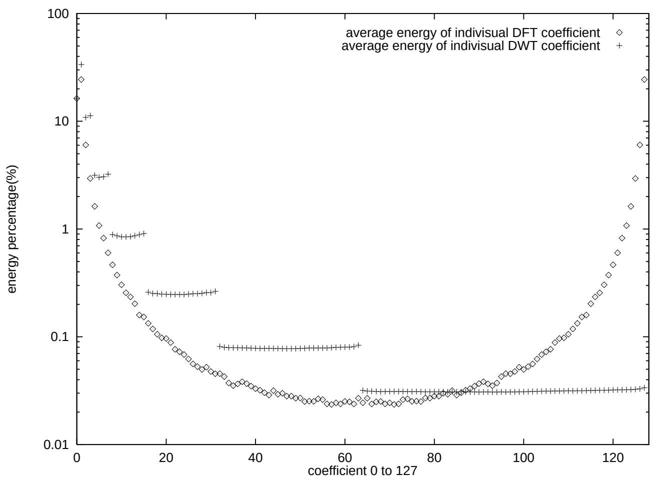

Figure 8: Averaged energy contained in each individual DFT/DWT coefficients

Another metric to measure the performance of transformation/thresholding processes is to evaluate how the transformed and thresholded sequences resemble the original sequences. In other words, we measure how much energy of the original sequence is preserved after the transformation/thresholding processes. The energy of the original sequence and the transformed sequence can be computed as described in Appendix A. Note that the Parseval's Theorem holds for both the DFT and DWT because the Haar wavelet we used in this paper is also orthonormal.

In the thresholding process, we keep the coefficients with most energy to reduce the transformation/thresholding error. Figure 8 shows the energy of the original sequence spread on individual DFT/DWT coefficient on average. For example, Figure 8 shows that the first DFT/DWT coefficients contain an average of 16% energy of the original sequence. Several observations can be made from Figure 8. First, the first DFT/DWT coefficients contains the same energy is not a coincident because they both represent the mean values of the original sequences. Second, the symmetric energy spread of the DFT coefficients suggest that most energy is preserved in the low-frequency and high-frequency coefficients but not in mid-frequency coefficients. Third, the hierarchical energy spread of the DWT coefficients suggest that most of the energy is reserved in the low resolution DWT coefficients. The above observations again suggest that in the thresholding step, keeping the first few coefficients can minimize transformation/thresholding error.

In some cases, selecting the first few coefficients does not guaranteed minimum error for thresholding. It is known that by keeping the largest m coefficients (in absolute value) is optimal in minimizing the absolute error for thresholding, assuming that the transformation is orthonormal [10]. The proof is in Appendix B. But keeping the largest coefficients need additional indexing space and indexing structures other than R-tree. Furthermore, the distance computation between two coefficients set, where the coefficients do not align and have different frequency contents, is expensive.

