Efficient Time Series Matching by Wavelets

By

CHAN, Kin Pong

Supervised By :

Prof. Wai-Chee FU Ada

Submitted to the Division of Department of Computer Science & Engineering

in partial fulfillment of the requirements for the degree of Master of Philosophy

at the

Chinese University of Hong Kong

JUNE, 1999

Acknowledgments

First of all, I would like to thank Prof. Ada Fu, my supervisor, for her guidance and patience. My research could not have been done reasonably without insightful advice from her. For the past two years, she gave me encouragement, support, and guidance on my papers (KDD98 and ICDE99) and my thesis. It would not be possible to get my paper published without her, and not even to Australia for my presentation.

My great gratitude goes to Prof. Man-Hon Wong and Prof. Pheng-Ann Heng, who marked my term paper and gave me valuable suggestions, Prof. Chak-Kuen Wong and Prof. Kwong-Sak Leung, who inspired me on thinking and research, Prof. Chun-Hung Cheng, who gave supervision on my final year project at the Department of System Engineering, CUHK.

My gratitude goes to the Department of Computer Science & Engineering, CUHK. It provides the best equipment and office environment required for high quality research to our students.

Finally, I thank to my fellow colleagues, who helped me in solving the programming, computer and official problems, and enlightened me to the new research ideas. They are (not in order) Kam-Wing Chu (advisor and database tutor), Chun-Hing Cai, King-Lam Cheung (database advisors), Hong-Ki Chu (always my project partner), Shing-Kwong Cheung (AI tutor), Tien-Tsin Wong and Chi-Wing Fu (computer graphics tutors), Yiu-Fai Fung (computer architecture tutor), Chun-Hung Cheng, Wang-Wai Kwong, Po-Shan Kam, Wai-Ching Wong, and Wai-Chiu Wong (database research group), Yuk-Ming Chan (IR tutor), Po-Shun Ngan (database and AI advisor), Tin-Lun Poon (numerical methods advisor), Man-Lee Liu (microprocessor tutors), King-Kwok Ng, Kwok-Leung Yu, Wai-Kwong Leung, Yuk-Chung Wong (discussion group).

... and they bring happiness to my university life.

Efficient Time Series Matching by Wavelets

submitted by

CHAN, Kin Pong

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Time series indexing has aroused much interest recently. Time series stored as feature vectors can be indexed by multi-dimensional index trees like R-Tree for fast retrieval. Due to the dimensionality curse problem, transformations are applied to time series to reduce the number of dimensions of the feature vectors while preserving most of the information. The transformation commonly used is the Discrete Fourier Transform (DFT) which maps the time series to the frequency domain. There are also different transformations available like Discrete Wavelet Transform (DWT), Karhunen-Loeve (K-L) transform or Singular Value Decomposition (SVD).

While the use of DFT and K-L transform or SVD have been studied in the literature, to our knowledge, there is no in-depth study on the application of DWT on this problem. In this paper, we propose to use Haar Wavelet Transform for time series indexing.

The major contributions are: (1) we show that Euclidean distance is preserved in the Haar transformed domain and no false dismissal will occur, (2) we show that Haar transform can outperform DFT through experiments, (3) a new similarity model is suggested to accommodate vertical shifts of time series, (4)

a two-phase method is proposed for efficient n-nearest neighbor query in time series databases, and (5) we propose two efficient strategies for approximation of time warping distance and show experimentally that they achieve significant speedup; The approximation function is also shown to be effective in suppressing the number of false alarms when acting as filtering function.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Time series data are of growing importance in many new database applications, such as data warehousing and data mining [4, 13, 2]. A time series (or time sequence) [1] is a sequence of real numbers, each number representing a value for an interested attribute at a time point. Typical examples include stock prices or currency exchange rates, the volume of product sales, biomedical measurements, weather data, etc ... collected over time. Hence, time series databases supporting fast retrieval of time series data and similarity queries are desired. For instance, we may want to retrieve stock prices around February 1997; or look for stocks that have a sharp drop after some price consolidations; or find stocks with similar price movements for a given stock. For these types of queries, approximate matching rather than an exact matching is needed.

In order to depict the similarity between two time series, we have to define a similarity measurement during the matching process. Given two time series $\vec{x} = (x_0, x_1, ..., x_{n-1})$ and $\vec{y} = (y_0, y_1, ..., y_{n-1})$, a standard approach is to compute the Euclidean distance $D(\vec{x}, \vec{y})$ between time series $\vec{x}$ and $\vec{y}$

$$D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} |x_i - y_i|^2 \right)^{\frac{1}{2}} \tag{1.1}$$

[1] We shall use the terms *time series* and *time sequence* interchangeably.

By using this similarity model, we can retrieve similar time series by considering distance $D(\vec{x}, \vec{y})$.

In order to support efficient retrieval and matching of time series, we resort to indexing to speed up the searching time. The general strategy in time series indexing and matching is depicted in Figure 1.1.

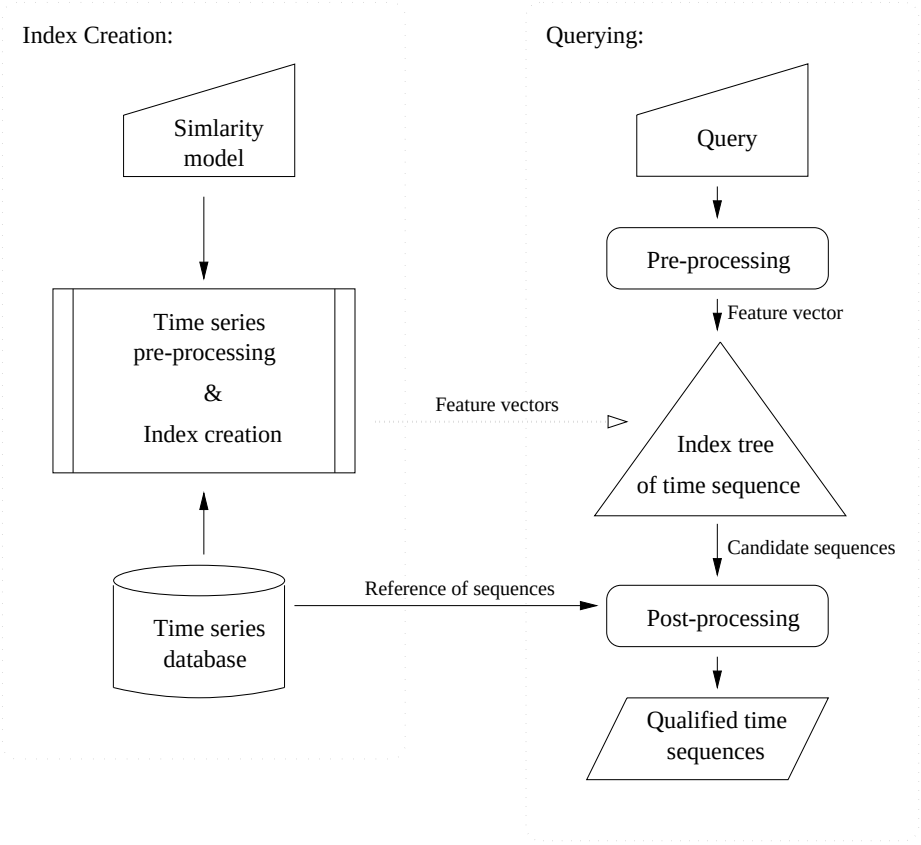

Figure 1.1: Time Series Indexing

For index creation, the time series are pre-processed, the pre-processing may include normalization, transformation, noise reduction, etc ... to produce a set of feature vectors. These vectors are then inserted into a multi-dimensional index tree, on which user can raise query. Upon the arrival of a query, pre-processing is done the same way as for the time series database in index creation. The feature vector acquired is matched against the index tree, which results in a set of candidate sequences. Non-qualified sequences or *false alarms* are filtered in a

post-processing step, by matching with the query sequence in time domain using full dimension. Qualified sequences are thus reported to the user.

In the pre-processing step of time series, some important issues have to be considered:

1. *Dimensionality reduction* – Many multi-dimensional indexing methods [19, 12, 9, 28] such as the R-Tree and R*-Tree [28, 9, 17] scale exponentially for high dimensionalities, eventually reducing the performance to that of sequential scanning or worse. Therefore, transformation is applied to map the time sequences to a new feature space of a lower dimensionality. The energy of the time sequence should be concentrated on as few coefficients as possible in the new space so that they are sufficient to differentiate between two sequences. These coefficients constitute a feature vector of a time sequence, which are inserted into multi-dimensional index tree.

2. *Completeness and Effectiveness* – In many cases, Euclidean distance is used as a similarity measure. When the number of dimensions is reduced, to avoid missing any qualifying object, the Euclidean distance in the reduced k-dimensional space should thus be *less than* or *equal to* the Euclidean distance between the two original time sequences. One of the transformations that satisfies this condition is the *Discrete Fourier Transform* (DFT) [2, 36]. It is shown by Parseval's Theorem [36] that the Euclidean distance is preserved in both frequency and time domains. The power concentration of a transformation on the reduced dimensionality should be effective to ensure a small amount of false alarms, which are filtered in the post-processing step.

3. *Nature of data series* – The effectiveness of *power concentration* of a particular transformation depends on the nature of the time series. The worst-case signal for DFT is *white noise*, where DFT fails to concentrate energy into

the first few coefficients and leads to tremendous amounts of false alarms. It is believed that only *brown noise* or *random walks* exists in real signals. In particular, stock movements and exchange rates can be modeled successfully as random walks in [16], for which a skewed energy spectrum can be obtained.

Discrete Fourier Transform (DFT) has been one of the most commonly used techniques. However, it misses the important feature of time localization. *Piecewise Fourier Transform* [2] [36] is proposed to mitigate this problem, but the size of the *pieces* leads to other problems. While large *pieces* reduce the power of multi-resolution, small *pieces* have weakness in modeling low frequencies.

1.1 Wavelet Transform

Wavelet Transform (WT), or *Discrete Wavelet Transform* (DWT) [15, 26] has been found to be effective in replacing DFT in many applications in computer graphics[43], image [35], speech [1] and signal processing [10, 5]. We propose to apply this technique in time series for dimension reduction and content-based search. DWT is a discrete version of WT for numerical signal. Although the potential application of DWT in this problem was pointed out in [33], no further investigation has been reported to our knowledge. Hence, it is of value to conduct studies and evaluations on time series retrieval and matching by means of wavelets.

The advantage of using DWT is *multi-resolution* representation of signals. It has the *time-frequency localization* property. Thus, DWT is able to give location in both time and frequency. For instance, the wavelet representation of a musical score can tell when the tones occur and what their frequencies are. Therefore,

[2] or short-time Fourier Transform (STFT)

wavelet representations of signals bear much more information than that of DFT, in which only frequencies are considered. While DFT extracts the lower harmonics which represent the general shape of a time sequence, DWT encodes a coarser resolution of the original time sequence with its preceding coefficients.

The differences between matching signals of coarser resolutions and matching only their frequency contents are discussed and we show by experiments that *Haar Wavelet Transform* [3], [15], which is a commonly used wavelet transform, can outperform DFT significantly.

We propose a similarity definition to handle the problem of vertical shifts of time series. We propose an algorithm on n-nearest neighbor query for the proposed wavelet method. The algorithm makes use of the range query and dynamically adjusts the *range* by the property of Euclidean distance preservation of the wavelet transformation.

1.2 Time Warping

For the sake of robustness, Euclidean distance is frequently adopted as the similarity model in time series matching. Other similarity models may also be employed. The choice of appropriate model, in accordance with a particular kind of time series, gives rise to better interpretation and semantics of the similarity between two sequences. In the speech recognition field, *time warping* techniques [37, 13] are used extensively for similarity matching. The idea of time warping is that word recognition is usually based on matching pre-stored word templates against a waveform of continuous speech, converted into a discrete time series. Successful recognition strategies are based on the ability to match words approximately in spite of wide variations in timing and pronunciation. The time

[3] We shall use Haar wavelet transform and DWT interchangeably throughout this paper, unless specified particularly.

warping approach tries to align the time series and a specific word template so that some distance measure is minimized. On the contrary, Euclidean distance fails to offer time axis alignment since it matches linearly two time sequences based on the same, fixed time axis.

Algorithms for time warping are proposed in [41, 40, 34]. Although they are able to match sequences with time shifts, the computations involved are rather high. This imposes a restriction to use this technique in an online system where prompt response is demanded. In general, index tree supporting time warping distance could be built following the model in Figure 1.1. Unfortunately, unlike Euclidean distance, time warping distance does not satisfy the triangle inequality. The consequence is that we are unable to guarantee the retrieval of all qualified sequences for a given query, thus giving rise to *false dismissals*.

The details of time warping technique based index are described and its drawbacks are identified. We propose a novel approximation function based on wavelets for time warping distance, which results in lower time complexity by trading off tiny amount of accuracy. On the other hand, this approximation function can also be employed as the filtering function in the post-processing step of the querying model of Figure 1.1, which is shown to be both effective and efficient experimentally.

1.3 Outline of the Thesis

The thesis is organized as follows.

In Chapter 2, we will give some backgrounds and state the previous works in time series retrieval, including the similarity models used in time series matching and various indexing methodologies. We will pay particular attention to dimension reduction applied to time series database. Moreover, we will introduce other

time series retrieval approaches.

In Chapter 3, we will describe our proposed approach, time series matching by means of wavelets. First of all, two similarity models are defined for our time sequence matching technique. We will introduce wavelets transform, in particular the Haar wavelet transform. Important properties of feature extraction by Haar transform are discussed. Second, the overall strategy of applying DWT to dimension reduction is given and the method for nearest neighbor query in dimension reduction problem is proposed. Third, performance evaluations using real and synthetic data, together with scalability test are given.

In Chapter 4, we will give the details of time warping techniques, and describe our strategy in approximating time warping distance between two time series. Moreover, approximation function is applied to post-processing step as filtering function. Experiments are carried out to evaluate the performance.

We will give a conclusion and summarize our future work in Chapter 5.

Chapter 2

Related Work

Time series is primarily concerned with the study of the time variations of process. The states for a duration of n time units of a process can be represented by a vector of real numbers

$$\vec{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \\ \vdots \\ x_{n-1} \end{bmatrix} \tag{2.1}$$

where x_t is the state recorded at time t. For the sake of convenience, we depict time series in Equation (2.1) as $\vec{x} = \{x_0, x_1, ..., x_{n-1}\}$ unless otherwise specified.

2.1 Similarity Models for Time Series

The matching of text subsequences [8] is considered as one kind of time series matching. In [8], algorithms are presented to find all the occurrences of a pattern

in a text. There are various similarity models [14, 20, 18, 3] proposed in time series matching. A similarity function that can deal with outliers and different scaling factors is introduced in [14]. The basic idea is to find two longest common subsequences $s\vec{x}$ and $s\vec{y}$ in $\vec{x}$ and $\vec{y}$ respectively, with distance between $s\vec{x}$ and $s\vec{y}$ under ϵ. Scaling and outliers can be tolerated by applying *different* linear transformation to $s\vec{x}$ and $s\vec{y}$ to maximize the length l of the common sequence which is an indication of the similarity. A more detailed version of [14] can be found in [20]. The computation incurred in finding the linear transformation is quadratic, which can be modified to a linear-time randomized approximation of the transformation. If a quadratic algorithm is used, it seems that the computation time will be too long when searching the whole database. If a linear algorithm is chosen, accuracy is sacrificed. In real practice, a linear algorithm can be used as a pruning strategy and a post-processing step should be introduced to filter out false alarms.

In [18], the slope of sequence is considered. The slope of segments $\{x_i, x_{i+1}\}$ and $\{y_i, y_{i+1}\}$ for sequences $\vec{x}$ and $\vec{y}$ should be confined within a range [-ϵ, ϵ]. Two sequences are *slope similar* if

$$-\epsilon \leq (y_{i+1} - y_i) - (x_{i+1} - x_i) \leq \epsilon \tag{2.2}$$

Linear scale of sequence can be dealt with by comparing the slope of non-consecutive sequence points

$$-\epsilon \leq (y_i - y_{i-1}) - (x_{\lceil \frac{i}{s} \rceil} - x_{\lceil \frac{i}{s} \rceil - 1}) \leq \epsilon \qquad s \geq 1 \tag{2.3}$$

$$-\epsilon \leq (y_{\lceil \frac{i}{s} \rceil} - y_{\lceil \frac{i}{s} \rceil - 1}) - (x_i - x_{i-1}) \leq \epsilon \quad 0 < s < 1 \tag{2.4}$$

where s is a scaling factor, ratio of the length of $\vec{x}$ to $\vec{y}$ is $1 : s$. This similarity measure can reflect human interpretations in matching similar sequences to a higher extent. The *upward* and *downward* trends are emphasized. However,

dimension reduction of the first derivative of sequences are not possible. White noises will certainly exist in slope of segments of sequences. It is not very efficient to index by multi-dimensional index tree through DFT [2]. Instead, the author uses a dynamic hashing function for accessing sequences in order to reduce the number of disk accesses.

In [3], a similarity model with noise tolerance and translation is introduced. Two subsequences $\vec{x}$ and $\vec{y}$ of equal length are considered similar if one can be enclosed within an envelop of a specified width by allowing the amplitude of one of the two sequences to be scaled by any suitable amount and its offset adjusted appropriately such that the distance between them is within threshold ϵ. The similarity between two sequences with 'gap' has also been addressed. For a maximum gap size γ, a *stitching window* of size ω, two sequences $\vec{x}$ and $\vec{y}$ are said to be similar if after some removal of non-matching gap with size $\leq \gamma$, the total length of similar subsequence pairs with size equal to ω is greater than λ times the total length of $\vec{x}$ and $\vec{y}$. Small *atomic* sequence sets are indexed on R-Tree [28] family of structure that can represent all the original sequences up to amplitude scaling and offset. As a result, all atomic subsequence matches within a user-specified distance ϵ can be efficiently computed by doing self-join on this structure. The methodology to process sequences with gaps and amplitude variations is novel. Unfortunately, there is no experimental result provided in the original paper. We have no way to evaluate the effectiveness of the similarity model and the time complexity involved in self-join.

A general framework for similarity queries for time series is introduced in [31]. A pattern language, a transformation rule language, and a query language are defined in this framework which enable a formal definition of the notion of similarity. To specify objects that match a pattern approximately, we attach to patterns in some transformation rules defined in a transformation language. An object A is considered to approximate an object B, if B can be reduced to it

by a sequence of transformations. Moreover, a calculus-based query language is devised that is an extension of the tuple relational calculus with function symbols, and with some built-in predicates. The framework is further specialized in [22] to adapt to real-valued sequences. It shrinks the data sequences into *signatures*, and the signatures are searched instead of the real sequences, with further comparison being required only when a possible match is indicated. Comparisons can be made faster with shorter signatures than the original sequences. Framework that facilitates a broad class of approximate queries over sequences is proposed in [42], where a more general notion of approximation appropriate for the complex queries is presented. For this kind of framework, there is no general rule to seek an appropriate transformation for a particular type of time series. Therefore, the effort still remains to the domain experts.

2.2 Dimensionality Reduction

Discrete Fourier Transform is often used for dimension reduction [2, 23] to achieve efficient indexing. An index built by means of DFT is also called an **F-index** [2]. It works as follows. Given N sequences, all of the same length n, we apply the n-point DFT to sequence $\vec{x}$

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t \exp(-j2\pi \frac{ft}{n}) \quad f = 0, 1, \ldots, n-1 \tag{2.5}$$

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

$$x_t = 1/\sqrt{n} \sum_{f=0}^{n-1} X_f \exp(j2\pi \frac{ft}{n}) \quad t = 0, 1, \ldots, n-1 \tag{2.6}$$

X_f is a complex number (with the exception of X_0, which is real provided that the signal $\vec{x}$ is real). Suppose the DFT of a time sequence $\vec{x}$ is denoted by $\vec{X}$. There is a mapping of signals from n-dimensional time domain to n-dimensional frequency domain. For many applications such as stock data, the low frequency components are located at the preceding coefficients of $\vec{X}$ which represent the general trend of the time sequence $\vec{x}$ where most energy is concentrated. These coefficients can be indexed in an R-Tree or R*-Tree for fast retrieval. In a query, a time sequence of length n and a tolerance ϵ are given. To resolve the query, n-point DFT is applied to the query sequence and again first f_c features are used for similarity matching by F-index, which returns all sequences within Euclidean distance ϵ. In most previous work, range querying is considered. A **range query** (or epsilon query) evaluation returns sequences with Euclidean distance within ϵ from the query point.

Parseval's Theorem [36] shows that the Euclidean distance between two signals $\vec{x}$ and $\vec{y}$ in time domain is the same as their Euclidean distance in frequency domain

$$\|\vec{x} - \vec{y}\|^2 \equiv \|\vec{X} - \vec{Y}\|^2 \tag{2.7}$$

Therefore, F-index may raise false alarms, but guarantees no false dismissal. After a range query in the F-index, qualified sequences are then checked against the query sequence in the original time domain. This post-processing step eliminates the false alarms.

F-index is further generalized and subsequence matching are proposed in [23]. This is called the ST-index which permits sequence query of varying length. Instead of mapping directly the whole sequence into the k-dimensional space, a *sliding window* of size ω is covered on the original sequence. Upon a shift of the sliding window, ω-point DFT is applied to the covered sequence and f_c coefficients are extracted as feature vector. Each time sequence is broken up into pieces of subsequences by a sliding window with a fixed length ω for DFT.

In view of the fact that feature points in nearby offsets will form a trail due to the effect of stepwise sliding window, the *minimum bounding rectangle* (MBR) of a trail is being indexed in an R-Tree instead of the feature points themselves. When a query arrives, all MBRs that intersect the query region are retrieved and their trails are matched. However, additional false alarms are introduced as there are cases where sub-trails do not intersect the query region while their MBRs do.

New similarity models are applied to F-index based time series matching in [38]. It achieves time warping, moving average, and reversing by applying transformations to feature points in the frequency domain. Given a query $\vec{q}$, a new index is built by applying a transformation to all points in the original index and feature points with a distance less than ϵ from $\vec{q}$ are returned. However, a lot of computations are involved in building the new index, which has a great impact on the actual query performance.

Another method that has been employed for dimension reduction is *Karhunen-Loeve* (K-L) transform [46]. (This method is also known as Singular Value Decomposition (SVD) [33], and is called Principle Component analysis in statistical literature.) Given a collection of n-dimensional points, we project them on a k-dimensional sub-space where $k < n$, maximizing the variances in the chosen dimensions. The key weakness of K-L transform is the deterioration of performance upon incremental update of the index, as the projection axes are pre-determined (static) by the covariance matrix in the first collection of feature vectors. Although the projection is optimal for a fixed set of vectors, new projection matrix should be re-calculated and the index tree has to be reorganized periodically to keep up the search performance. Efficient methods for incremental update of SVD-based index are discussed in [32].

Clustering with Singular Value Decomposition (CSVD) is introduced in [44]

to improve the efficiency of standard SVD. Generally, SVD relies on global information derived from all the vectors in the dataset, which is more effective for datasets consisting of homogeneously distributed feature vectors. For databases with heterogeneously distributed vectors, more efficient representation can be generated by subdividing the vectors into more similar groups such that the points in each group or cluster are more amenable to dimensionality reduction than the original dataset by SVD. It is shown experimentally that CSVD achieves higher dimensionality reduction than SVD in terms of total variance preserved. However, the overhead in updating the clusters as well as the SVD axes makes it less attractive to other dynamic-based methods.

The Pyramid-Technique adapting well to high dimensional data queries is introduced in [11]. In contrast with all other index structures, the performance of the Pyramid-Technique does not deteriorate when processing range queries on data of high dimensionality. It is based on a special partitioning strategy, which divides the data space into pyramids sharing the center point of the space as a top. Single pyramid is cut into slices parallel to the basis of the pyramid which is shown in Figure 2.1.

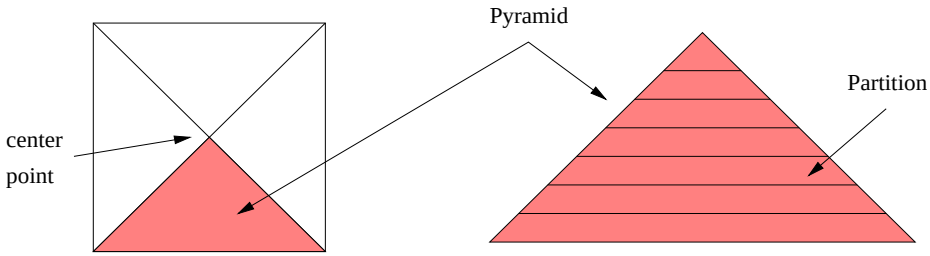

Figure 2.1: Partitioning the data space into pyramids (2-dimensional data)

This partition enables a mapping from the given d-dimensional space to a 1-dimensional space which can be handled efficiently by B^+-Tree. An entry in B^+-Tree composes of the 1-dimensional index key and the original d-dimensional

feature. However, it is shown in [11] that this technique performs worse than the sequential scan for very skewed queries. Moreover, range query is evaluated only and no insight into nearest neighbor query has been shown.

2.3 Wavelet Transform

Wavelets are basis functions used in representing data or other functions. Wavelet-based algorithms process data at different *scales* or *resolutions* in contrast with DFT where only frequency components are considered. The origin of wavelets can be traced to the work of Karl Weierstrass [45] in 1873. The construction of the first orthonormal system by Haar [29] is an important milestone. Haar basis is still a foundation of modern wavelet theory. Another significant advance is the introduction of a nonorthogonal basis by Dennis Gabor in 1946 [24]. In this work we shall advocate the use of the Haar wavelets in the problem of time series search.

To get some idea of what wavelet transform is, lets consider its loose definition. A signal or a function $f(t)$ can often be better analyzed, described, or processed if expressed as a linear decomposition by

$$f(t) = \sum_{l} a_l \psi_l(t) \tag{2.8}$$

where l is an integer index for the finite or infinite sum, a_l are the real-valued expansion coefficients, and $\psi_l(t)$ are a set of real-valued functions of t called the expansion set. If the expansion in Equation (2.8) is unique, the set is called a *basis* for the class of functions that can be so expressed. If the basis is orthogonal, meaning

$$\langle \psi_k(t), \psi_l(t) \rangle = \int \psi_k(t)\psi_l(t)\,dt = 0 \quad \text{for } k \neq l \tag{2.9}$$

then the coefficients can be calculated by the inner product

$$a_k = \langle f(t), \psi_k(t) \rangle = \int f(t)\psi_k(t)\,dt \tag{2.10}$$

One can see that substituting Equation (2.8) into Equation (2.10) and using Equation (2.9) gives the single a_k coefficient. For a Fourier series, the orthogonal basis functions $\psi_k(t)$ are $\sin(k\omega_0 t)$ and $\cos(k\omega_0 t)$ with frequencies of $k\omega_0 t$.