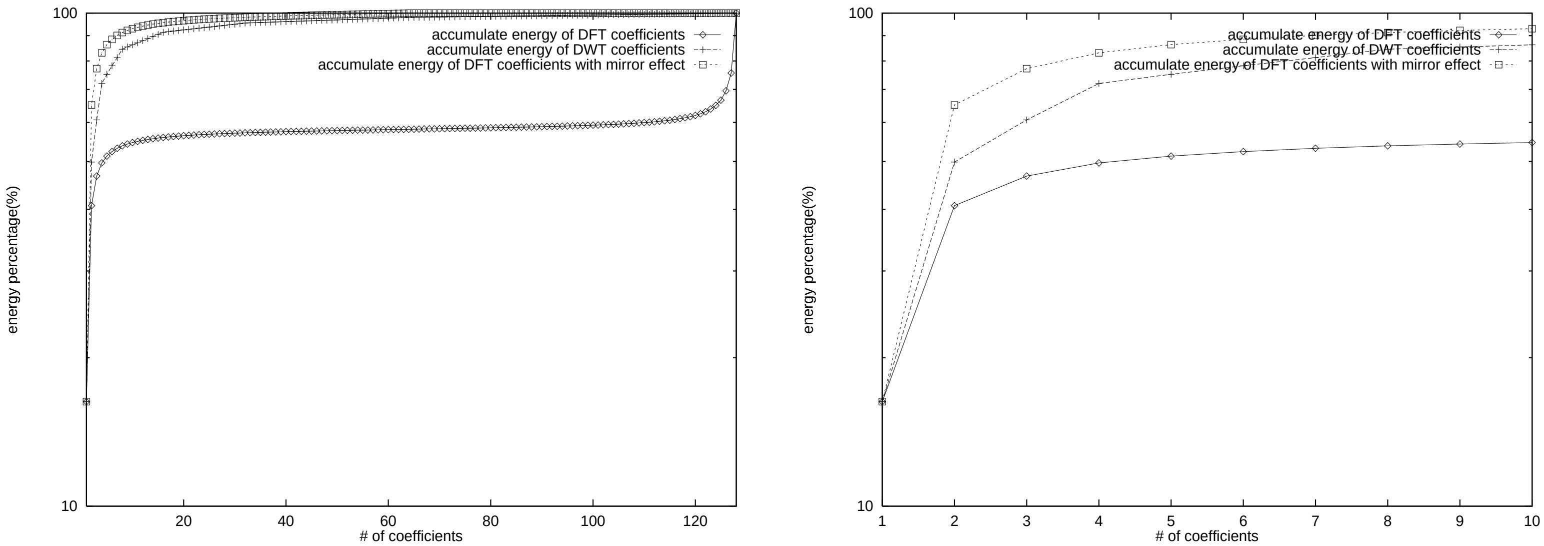

Figure 9: Accumulated energy of DFT/DWT coefficients

Next, we want to verify the hypothesis that the DWT has less transformation/thresholding error in time-series application than the DFT. Figure 9(a) shows the preserved energy by keeping varying numbers of DFT/DWT coefficients in the thresholding process. It shows that by keeping the first 20 DFT coefficient in the thresholding process we can preserved an average 56% energy of the original sequence; while keeping 20 DWT coefficients we can preserved 92% of the energy on average. It clearly shows that DWT is superior than the DFT in preserving energy. But when we consider the conjugate property of DFT explored by Rafiei and Mendelzon [5] as described in Section 2.1, Figure 9(a) shows that the first 20 DFT coefficients can recover an average 97% energy of the original sequence. This result suggest that the hypothesis by Chan and Fu [1], that by replacing DFT with DFT can reduce time series matching error, does not hold if we consider the conjugate property of DFT. In practice, we can only keep a very small number of coefficients, usually less than six, because by keep one more coefficient we increase the dimension of the indexing structure by one; and the R-tree family is known to degenerate as the dimension increase. Figure 9(b) is a zoom-in version of Figure 9(a). Figure 9(b) shows that even in very low dimension, by keeping very few coefficients, the DWT can not preserve more energy than the DFT if we consider the conjugate property of DFT.

4 Experiments

In this section, we conduct extensive experiments to evaluate the retrieval efficiency of DFT and DWT. The original time-series sequences are taken from real stock closing prices for a time span of 360 transaction days. There are more than nine thousands company stocks in the market today. We take the first 100 stocks in the order of ascending tick symbols from AA to $ADBE$. For each stock closing price sequence we use a 128 days window and slide it through the 360 days sequences from the beginning and take a 128 days long sample beginning at each data point. When the 128 days window reaches the end of the 360 days original sequence we simply warp the beginning of the 360 days sequence to the end. After the sampling process we have 360 subsequence samples of the original data sequences with each sample being 128 days long. Since we have 100 different stocks in our experiment, we end up with 36000 samples that are 128 days long.

4.1 Approximation Errors after Fourier and Wavelet Transforms

The matching processes relied on first applying the DFT/DWT on the data sequence, taking only a few coefficients as the basis for matching and discarding all the other coefficients. By reducing the number of transformed coefficients, the original data sequence cannot be obtained by the reversed transform and thus we introduce errors. One can measure the performance of different transformations by measuring the errors after transformation and thresholding. In the first experiment, we take each of the 36000 data samples in the database, perform the transformations, retain only a few coefficients, apply the reverse transform and compare it to the original data. We measure the error by the Euclidean distance between the transformed sequence and the original data sequence. The relative error is defined as,

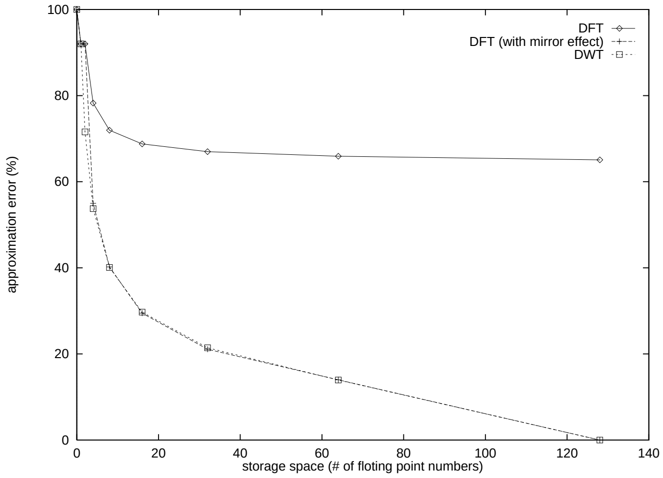

Figure 10: Relative errors with different transformations

$$E_{rel} = \frac{D(\vec{x}, \hat{x})}{D(\vec{x}, \vec{0})} \tag{6}$$

Relative error 0% means that after the transformation and thresholding process, the original data can be recovered completely; i.e., the energy is 100% preserved.