For the *wavelet expansion*, a two-parameter system is constructed such that Equation (2.8) becomes

$$f(t) = \sum_k \sum_j a_{j,k}\psi_{j,k}(t) \tag{2.11}$$

where both j and k are integer indices and the $\psi_{j,k}(t)$ are the wavelet expansion functions that usually form an orthogonal basis. The set of expansion coefficients $a_{j,k}$ are called the Discrete Wavelet Transform (DWT) of $f(t)$ and Equation (2.11) is the inverse transform.

2.4 Similarity Search under Time Warping

The ability of time warping to match sequences with time shifts makes it an important similarity model in speech recognition, since human speech consists of varying durations and paces.

The time warping distance for two sequences $\vec{x}$ and $\vec{y}$ is defined as

$$
\begin{aligned}
D_{timewarp}(\langle\rangle, \langle\rangle) &= 0, \\
D_{timewarp}(\vec{x}, \langle\rangle) &= D_{timewarp}(\langle\rangle, \vec{y}) = \infty, \\
D_{timewarp}(\vec{x}, \vec{y}) &= D_{base}(Head(\vec{x}), Head(\vec{y})) \\
&\quad + \min \left\{ \begin{array}{ll} D_{timewarp}(\vec{x}, Rest(\vec{y})) & \text{x - stutter} \\ D_{timewarp}(Rest(\vec{x}), \vec{y}) & \text{y - stutter} \\ D_{timewarp}(Rest(\vec{x}), Rest(\vec{y})) & \text{no stutter} \end{array} \right\}
\end{aligned} \tag{2.12}
$$

where $\langle\rangle$ denotes a null sequence. D_{base} can be any of the distance functions, like the city-block distance, although our primary concern is the Euclidean distance. Also note that this definition does not require two sequences to be of the same length. The symbols can be looked up in Table 2.1.

Symbol	Definition
D_{base}	base distance function, e.g., D_1 and D_2
$D_{timewarp}$	time warping distance function
$\vec{x}$, $\vec{y}$	time sequences
$\langle\rangle$	null time sequence
$Head(\vec{x})$	the first element of $\vec{x}$
$Rest(\vec{x})$	the remaining elements of $\vec{x}$ other than the first

Table 2.1: Notations in the definition of time warping distance

As for Euclidean distance, searching techniques are proposed to support the retrieval of similar time series based on the increasingly important time warping distance. In [47], a time series database supporting time warping is proposed whose strategy is shown in Figure 2.2.

It follows the architecture of the general strategy shown in Figure 1.1, by further specifying the transformation used in index creation/pre-processing and the filtering functions in post-processing. To elaborate, two steps are involved. First, K-L transform is applied to map the original time sequences to lower dimension feature vectors, then a multi-dimensional index is built (Fastmap index). If we are looking for set of sequences $\vec{y}$ within time warping distance

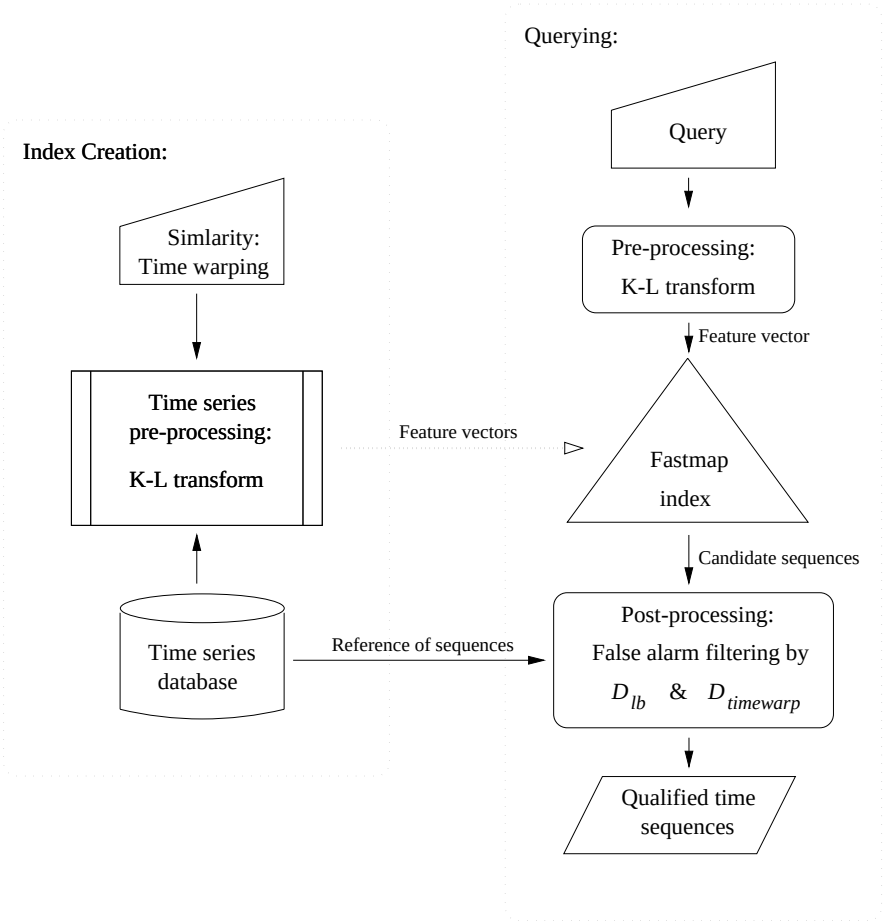

Figure 2.2: Time series indexing supporting time warping distance

$\epsilon_{timewarp}$ of query $\vec{x}$, i.e. $D_{timewarp}(\vec{x}, \vec{y}) \leq \epsilon_{timewarp}$, then the Fastmap index is queried using the same search range $\epsilon_{timewarp}$ with *Euclidean distance function*, i.e. $D(\vec{x}, \vec{y'}) \leq \epsilon_{fastmap} = \epsilon_{timewarp}$, where $\vec{y'}$ are set of candidate sequences containing some false alarms as well as *false dismissals*. As time warping distance does not satisfy the triangle inequality, any indexing technique which assumes the triangle inequality, can not avoid producing false dismissals, so does the Fastmap index. We just use $\epsilon_{timewarp}$ as an estimation to the search range of Euclidean distance function D in order to retrieve a smaller set of candidate sequences in the database. Second, a filtering function is proposed to prune away false alarms from the candidate sequences in a post-processing step. This lower bound distance function D_{lb} underestimates the time warping distance function, such that $D_{lb}(\vec{x}, \vec{y}) \leq D_{timewarp}(\vec{x}, \vec{y})$. To get a better intuition and insight of D_{lb}, consider an illustration in Figure 2.3, with time sequences $\vec{x}$ (solid) and $\vec{y}$ (dashed), having corresponding vertical ranges $R_{\vec{x}}$ and $R_{\vec{y}}$ overlapped.

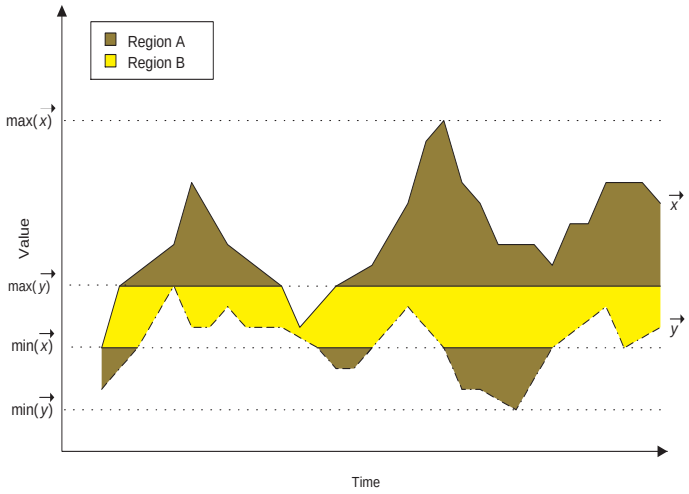

Figure 2.3: Intuitive idea behind D_{lb}

The shaded region between the two sequences is separated into two disjoint parts A and B. A is the shaded region above $\max(\vec{y})$ and below $\min(\vec{x})$, and B lies in between. Note that $D_{timewarp}(\vec{x}, \vec{y})$ is equal to area(A) + area(B) after time warping. Time warping attempts to minimize this area sum, however, it

may reduce area(B), but not area(A). The reason is that the horizontal span of Region A is widen upon stuttering (refer to Equation (2.12)), leading to an increase in area(A). As time warping attempts to minimize area(A) + area(B), area(B) should only decrease as area(A) increases, such that the area sum can be reduced. This gives rise to the following observation

$$\text{area}(A) \leq \text{area}(A') \leq \text{area}(A') + \text{area}(B') = D_{timewarp}(\vec{x}, \vec{y})$$

where A' and B' denote A and B after time warping respectively. This observation provides the fundamentals to the definition of the lower bound distance function

$$D_{lb}(\vec{x}, \vec{y}) =$$

$$\begin{cases} \sum_{x_i > \max(\vec{y})} |x_i - \max(\vec{y})| + \sum_{y_j < \min(\vec{x})} |y_j - \min(\vec{x})| & \text{if } R_{\vec{x}} \text{ and } R_{\vec{y}} \text{ overlap} \\ \sum_{x_i > \max(\vec{y})} |x_i - \max(\vec{y})| + \sum_{x_i < \min(\vec{y})} |x_i - \min(\vec{y})| & \text{if } R_{\vec{x}} \text{ encloses } R_{\vec{y}} \\ \max(\sum_{i=1}^{m} |x_i - \max(\vec{y})|, \sum_{j=1}^{n} |y_j - \min(\vec{x})|) & \text{if } R_{\vec{x}} \text{ and } R_{\vec{y}} \text{ are disjoint} \end{cases} \tag{2.13}$$

Instead of merely using $D_{timewarp}$ ($O(\|\vec{x}\| \times \|\vec{y}\|)$ complexity), D_{lb} (linear complexity) can be used as a filter in addition to prune away quickly non-qualified time series in the candidate sequences set $\vec{y'}$. As a result of $D_{timewarp}$ underestimation, some false alarms may not be pruned by D_{lb}, and any remained sequences in $\vec{y'}$ are checked against the $D_{timewarp}$ to obtain the answer set $\vec{y}$. Experiments in [47] show that a significant speedup can be achieved by trading off a tiny amount of false dismissals. We will describe the drawbacks of this approach in detail in Chapter 4 and suggest a more efficient mechanism.

Chapter 3

Dimension Reduction by Wavelets

3.1 The Proposed Approach

Following a trend in the disciplines of signal and image processing, we propose to study the use of wavelet transformation for the time series indexing problem. Before we go into the details of our proposed techniques, we would first like to define the similar models used in sequence matching. The first definition is based on the Euclidean distance $D(\vec{x}, \vec{y})$ between time sequences $\vec{x}$ and $\vec{y}$.

Definition 1 Given a pre-determined threshold ϵ, two time sequences $\vec{x}$ and $\vec{y}$ of equal length n are said to be **similar** if

$$D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} (y_i - x_i)^2 \right)^{\frac{1}{2}} \leq \epsilon \tag{3.1}$$

■

A shortcoming of Definition 1 is demonstrated in Figure 3.1. Consider the two time sequences $\vec{x}$ and $\vec{y}$. From human interpretation, $\vec{x}$ and $\vec{y}$ may be quite

similar because $\vec{y}$ can be shifted up vertically to obtain $\vec{x}$. However, if Definition 1 is used as the similarity measure, they will be considered *not* similar because errors are accumulated at each pair of x_i and y_i. Hence, we attempt to introduce another similarity model.

Definition 2 Given a pre-determined threshold ϵ, two time sequences $\vec{x}$ and $\vec{y}$ of equal length n are said to be **v-shift similar** if

$$D(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} ((y_i - x_i) - (y_A - x_A))^2 \right)^{\frac{1}{2}} \leq \epsilon \tag{3.2}$$

where

$$x_A = \frac{1}{n} \sum_{i=0}^{n-1} x_i \quad \text{and} \quad y_A = \frac{1}{n} \sum_{i=0}^{n-1} y_i$$

■

From Definition 2, any two time sequences are said to be *v-shift* similar if the Euclidean distance is less than or equal to a threshold ϵ neglecting their vertical offsets from x-axis. This definition can give a better estimation of the similarity between two time sequences with similar trends running at two completely different levels.

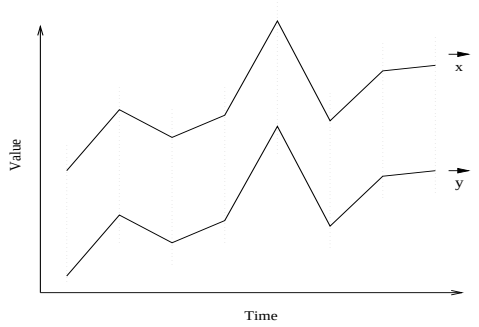

Figure 3.1: Example of vertical shifts of time sequences

3.1.1 Haar Wavelets

We want to have a decomposition that is fast to compute, requires little storage for each sequence. The Haar wavelet is chosen for the following reasons: (1) it allows good approximation with a subset of coefficients, (2) it can be computed quickly and easily, requiring linear time in the size of the sequence and simple coding, and (3) it preserves Euclidean distance (see Section 3.1.3).

The *Haar wavelets* are defined as

$$\psi_i^j(x) = \psi(2^j x - i) \quad i = 0, \ldots, 2^j - 1 \tag{3.3}$$

where

$$\psi(t) = \begin{cases} 1 & 0 < t < 0.5 \\ -1 & 0.5 < t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

together with a *scaling* function

$$\varphi(t) = \begin{cases} 1 & 0 < t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

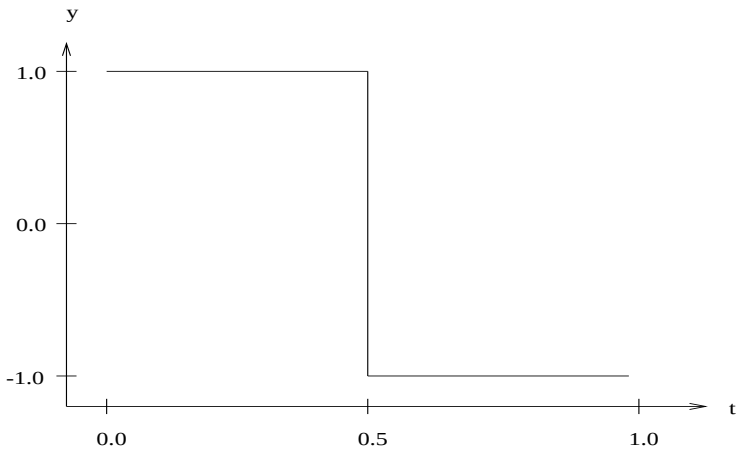

Figure 3.2: Haar wavelet for $\psi_0^0(t)$

They are shown graphically in Figure 3.2 and Figure 3.3 respectively.

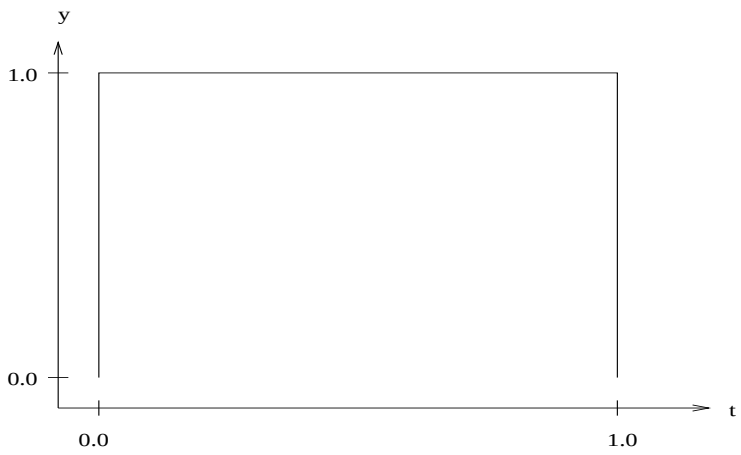

Figure 3.3: Haar scaling function

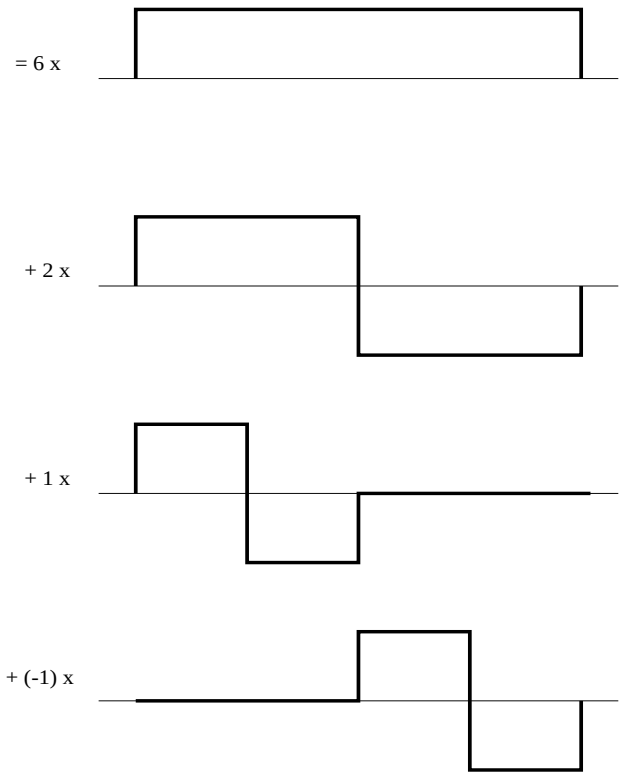

Figure 3.4: An example of Haar transform

Example For the piecewise constant function $f(t)$, we illustrate with an example taken from [15]:

$$f(t) = \begin{cases} 9 & 0 \leq t < 0.25 \\ 7 & 0.25 \leq t < 0.5 \\ 3 & 0.5 \leq t < 0.75 \\ 5 & 0.75 \leq t \leq 1 \end{cases} \tag{3.6}$$

We can express $f(t)$ as a linear combination of ψ and φ

$$f(t) = c\varphi(x) + d_0^0\psi_0^0(x) + d_0^1\psi_0^1(x) + d_1^1\psi_1^1(x) \tag{3.7}$$

which is shown in Figure 3.4. We notice that $c = 6, d_0^0 = 2, d_0^1 = 1$, and $d_1^1 = -1$.

These coefficients {6,2,1,-1} are actually the Haar transform of the discrete function $f(x) = \{9,7,3,5\}$. It should be pointed out that c is the *overall average value* of the whole time sequence, which is equal to $(9 + 7 + 3 + 5)/4 = 6$. ∎

Concrete mathematical foundations can be found in [15, 27] and related implementations in [21].

Haar transform can be seen as a series of averaging and differencing operations on a discrete time function. We compute the average and difference between every two adjacent values of $f(x)$. The procedure to find the Haar transform of a discrete function $f(x) = \{9,7,3,5\}$ is shown below.

Example

Resolution	**Averages**	**Coefficients**
4	{9,7,3,5}	
2	{8,4}	{1,-1}
1	{6}	{2}

Resolution 4 is the full resolution of the discrete function $f(x)$. In Resolution 2, {8,4} are obtained by taking averages of {9,7} and {3,5} at Resolution 4 respectively. {1,-1} are the differences of {9,7} and {3,5} divided by two respectively.

This process is continued until a resolution of 1 is reached. The Haar transform $H(f(x)) = \{c, d_0^0, d_0^1, d_1^1\}$ = {6,2,1,-1} is obtained which composes of the last average value 6 and the coefficients found on the right most column, 2, 1, and -1. It should be pointed out that c is the *overall average value* of the whole time sequence, which is equal to $(9 + 7 + 3 + 5)/4 = 6$. Different resolutions can be obtained by adding difference values back to or subtract difference from an average. For instance, {8,4} = {6+2,6-2} where 6 and 2 are the first and second coefficients respectively. This process can be done recursively until the full resolution is reached. ∎

Haar transform can be realized by a series of matrix multiplications as illustrated in Equation (3.8). Envisioning the example input signal $\vec{x}$ as a column vector with length [1] $n = 4$, an intermediate transform vector $\vec{w}$ as another column vector and Haar transform matrix **H**

$$\begin{bmatrix} x'_0 \\ d_0^1 \\ x'_1 \\ d_1^1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.8}$$

The factor 1/2 associated with the Haar transform matrix can be varied according to different *normalization* [2] conditions. After the first multiplication of $\vec{x}$ and **H**, half of the Haar transform coefficients can be found which are d_0^1 and d_1^1 in $\vec{w}$ interleaving with some intermediate coefficients x'_0 and x'_1. Actually, d_0^1 and d_1^1 are the last two coefficients of the Haar transform. x'_0 and x'_1 are then extracted from $\vec{w}$ and put into a new column vector $\vec{x'} = [x'_0\ x'_1\ 0\ 0]^T$. $\vec{x'}$ is treated as the new input vector for transformation. This process is done recursively until one element is left in $\vec{x'}$. In this particular case, c and d_0^0 can be found in the second

[1] As for DFT, the length of the signal is restricted to numbers which are power of 2.
[2] The normalization is described in Section 3.1.3.

iteration.

The complexity of Haar transform can be evaluated by considering the number of operations involved in the recursion process.

Lemma 1 Given a time sequence of length n where n is an integral power of 2, the complexity of Haar transform is $O(n)$.

Proof: There are totally n matrix additions *or* subtractions in the first iteration of matrix operation. The size of the input vector is halved in each iterations onwards. The total number of operations are formulated as

$$\overbrace{n + n/2 + \cdots + 2}^{\log_2 n} = 2\frac{2^{\log_2 n} - 1}{2 - 1} = 2(n-1)$$

which is bounded by $O(n)$. ∎

3.1.2 DFT versus Haar Transform

Our motivation of using Haar transform to replace DFT is based on several evidences and observations, some of which are also the reasons why the use of wavelet transforms instead of DFT is considered in areas of image and signal processing.

1. Better Pruning Power

   The nature of the Euclidean distance preserved by Haar transform and DFT are different. In DFT, comparison of two time sequences is based on their low frequency components, where most energy is presumed to be concentrated on. On the other hand, the comparison of Haar coefficients is matching a gradually refined resolution of the two time sequences. The time-frequency localization property possessed by DWT may probably be

the reason for more effective pruning of Haar wavelets, such that fewer false alarms are produced which is confirmed by experiments in Section 3.3. This in turn can save disk accesses as well as computation, especially when the time sequences are long and the size of the database is large.

2. Lower Complexity

   The complexity of Haar transform is $O(n)$ whilst $O(n \log n)$ computation is required for Fast Fourier Transform (FFT) [25]. Both impose restriction on the length of time sequence which must be an integral power of 2. Although these computations are all involved in pre-processing stage, the complexity of the transformation can be a concern especially when the dataset is large. From experiments, the pre-processing time for DFT is 3 to 4 times longer than Haar transform.

3. Better Similarity Model

   Apart from Euclidean distance, our model can easily accommodate *v-shift* similarity of two time sequences (Definition 2) at a little more cost. That is, the situation where vertically shifted signals can match is accommodated. On the contrary, previous study on F-index did not make use of this similarity model.

Note that similar to DFT, DWT will not require massive index re-organization because of database updating, which is a major drawback in using the K-L transform or SVD approach.

3.1.3 Guarantee of no False Dismissal

For FT and DFT, it is shown by Parseval's Theorem [36] that the energy of a signal conserves in both time and frequency domains. Parseval's Theorem also shows that this situation is true for wavelet transforms. Moreover, the Euclidean distances of both time and frequency domains are the same for DFT by Equation (2.7). This is a very important property in order that dimension reduction of sequence data is possible. It guarantees that no qualified time sequence will be rejected, thus no false dismissal. However, this property has not been shown for DWT in general, and not for the Haar wavelets. The following lemmas show how the Euclidean distance in time domain can be formulated in terms of the coefficients of Haar wavelet transform.

Lemma 2 Given a sequence $\vec{x} = \{x_0, x_1\}$ and a sequence $\vec{y} = \{y_0, y_1\}$. The Haar transforms of $\vec{x}$ and $\vec{y}$ are $H(\vec{x}) = \vec{s} = \{s_0, s_1\}$ and $H(\vec{y}) = \vec{r} = \{r_0, r_1\}$ respectively. Lengths of $\vec{x}$, $\vec{y}$, $\vec{s}$, and $\vec{r}$ are all equal to 2. Then Euclidean distance $D(\vec{x}, \vec{y})$ is $2^{\frac{1}{2}}$ times of Euclidean distance $D(\vec{s}, \vec{r})$

$$D(\vec{x}, \vec{y}) = 2^{\frac{1}{2}} D(\vec{s}, \vec{r}) \tag{3.9}$$

Proof: Express $\vec{s}$ in terms of $\vec{x}$ and $\vec{r}$ in terms of $\vec{y}$ by applying Equation (3.8) accordingly,

$$\vec{s} = \left\{ \frac{x_0 + x_1}{2} \ , \ \frac{x_0 - x_1}{2} \right\}$$
$$\vec{r} = \left\{ \frac{y_0 + y_1}{2} \ , \ \frac{y_0 - y_1}{2} \right\}$$

Square of Euclidean distance of $\vec{s}$ and $\vec{r}$

$$D^2(\vec{s}, \vec{r}) \quad = \quad \left(\frac{x_0 + x_1}{2} - \frac{y_0 + y_1}{2} \right)^2 + \left(\frac{x_0 - x_1}{2} - \frac{y_0 - y_1}{2} \right)^2$$

$$
\begin{aligned}
&= \left(\frac{x_0 - y_0}{2} + \frac{x_1 - y_1}{2}\right)^2 + \left(\frac{x_0 - y_0}{2} - \frac{x_1 - y_1}{2}\right)^2 \\
&= \left(\frac{x_0 - y_0}{2}\right)^2 + \left(\frac{x_1 - y_1}{2}\right)^2 + \left(\frac{x_0 - y_0}{2}\right)^2 + \left(\frac{x_1 - y_1}{2}\right)^2 \\
&= \frac{(x_0 - y_0)^2}{2} + \frac{(x_1 - y_1)^2}{2} \\
&= \frac{(x_0 - y_0)^2 + (x_1 - y_1)^2}{2} \\
&= \frac{D^2(\vec{x}, \vec{y})}{2}
\end{aligned}
$$

Thus,

$$D^2(\vec{s}, \vec{r}) = \frac{D^2(\vec{x}, \vec{y})}{2}$$

$$D(\vec{x}, \vec{y}) = 2^{\frac{1}{2}} D(\vec{s}, \vec{r})$$

■

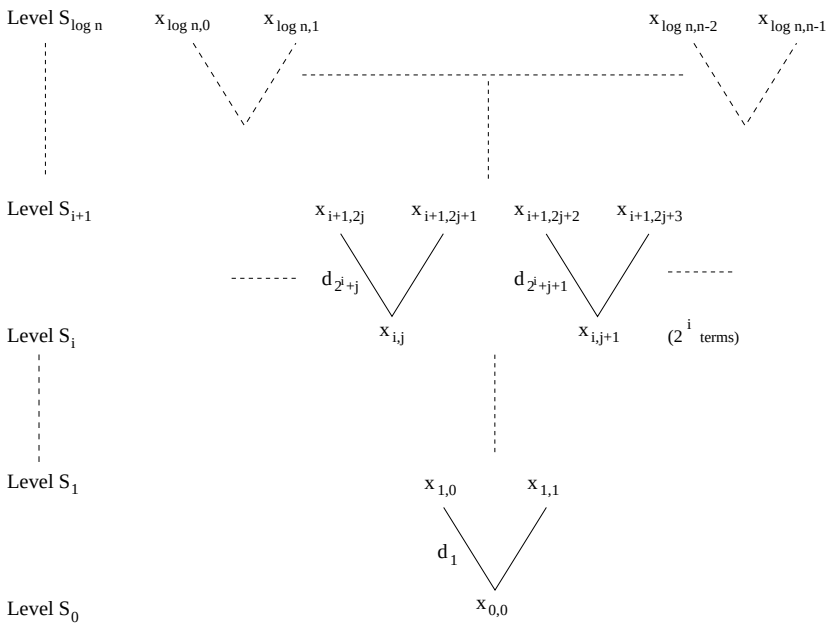

Figure 3.5: Hierarchy of Haar wavelet transform of sequence $\vec{x}$ of length n

Lemma 3 Given two sequences $\vec{x}$ and $\vec{y}$, and the Haar transforms of $\vec{x}$, $\vec{y}$ are $\vec{s}$, $\vec{r}$ respectively. Lengths of $\vec{x}$, $\vec{y}$, $\vec{s}$, and $\vec{r}$ are all n ($n \geq 2$ and n is a power of 2).