Note that, unlike the DWT coefficients that are real number,the DFT coefficients are complex numbers thus require twice the storage space. With the same storage space and indexing scheme, we can keep twice as many coefficients if we use DWT instead of DFT. Throughout this section, all the experiments assume that keeping a DFT coefficient takes two floating numbers while keeping a DWT coefficient takes only one floating number.

Figure 10 shows the relative errors of thresholding after DFT or DWT. It shows that keeping more DFT coefficients does not reduce the relative errors accordingly because the rest of the coefficients do not carry much information and the energy they carried approaches zero. However, if we consider the conjugate effect as described in Section 2.1, the relative error reduces much faster because the coefficients the rear end are complex conjugates of the front end and thus contains the same energy. Figure 10 also shows that, when considering the the conjugate property of the DFT, applying the DWT or DFT has marginal impact on relative errors.

4.2 Matching Errors after Fourier and Wavelet Transforms

In the next experiment we choose one sequence from the 36000 sample data database as the query sequence and measure the relative error when comparing the query sequence to each sample data sequence. The relative error between the query sequence and each sample data is defined as

$$E_{rel}(\vec{Q}, \vec{X}) = \frac{D(\hat{Q}, \hat{X})}{D(\vec{Q}, \vec{X})} \tag{7}$$

This metric tells us how good the transformation/thresholding process can estimate the true distance between two sequences. Then the average relative error of a query sequence and all sample data sequences is