$\vec{r}$ - $\vec{s}$ = $\{C, D_1, D_2, \ldots, D_{n-1}\}$. The Euclidean distance $D(\vec{x}, \vec{y}) = S_{\log_2 n}$ can be expressed in terms of $\{C, D_1, D_2, \ldots, D_{n-1}\}$ recursively by

$$\begin{aligned} S_{i+1} &= 2^{\frac{1}{2}} \times \{(S_i^2 + D_{2^i}^2 + D_{2^i+1}^2 + \cdots + D_{2^{i+1}-1}^2)\}^{\frac{1}{2}} \quad \text{for } 0 \leq i \leq \log_2 n - 1 \\ S_0 &= C \end{aligned} \tag{3.10}$$

Proof: In Figure 3.5, the original sequence $\vec{x}$ is represented at level $\log_2 n$. The values of $x_{i,j}$ and d_{2^i+j} are defined by

$$x_{i,j} = \frac{x_{i+1,2j} + x_{i+1,2j+1}}{2}$$

$$d_{2^i+j} = \frac{x_{i+1,2j} - x_{i+1,2j+1}}{2}$$

The Haar transform of $\vec{x}$, $H(\vec{x})$ is represented by $\{x_{0,0}$,d_1,d_2,...,d_{2^i+j},d_{2^i+j+1}, ...,$d_{n-1}\}$. A similar hierarchy exists for another sequence $\vec{y}$. Denote $C = x_{0,0} - y_{0,0}$ and $D_i = d_i$ of sequence $\vec{x} - d_i$ of sequence $\vec{y}$, where $1 \leq i \leq n-1$.

We can treat the elements at each horizontal level of the hierarchy to be a data sequence. Hence the sequence at level S_i contains data $\{x_{i,0}, x_{i,1}, ..., x_{i,2^i-1}\}$.

Let us define S_i to be

$$S_i = \left\{ \sum_{j=0}^{2^i-1} (x_{i,j} - y_{i,j})^2 \right\}^{\frac{1}{2}}$$

S_i can be seen as the Euclidean distance between the data sequences at level i in the hierarchies for $\vec{x}$ and $\vec{y}$. Also, $S_{\log_2 n}$ is the Euclidean distance between the given time series.

Next we prove the following statement:

$$\begin{aligned} S_{i+1} &= 2^{\frac{1}{2}} \times \{(S_i^2 + D_{2^i}^2 + D_{2^i+1}^2 + \cdots + D_{2^{i+1}-1}^2)\}^{\frac{1}{2}} \quad \text{for } 0 \leq i \leq \log_2 n - 1 \\ S_0 &= C \end{aligned} \tag{3.11}$$

The base case is shown true by Equation (3.9) of Lemma 2 when $i = 0$,

$$S_1 = 2^{\frac{1}{2}} \times \{(S_0^2 + D_1^2)\}^{\frac{1}{2}}$$

We next prove the case for $i = k > 0$. In order to do this, we first note that in the given hierarchy, for a pair of adjacent elements at a level > 0 of the form $\{x_{i+1,2j}, x_{i+1,2j+1}\}$, we have the following relation

$$\begin{aligned} & (x_{i+1,2j} - y_{i+1,2j})^2 + (x_{i+1,2j+1} - y_{i+1,2j+1})^2 \\ = \; & 2\left((x_{i,j} - y_{i,j})^2 + \left(d_{2^i+j} - d'_{2^i+j}\right)^2\right) \end{aligned} \tag{3.12}$$

where d'_{2^i+j} is the element in the hierarchy for $\vec{y}$ corresponding to d_{2^i+j}.

This can be shown by repeating the proof in Lemma 2, replacing $\vec{x}$ by $\{x_{i+1,2j}, x_{i+1,2j+1}\}$, $\vec{y}$ by $\{y_{i+1,2j}, y_{i+1,2j+1}\}$, $\vec{s}$ by $\{x_{i,j}, d_{2^i+j}\}$, and $\vec{r}$ by $\{y_{i,j}, d'_{2^i+j}\}$. Note that $\left(d_{2^i+j} - d'_{2^i+j}\right)^2 = D_{2^i+j}^2$.

For $i = k$,

$$\begin{aligned} S_{k+1} &= \left\{\sum_{j=0}^{2^{k+1}-1} (x_{k+1,j} - y_{k+1,j})^2\right\}^{\frac{1}{2}} \\ &= \{(x_{k+1,0} - y_{k+1,0})^2 + (x_{k+1,1} - y_{k+1,1})^2 + \cdots + \\ &\quad (x_{k+1,2^{k+1}-1} - y_{k+1,2^{k+1}-1})^2\}^{\frac{1}{2}} \end{aligned}$$

By Equation (3.12), we have

$$\begin{aligned} S_{k+1} = \; & \left\{2\left[(x_{k,0} - y_{k,0})^2 + D_{2^k}^2\right] + 2\left[(x_{k,1} - y_{k,1})^2 + D_{2^k+1}^2\right] + \cdots \right. \\ & \left. + 2\left[(x_{k,2^k-1} - y_{k,2^k-1})^2 + D_{2^k+2^k-1}^2\right]\right\}^{\frac{1}{2}} \end{aligned}$$

$$\begin{aligned} &= \left\{2\left[(x_{k,0}-y_{k,0})^2+(x_{k,1}-y_{k,1})^2+\cdots+(x_{k,2^k-1}-y_{k,2^k-1})^2\right]\right. \\ &\quad \left.+2\left[D_{2^k}^2+D_{2^k+1}^2+\cdots+D_{2^k+2^k-1}^2\right]\right\}^{\frac{1}{2}} \end{aligned}$$

Finally by the definition of S_k,

$$S_{k+1} = 2^{\frac{1}{2}} \times \{(S_k^2 + D_{2^k}^2 + D_{2^k+1}^2 + \cdots + D_{2^{k+1}-1}^2)\}^{\frac{1}{2}}$$

which completes the proof. ∎

Example To illustrate, consider two sequences $\vec{x}$ = {1,4,5,6,3,2,4,5} and $\vec{y}$ = {2,5,4,3,2,5,6,8} with Euclidean distance

$$D(\vec{x},\vec{y}) = (1^2+1^2+1^2+3^2+1^2+3^2+2^2+3^2)^{\frac{1}{2}} = 35^{\frac{1}{2}}$$

Their Haar transform are found to be

$$H(\vec{x}) = \vec{s} = \{3.75, 0.25, -1.5, -1.0, -1.5, -0.5, 0.5, -0.5\}$$

$$H(\vec{y}) = \vec{r} = \{4.375, -0.875, 0.0, -1.75, -1.5, 0.5, -1.5, -1.0\}$$

Moreover,

$$\vec{r}-\vec{s} = \{C, D_1, D_2, \ldots, D_7\} = \{0.625, -1.125, 1.5, -0.75, 0.0, 1.0, -2.0, -0.5\}$$

From Equation (3.10),

$$\begin{aligned} D(\vec{x},\vec{y}) &= \{((((((0.625^2+1.125^2)\times 2)+1.5^2+0.75^2)\times 2) \\ &\quad +0.0^2+1.0^2+2.0^2+0.5^2)\times 2)\}^{\frac{1}{2}} \\ &= 35^{\frac{1}{2}} \end{aligned}$$

which shows its correctness. ∎

The expression of the Euclidean distance between time sequences in terms of their Haar coefficients is not sufficient for proper use in multi-dimensional

index trees until Euclidean distance preserves in both Haar and time domains, as for DFT in Equation (2.7). This can be achieved by a normalization step which replaces the scaling factor in Equation (3.8) from 1/2 to $1/2^{\frac{1}{2}}$ in the Haar transformation. After the normalization step, Euclidean distance between sequences in Haar domain will be equivalent to $S_{\log_2 n}$ in Equation (3.10). The preservation of Euclidean distance of Haar transform ensures the completeness of feature extraction as in DFT.

If only the first h_c dimensions ($1 \leq h_c \leq n$) of Haar transform are used in calculation of Euclidean distance in Equation (3.10), then we should replace 0's in the Haar transformed sequences. This replacement starts from h_c+1 th to n th coefficients in the transformed sequences.

Lemma 4 If the first h_c ($1 \leq h_c \leq n$) dimensions of Haar transform are used, no false dismissal will occur for range queries.

Proof: Considering the inequality in Definition 1 and Lemma 3

$$D(\vec{x}, \vec{y}) = S_{\log_2 n} \leq \epsilon \tag{3.13}$$

Using the first h_c dimensions as index, the value of D_i in Equation (3.10) will become zero for $i \geq h_c$. Thus the Euclidean distance between two sequences is $\leq S_{\log_2 n} \leq \epsilon$. This completes the proof. ∎

3.2 The Overall Strategy

In this section, we present the overall strategy of our time series search and propose our own method for nearest neighbor query. Before querying is performed, we shall do some pre-processing to extract the feature vectors with reduced dimensionality, and to build the index. After the index is built, content-based

search can be performed for two types of querying: range querying and n-nearest neighbors querying.

3.2.1 Pre-processing

1. Similarity Model Selection

   According to their applications users may choose to use either the simple Euclidean distance (Definition 1) or the v-shift similarity (Definition 2) as their similarity measurements. For Definition 1, Haar transform is applied to time series. For Definition 2, Haar transform is applied to time series, but the first Haar coefficient will not be used in indexing, as there is no need to match their average values anymore.

2. Index Construction

   Given a database of time series of varying lengths. We pre-process the time series as follows. We obtain the ω-point Haar transform by applying Equation (3.8) with normalized factor, to each subsequences with a sliding window of size ω for each sequence in the database.

   An index structure such as an R-Tree is built, using the first h_c [3] Haar coefficients where h_c is an optimal value found by experiments based on the number of page accesses. This is because of a trade off between post-processing cost and index dimension.

3.2.2 Range Query

After we have built the index, we can carry out range query or nearest neighbor query evaluation. For range queries, two steps are involved:

[3] Using Definition 2, one dimension can be saved in the index tree.

1. Similar sequences with distance $\leq \epsilon$ from the query are looked up in the index and returned.

2. A post-processing step is applied to these sequences to obtain the actual distance in time domain to remove all false alarms.

3.2.3 Nearest Neighbor Query

For nearest neighbor query, we propose a two-phase evaluation as follows. A viewgraph is shown in Figure 3.6.

- *Phase 1*

  In the first phase, n nearest neighbors of query $\vec{q}$ are found in the R-Tree index using the algorithm in [39]. The Euclidean distances D in time domain (full dimension) are computed between the query sequence and all n nearest neighbors obtained which are $D(\vec{q}, \vec{nn}_i^1)$, where $\vec{nn}_i^1$ denotes the nearest neighbor i $(1 \leq i \leq n)$, with $\vec{nn}_n^1$ farthest from the query $\vec{q}$. Note that the nearest neighbors found in current phase are not the final answer to the query, since the number of dimensions of sequences is reduced in the index tree. We just aim at acquiring a range that will be employed in the epsilon search in next phase.

- *Phase 2*

  A range query evaluation is then performed on the same index by setting $\epsilon = D(\vec{q}, \vec{nn}_n^1)$ initially. During the search, we keep a list of n nearest sequences $\vec{nn}_i^2$ found so far and their Euclidean distances in time domain (full dimension) $D(\vec{q}, \vec{nn}_i^2)$ with query $\vec{q}$ $(1 \leq i \leq n)$. The post-processing step mentioned in Section 3.2.2 is avoided since the Euclidean distances are found already in time domain during the search.

During the search we keep updating [4] the value of ϵ by $D(\vec{q}, \vec{nn_n^2})$ which is the distance of the current farthest neighbor.