$$E_{avg}(Q) = \frac{\sum_{i=1}^{n} E_{rel}(Q, X_i)}{n} \tag{8}$$

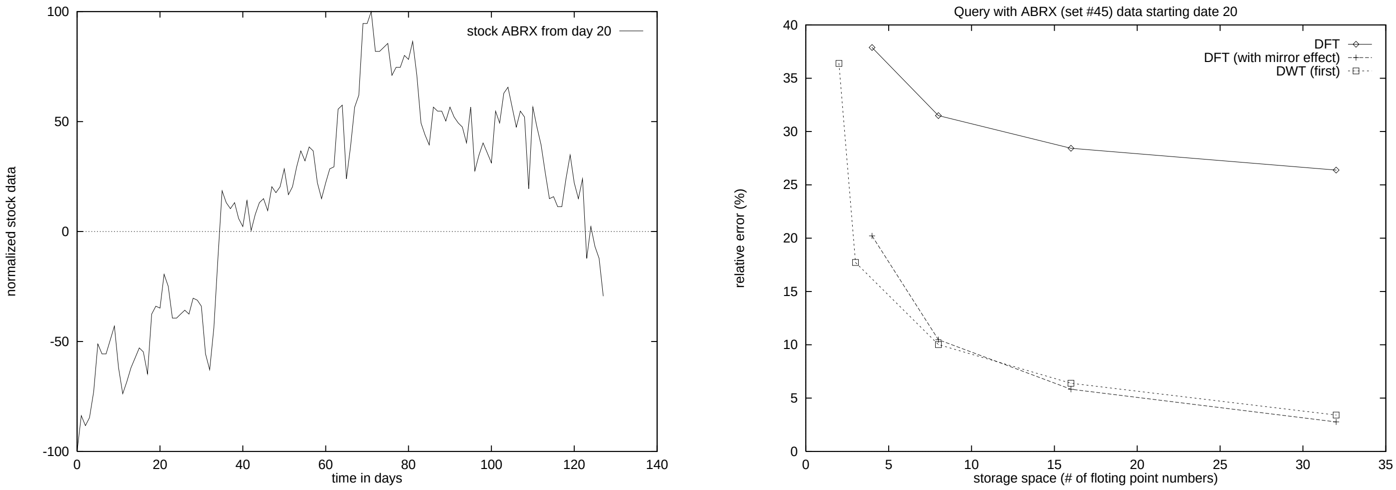

Figure 11: Query sequence ABRX and average error

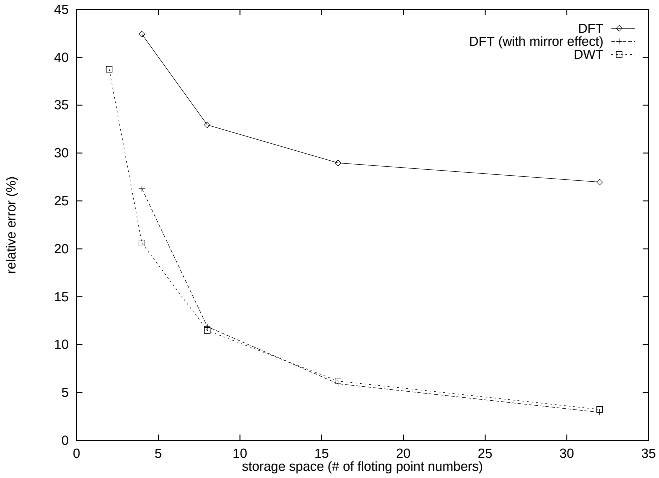

Figure 12: Average Errors for 100 Random Queries

Figure 11(a) shows the first query sequence, which is stock ABRX closing prices with 128 days time span. Figure 11(b) shows the average error when we use the DFT and DWT and keep different number of coefficients in the thresholding process. It shows that by using DFT and keeping only the first few coefficients as a distance measurement produces large average error, no matter how many coefficients are kept. By using the DWT and keeping that same number of coefficients we can achieve much smaller relative error than using the DFT. But with the conjugate property, the average relative error of the DFT is greatly reduced and comparable to the DWT. Figure 12 shows the average relative errors of 100 random queries. It suggests similar result.

4.3 Precision of Epsilon Query with Fourier and Wavelet Transforms

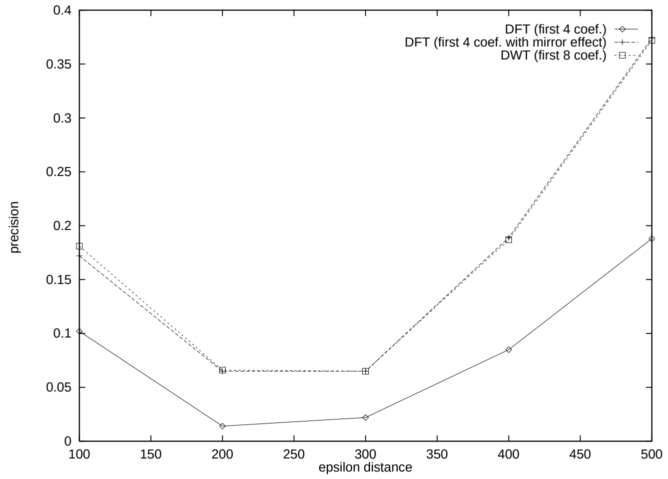

Figure 13: Average Precisions for 100 Random Queries

One of the most important categories of query in time-series database is epsilon distance query: Given a query sequence Q, find all the data sequence X_i in the database such that the distance between Q and X_i is below a threshold d, i.e., $D(Q, Xi) <= d$. In other words, find all similar time sequences in the database to a query sequence Q with similarity threshold d. For example, when the similarity threshold is 200, there are 3 data sequences in the data base which are under that range. As described in Section 2.1, according to Parseval's theorem, the transformation/thresholding processed will always result in underestimating the distance in the transformed domains; i.e., no false dismissal but false hits. We want to reduce the size of the estimated result set to be as close to the actual result set as possible, such that we can save more real distances computations in the post processing stage; i.e., increase precision to reduce false admissal.

The precision of an epsilon query is defined as

$$PrecisionOfEpsilonQuery(Q) = \frac{|ActualResultSet(Q)|}{|EstimatedResultSet(Q)|} \tag{9}$$

Figure 13 shows the epsilon distance query precision result of 100 random queries. We can see that the precisions of epsilon distance queries using the DWT are similar to the DFT. The result suggests that the DWT is not likely to produce a smaller estimated epsilon distance query result set than the DFT if we consider the conjugate property of DFT.

5 Conclusion

The DFT has been traditionally used as the de facto processing function for time-series data because time-series data can be treated as a 1-dimension signal and DFT is a proven technique that has been widely used in signal processing domain. In this paper, we explore the feasibility of replacing the DFT by DWT with a comprehensive analysis of the DFT and DWT as matching functions in time-series databases. Our results show that although the DWT based technique has several advantages, e.g., the DWT has complexity of $O(N)$ whereas DFT is $O(N \log N)$, DWT does not reduce relative matching error and does not increase query precision in similarity search. We conclude that, by exploiting the conjugate property of DFT in the real domain, the DFT-based and DWT-based techniques yield not much different behavior in the context of similarity search in time-series databases.