The n nearest neighbors stored in the list are returned as answers when the range query evaluation is finished. The distance of the farthest nearest neighbor with query $\vec{q}$ is $D(\vec{q}, \vec{nn_n^{ans}})$.

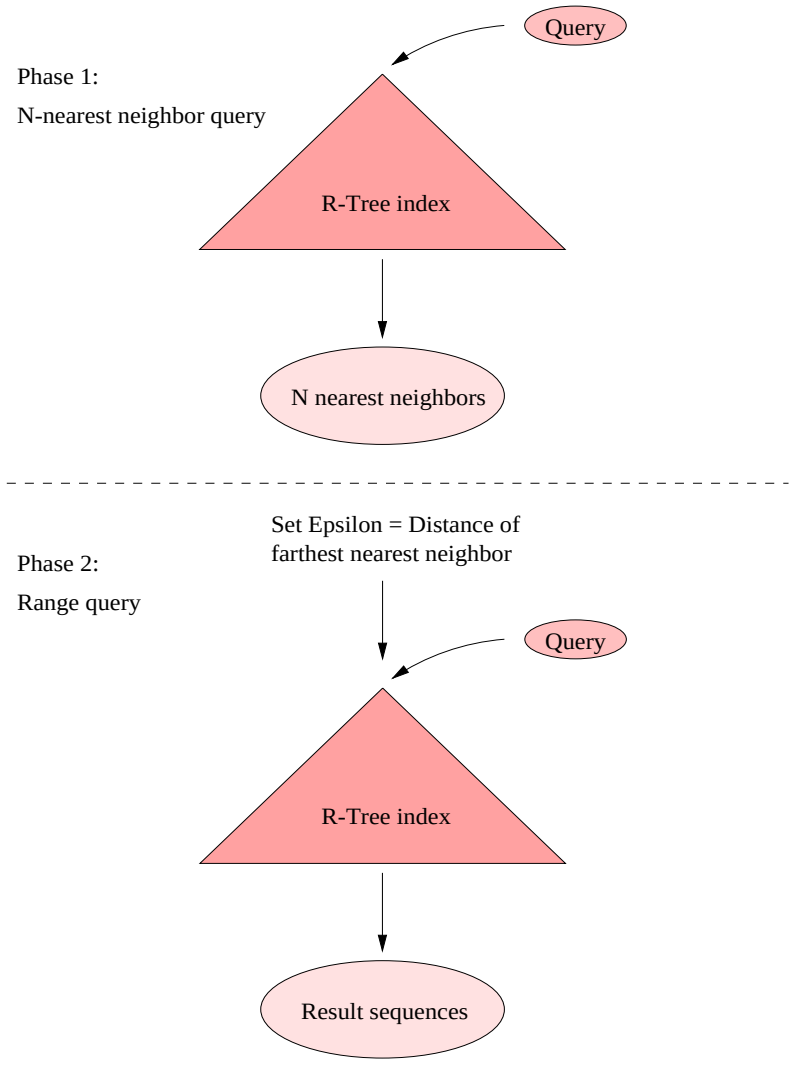

Figure 3.6: Two-phase nearest neighbor query

The correctness of the above algorithm can be shown by considering two cases, which are shown in Figure 3.7. For the first case (upper diagram), assume the

[4] The updating process begins only when the list storing the nearest neighbors has been filled up already.

n nearest neighbors in the final answer all appear in the results in Phase 1, $\vec{nn}_i^1 = \vec{nn}_j^{ans}$, where $1 \leq i, j \leq n$ and i need not be equal to j. Obviously, $D(\vec{q}, \vec{nn}_n^1) = D(\vec{q}, \vec{nn}_n^{ans})$. In the second case (lower diagram), assume some or no nearest neighbor obtained in the final answer appears in the results in Phase 1, $\vec{nn}_i^1 \neq \vec{nn}_j^{ans}$, where $1 \leq i, j \leq n$ for some i and j. Thus, $D(\vec{q}, \vec{nn}_n^1) > D(\vec{q}, \vec{nn}_n^{ans})$.

Therefore, combining the two cases, $D(\vec{q}, \vec{nn}_n^1) \geq D(\vec{q}, \vec{nn}_n^{ans})$ and by Lemma 4 there are only false alarms produced in the range query of Phase 2 since the value of ϵ upper bounds the distance of the farthest neighbor $\vec{nn}_n^{ans}$.

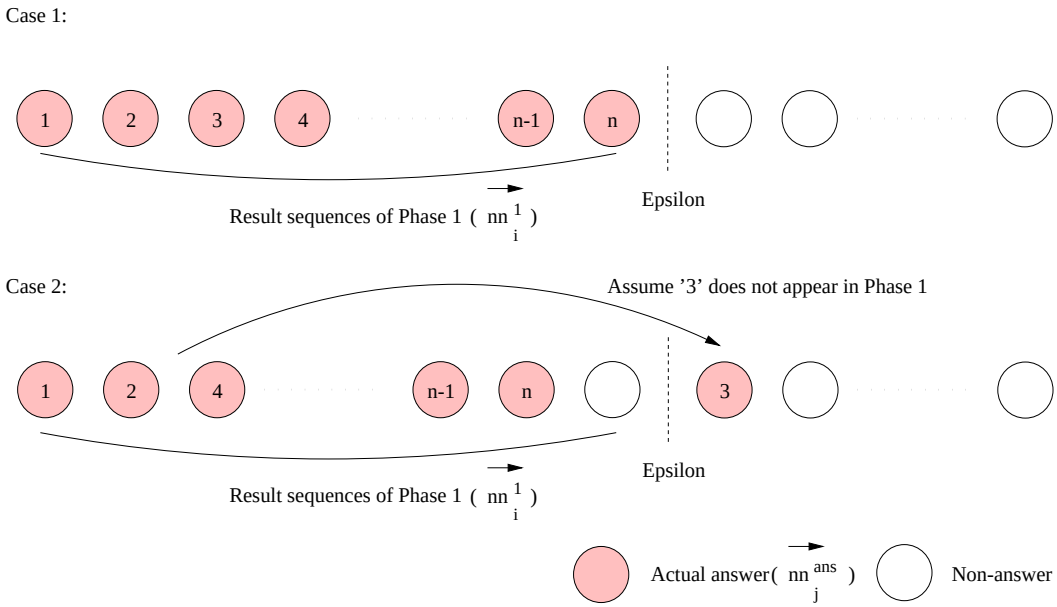

Figure 3.7: Epsilon used in the second phase ensures no false dismissal

The effectiveness of this n-nearest neighbor search algorithm arises from the value of $D(\vec{q}, \vec{nn}_n^1)$ found in Phase 1 which provides a sufficient small query range to prune away a large amount of candidates in Phase 2. No false dismissal will occur in Phase 2 as $D(\vec{q}, \vec{nn}_n^1)$ gives the upper bound distance for $D(\vec{q}, \vec{nn}_n^{ans})$ which is the farthest n nearest neighbor in the final answer.

The extra step introduced in Phase 2 to update ϵ can enhance the performance by pruning more non-qualified MBRs during the traversal of R-Tree.

3.3 Performance Evaluation

Experiments using real stock data and synthetic random walk data have been carried out. All experiments are conducted on a Sun UltraSPARC-1 workstation with 686MBytes of main memory. Page size is set to 1024 bytes. A branching factor of 20 is chosen for the R-Tree so that the index tree nodes can be fitted within one disk page. We have pointed out earlier that pre-processing time for Haar wavelet is much less than that for DFT. Here we shall compare the querying performance.

3.3.1 Stock Data

Real data are extracted from different equities of Hong Kong stock market from 12/7/90 to 7/11/96. The data have been collected daily over the time period. Totally 10k feature vectors are extracted by a sliding window of size $\omega = 512$ and inserted into an R-Tree.

Both range and nearest neighbor queries are examined and the results are shown from Figure 3.8 to Figure 3.15. Random queries are applied with varying epsilons ϵ, which range from 0.5% to 5% of the database size. The number of nearest neighbors for nearest neighbor query is between 20 and 40. All results are obtained from the average of 100 trials. In each figure, transformations using Definition 1 as similarity model are denoted by their abbreviations, while transformations using Definition 2 are denoted by (V-shift) in addition. For instance, Haar transforms using Definition 1 and Definition 2 as similarity models are denoted as 'Haar' and 'Haar(V-shift)' respectively.

In Figure 3.8, precision against the first tenth indexed coefficients/dimensions of range query is investigated using Definition 1 (non-v-shift similarity model).

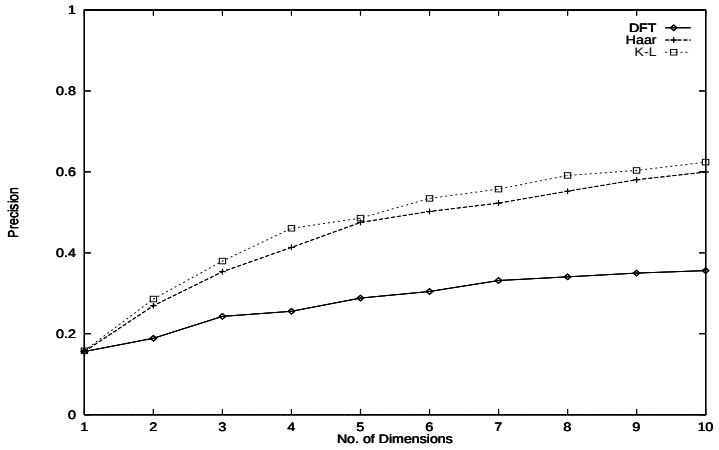

Figure 3.8: Precision of range query (Non-v-shift)

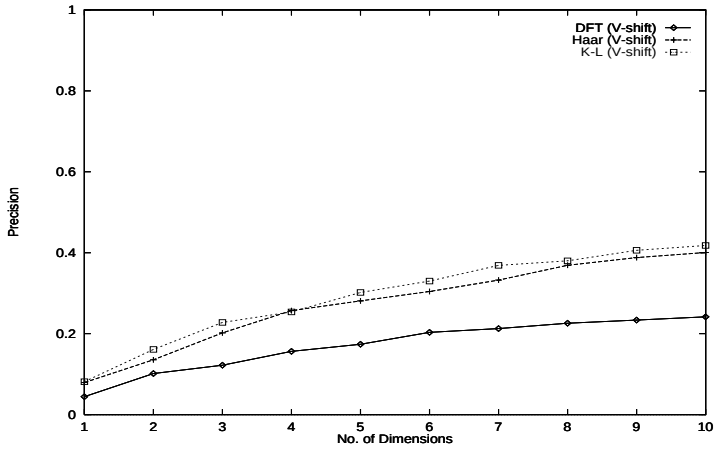

Figure 3.9: Precision of range query (V-shift)

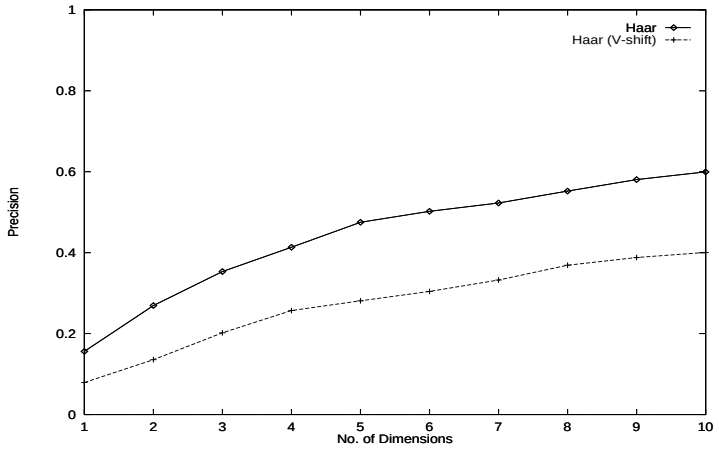

Figure 3.10: Precision of range query (Haar)

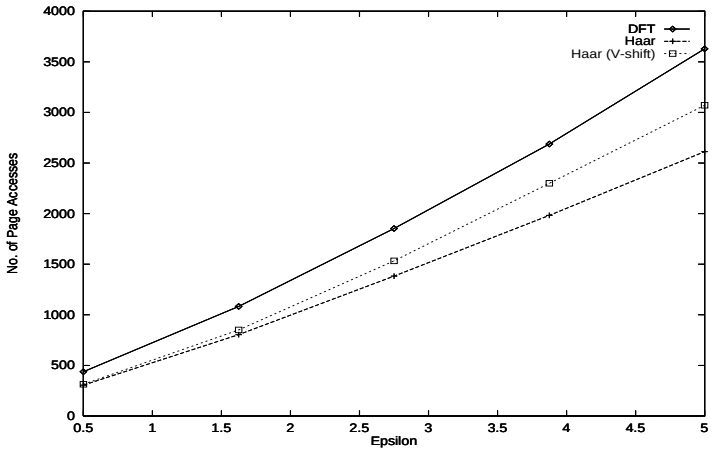

Figure 3.11: Page accesses of range query

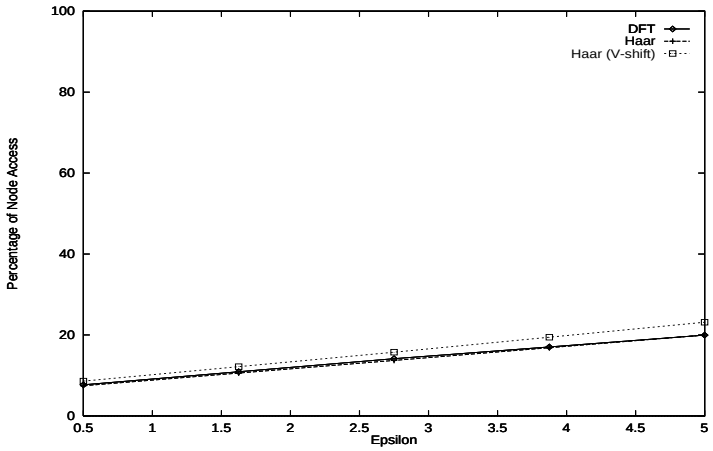

Figure 3.12: Node accesses of R-Tree for range query

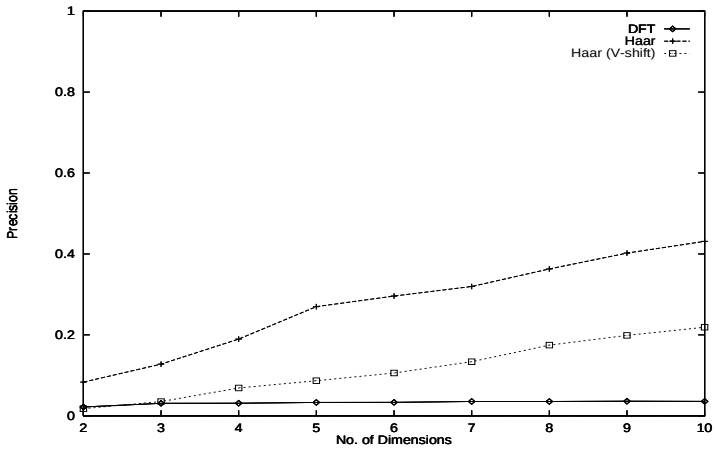

Figure 3.13: Precision of nearest neighbor query

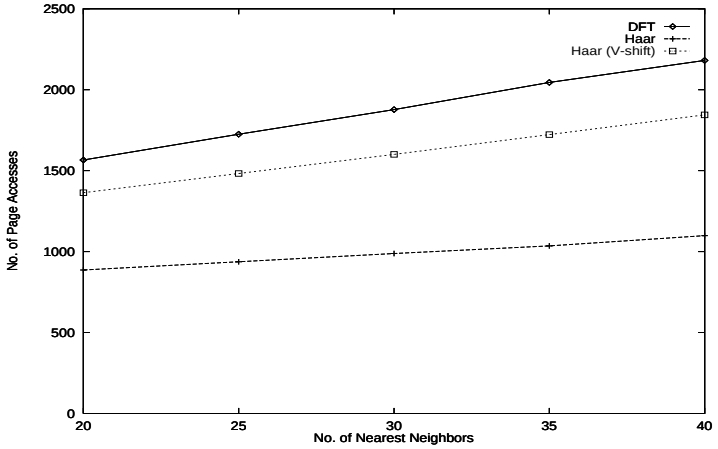

Figure 3.14: Page accesses of nearest neighbor query

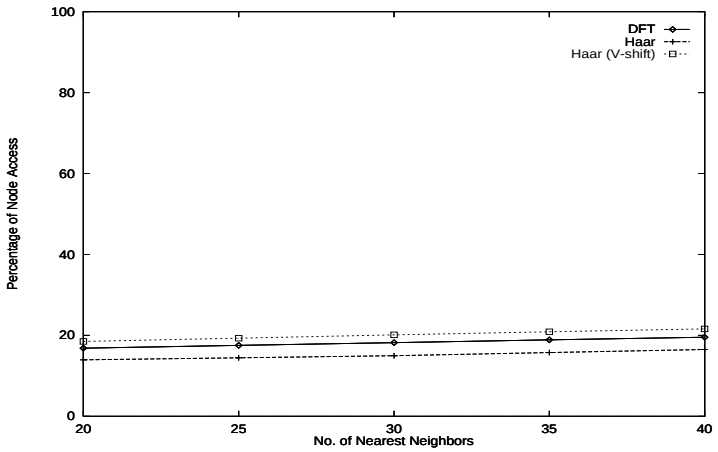

Figure 3.15: Node accesses of R-Tree for nearest neighbor query

It is defined as

$$\text{Precision} = \frac{S_{time}}{S_{transform}} \tag{3.14}$$

where S_{time} refers to the number of time sequences qualified in time domain while $S_{transform}$ is the number of time sequences qualified in the transformed domain. As we can observe, K-L transform gives the best precision at each dimension. On the other hand, the precision attained by Haar transform is close to the best and it outperforms DFT significantly at all except the first dimension. The enhancement in precision of Haar transform over DFT increases with the number of dimensions.

Moreover, we evaluate the precision of the three transformations using Definition 2 (v-shift similarity model). For Haar transform, we can achieve v-shift similarity matching by removing the first Haar coefficient prior to indexing, as the first coefficient stores the average value of the time series. Unfortunately, there is no evidence or result at present showing that we can apply this technique to DFT or K-L transform. Therefore, we achieve v-shift similarities for DFT and K-L transform in a static way by normalizing/shifting the time series in database, resulting in zero average value of each sequence.

The precision of the three transformations using v-shift model is shown in Figure 3.9. We observe that DFT(V-shift), Haar(V-shift), and K-L(V-shift) report a loss in precision when compared with their non-v-shift counterparts. The main reason is that the time series of financial data consist of a sequence of time values fluctuating around a relative constant level, which is the average value of a time sequence. This average value is very effective in discriminating time series in the sense that every sequence distributes farther away in the x-axis. As a result, its removal will cause a sudden drop in precision. Another observation is that the gain of precision upon addition of extra coefficients diminishes after the removal of average value. K-L using two coefficients in indexing gains 12% more precision, contrasting with that using one coefficient. However, only 9% more can

be gained for K-L(V-shift). With reference to the same figure, despite the loss in precision in v-shift model, K-L(V-shift) still gives the highest precision, and the precision of Haar(V-shift) is close to that of K-L(V-shift). The performance gap between DFT and Haar in Figure 3.8 still exists for v-shift model in this case.

For clarity, the precision of Haar and Haar(V-shift) is shown in Figure 3.10. Obviously, the precision of the non-v-shift model outperforms the v-shift model by 20% at most. The large difference is attributed to the removal of the first Haar coefficient to achieve v-shift similarity, which poses a loss of discrimination power addressed in previous paragraph. From another point of view, precision is traded for a better similarity model.

As most of the page accesses [5] of a query are devoted to remove false alarms and only a small proportion arises from index accesses, the precision is crucial to the overall performances of query evaluation. This agrees with the result depicted in Figure 3.11, where the page accesses of the best dimensions of DFT, Haar and Haar(V-shift) are shown. Page accesses increase linearly with ϵ. Haar has the minimum page accesses while DFT performs the worst. Page accesses of Haar(V-shift) model have been traded for better similarity model. Even so, it outperforms DFT. The page accesses of K-L transform are not shown. K-L transform is static-based as opposed to Haar transform and DFT, which are dynamic transformations. Therefore, it is more appropriate to compare only the performances between Haar transform and DFT. Nevertheless, the evaluation of precision in Figure 3.8 and Figure 3.9 give us some idea for the close performance of Haar and K-L transforms.

In Figure 3.12, the percentage of node access of R-Tree against ϵ is shown. All results follow a linear trend and have approximately the same value. Therefore,

[5] Performance is measured in terms of page access due to I/O time domination over computation time in database applications. Page accesses = non-leaf node accesses + leaf node accesses + post processing page accesses

the page accesses involved in index traversal are the same. The difference in performances is dominated by the precision of transformations. The percentage of node access for Haar(V-shift) is slightly higher than that of DFT and Haar because Haar(V-shift) needs more dimensions to attain sufficient precision in building the R-Tree. The best dimension of DFT (dimension 5) is smaller than Haar (dimension 7) and Haar(V-shift) (dimension 10) as there is no significant gain in precision with additional dimensions.

Results of nearest neighbor query are shown from Figure 3.13 to Figure 3.15. Figure 3.13 shows the precision of nearest neighbor query of DFT, Haar, and Haar(V-shift). In nearest neighbor search, the result is similar to that of range search. Haar attains the highest precision among the three. As expected, Haar outperforms Haar(V-shift) in the nearest neighbor query according to the same argument. There is no observable improvement with additional coefficients for DFT.

The trends for page access [6] in Figure 3.14 are consistent with those in range query in Figure 3.11, Haar and Haar(V-shift) still outperform DFT. The node accesses in Figure 3.15 are higher than that of Figure 3.12 since the number of nodes that are accessed must be relatively high for nearest neighbor than range query.

3.3.2 Synthetic Random Walk Data

Since many real data like stock movements and exchange rates can be modeled successfully by random walks [16], we study here the performance of our proposed technique for random walk data. Synthetic random walk data consisting of 30k time sequences are generated. As we want to show the effectiveness of our approach for different sequence lengths, we set $\omega = 1024$. The same set of

[6] Page accesses = non-leaf node accesses + leave node accesses

experiments as for the real data are performed and the results are shown from Figure 3.16 to Figure 3.23.

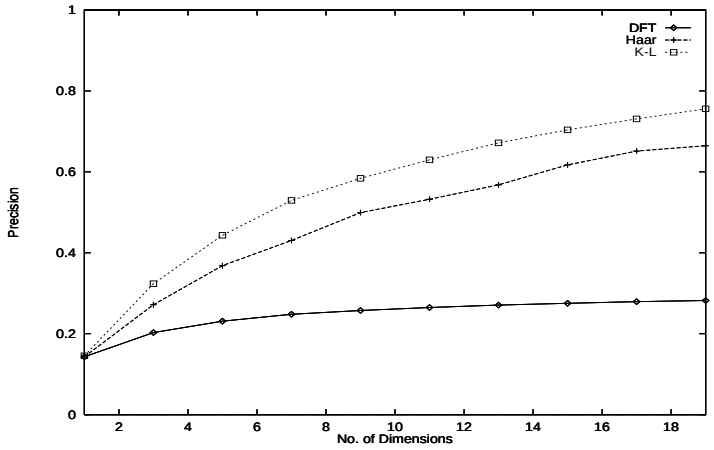

Figure 3.16: Precision of range query (Non-v-shift)

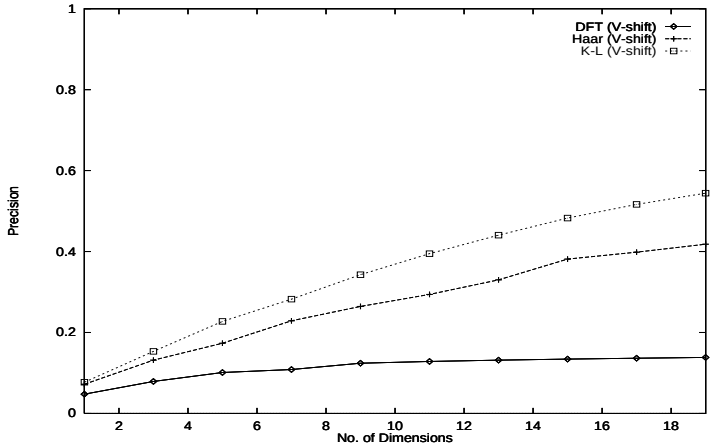

Figure 3.17: Precision of range query (V-shift)

The precision of non-v-shift model is shown in Figure 3.16. As the value of ω is doubled, more dimensions have to be used to attain sufficient precision. Therefore, we show the first twentieth dimensions. Similarly, the precision of Haar is near optimal while DFT flats out starting at dimension 8. The difference in performances among various transformations enlarges for longer time series. The precision of v-shift model is depicted in Figure 3.17. The precision drops of all transformations are consistent with the experiment using real dataset, which

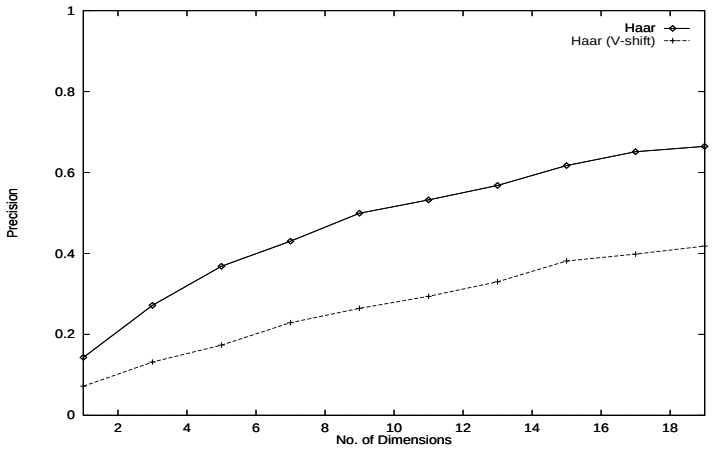

Figure 3.18: Precision of range query (Haar)