Our suggestions for future work are to explore the unique properties of DWT which the DFT does not have; e.g, multiresolutional property. For example, since DWT is inherently multiresolutional, it is possible to match time-series data at different resolutions concurrently. More efficient hierarchical searching structures other than the R-tree families can be derived by exploring this unique multiresolution property of the DWT.

References

[1] Kin pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 126–133. IEEE Computer Society, 1999.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Procedings of the 4th Int'l Conference on Foundations of Data Organization and Algorithms*, pages 69–84, Chicago, Oct 1993.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence MAtching in Time-Series Databases. In *Procedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 419–429, Minneapolis, May 1994.

[4] Davood Rafiei and Alberto Mendelzon. Similarity-Based Queries for Time Series Data. In *Procedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 13–24, Tucson, Arizona, May 1997.

[5] Davood Rafiei and Alberto Mendelzon. Efficient Retrieval of Similar Time Sequences Using DFT. In *Procedings of the International Conference on Foundations of Data Organizations and Algorithms - FODO 98*, Kobe, Japan, November 1998.

[6] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. In *Procedings of the 21th VLDB Conference*, pages 490–501, Zurich, Switzerland, 1995.

[7] Byoung-Kee Yi, H.V. Jagadish, and Christos Faloutsos. Efficient Retrieval of Similar Time Sequences under Time Warping. In *Procedings of the International Conference on Data Engineering - ICDE 98*, pages 201–208, Orlando, Florida, Feburary 1998.

[8] Sanghyun Park, Wesley W. Chu, Jeehee Yoon, and Chihcheng Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *Proceedings of the 16th International Conference on Data Engineering, 28 February - 3 March, 2000, San Diego, California*, pages 23–32. IEEE Computer Society, 2000.

[9] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Stott Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proceedings of the 16th International Conference on Data Engineering, 28 February - 3 March, 2000, San Diego, California*, pages 33–42. IEEE Computer Society, 2000.

[10] C. Sidney Burrus, Remesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer.* Prentice-Hall, Inc., 1998.

[11] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann Publishers, Inc., 1996.

Appendix A: Mathematics for Discrete Fourier Transform (DFT)

The n-point DFT of a signal $x = [x_t], t = 0, ..., n-1$ is defined to be a sequence X of n complex numbers $X_f, f = 0, ..., n-i$, given by

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t e^{-j2\pi ft/n}, f = 0, 1, ..., n-1 \tag{10}$$

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

$$x_t = 1/\sqrt{n} \sum_{f=0}^{n-1} X_f e^{j2\pi ft/n}, t = 0, 1, ..., n-1 \tag{11}$$

X_f is a complex number. If the signal is real, then X_0 is real.

The energy of Signal x is defined as the sum of energies at every point of the sequence:

$$E(x) \equiv \|x\|^2 \equiv \sum_{t=0}^{n-1} |x|^2 \tag{12}$$

The Parseval's Theorem

$$\sum_{i=0}^{n-1} |x_i|^2 = \sum_{j=0}^{n-1} |X_j|^2 \tag{13}$$

we have

$$\|x-y\|^2 = \|X-Y\|^2 \tag{14}$$

Appendix B: Minimizing Thresholding Error

Let the original function be

$$f(x) = \sum_{i=1}^{m} c_i u_i(x) \tag{15}$$

and let $\pi(i)$ be a permutation of $1, ..., m$ and the function after thresholding is

$$\hat{f}(x) = \sum_{i=1}^{\hat{m}} c_{\pi(i)} u_{\pi(i)}(x) \tag{16}$$

where $\hat{m} < m$. Then the L^2 error between the original function and the thresholded one is given by

$$\left\|f(x) - \hat{f}(\hat{x})\right\|^2 = \langle f(x) - \hat{f}(\hat{x}) | f(x) - \hat{f}(\hat{x})\rangle = \sum_{i=\hat{m}+1}^{m} \sum_{j=\hat{m}+1}^{m} c_{\pi(i)} c_{\pi(j)} \langle u_{\pi(i)} | u_{\pi(j)}\rangle = \sum_{i=\hat{m}+1}^{m} (c_{\pi(i)})^2 \tag{17}$$

if the basis is orthonormal. In order to minimize this error for any give $\hat{m}$, $\pi(i)$ satisfies $|c_{\pi(1)}| \geq ... \geq |c_{\pi(m)}|$. The complete proof can be found in [11].