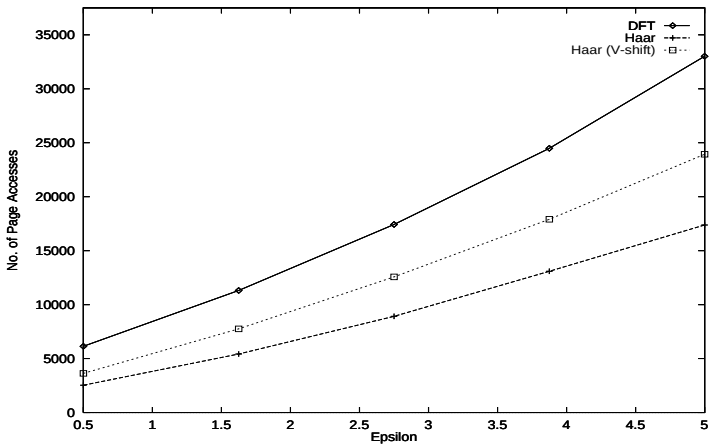

Figure 3.19: Page accesses of range query

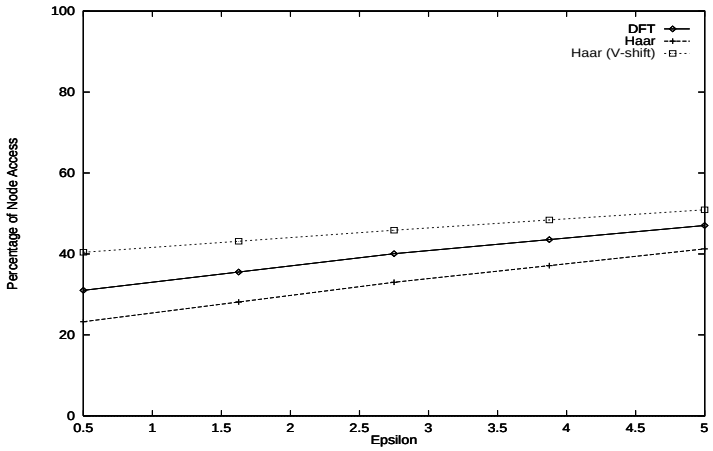

Figure 3.20: Node accesses of R-Tree for range query

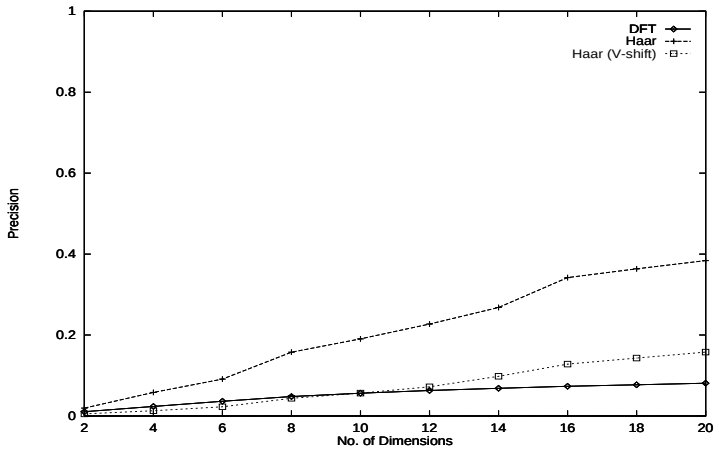

Figure 3.21: Precision of nearest neighbor query

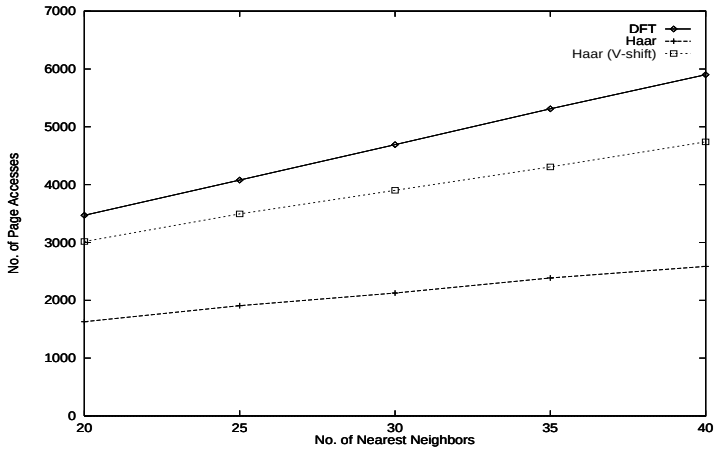

Figure 3.22: Page accesses of nearest neighbor query

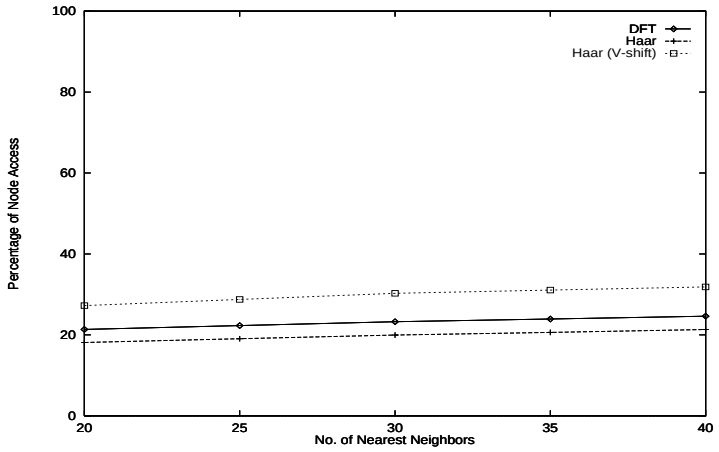

Figure 3.23: Node accesses of R-Tree for nearest neighbor query

is caused by the deterioration of discriminating power without the average value. In Figure 3.18, Haar(V-shift) has 20% loss of precision with respect to Haar which agrees with Figure 3.10.

Page access of range query is shown in Figure 3.19, Haar outperforms both Haar(V-shift) and DFT, with DFT performs the worst. Haar scales the best with epsilon. The differences in performances of the three methods enlarge since we use a larger dataset size and longer time sequences of synthetic data. Figure 3.20 shows the percentage of node access with Haar(V-shift) being the worst. Although Haar(V-shift) accesses at most 10% more index nodes than DFT, it still outperforms DFT in terms of page accesses. This confirms our expectations that the number of page accesses associated with index node is relatively small.

The precision of nearest neighbor query of DFT, Haar, and Haar(V-shift) is shown in Figure 3.21. As in Figure 3.13, Haar outperforms the others. An exception is that the precision of DFT outperforms Haar(V-shift) a tiny amount in the first eight dimensions. However, this will not affect the overall performance of Haar(V-shift) in terms of number of page accesses, as the optimal number of dimension is found to be greater than 10. Again, both Haar and Haar(V-shift) outperform DFT in nearest neighbor query which is shown in Figure 3.22, agreeing with the results in range query. Haar outperforms DFT significantly in particular. On the other hand, Figure 3.23 depicts a similar result as in Figure 3.15.

Therefore, our approach using Haar or Haar(V-shift) for time series matching justifies for real and synthetic datasets in both range and nearest neighbor queries. Moreover, the two-phase nearest neighbor query is shown to be effective by considering the low page access associated with both datasets.

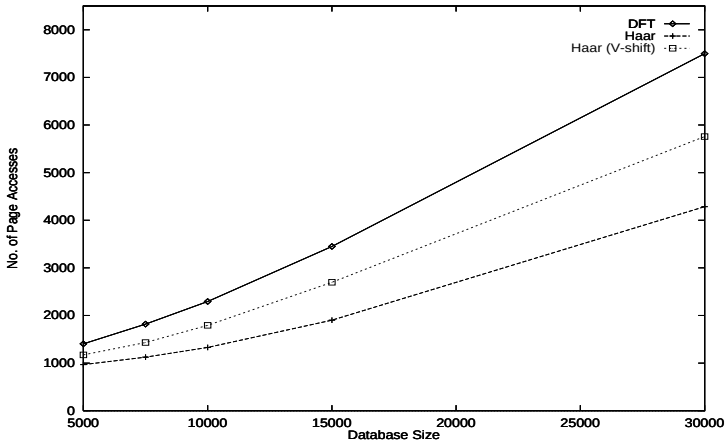

Figure 3.24: Database size (Range query)

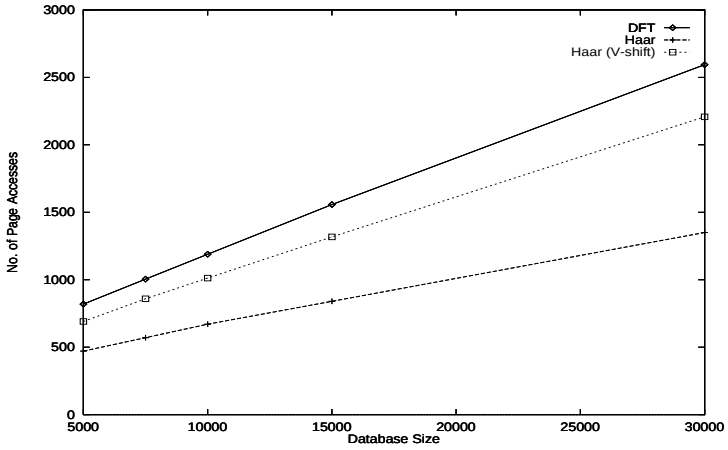

Figure 3.25: Database size (Nearest neighbor query)

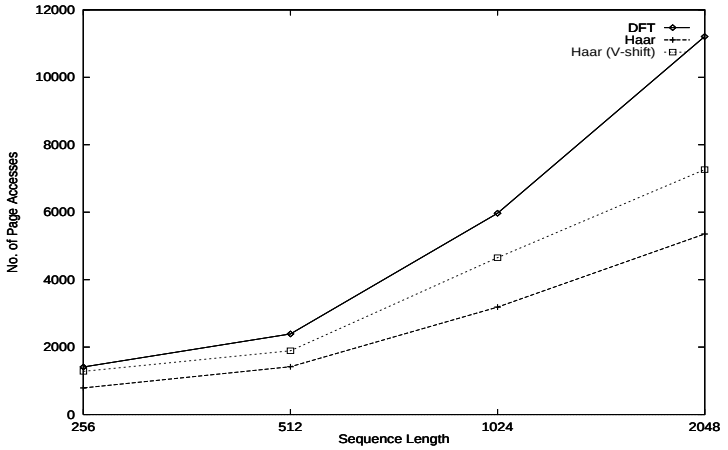

Figure 3.26: Sequence length (Range query)

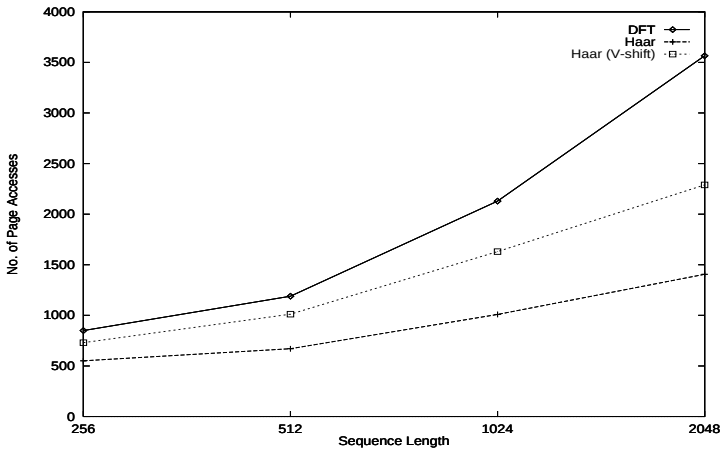

Figure 3.27: Sequence length (Nearest neighbor query)

3.3.3 Scalability Test

We study the scalability of our method by varying the size (Figure 3.24 and Figure 3.25) or the length (Figure 3.26 and Figure 3.27) of synthetic time series database. For scalability in database size, different time sequence databases of size ranges from 5k to 30k are generated as described in Section 3.3.2. Length of sequence is fixed to 512. For scalability in sequence length, databases with sequence of length 256, 512, 1024, and 2048 are generated. Size of each database is fixed to 10k sequences.

Figure 3.24 and Figure 3.25 show the scalability of both range and nearest neighbor queries. In both cases, Haar and Haar(V-shift) have a better scaling with database size increase than DFT. The difference in the amount of page accesses is tremendous and significant for large database size. A similar situation exists for database with long sequences which is shown in Figure 3.26 and Figure 3.27. The difference in page accesses is enormous for sequence of length 2048. As revealed from previous experiments, a considerable portion of page accesses is devoted to the post-processing step. The poorer precision of DFT creates more works in the post-processing step and this affects the overall performance, especially in terms of the amount of disk accesses for large databases with long

sequences.

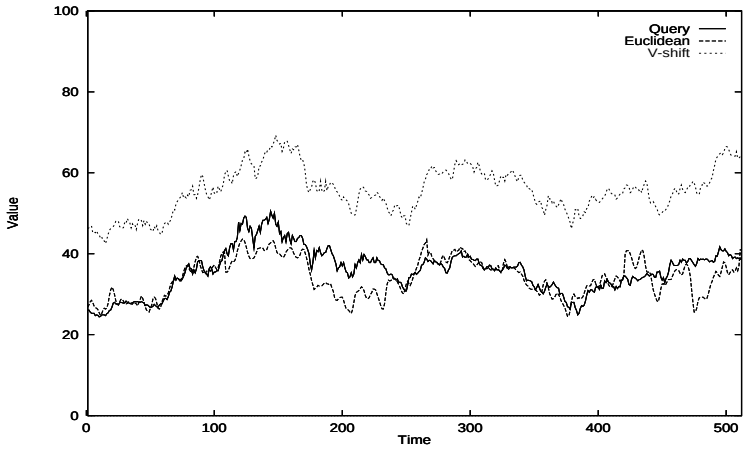

Figure 3.28: Visualization of query and result time sequences

Figure 3.28 shows the best time sequences matching a query using both non-v-shift and v-shift models. The reader may judge that the upper sequence returned by v-shift model has a more similar shape to the query, while the time sequence returned by non-v-shift model fails to follow a consistent shape with the query at some regions. This phenomenon can be explained by their differences in the similarity model definition. In the extreme case, an identical shape sequence can not be returned with the simple non-v-shift model if its vertical offset with respect to the query is large.

3.3.4 Other Wavelets

There are many kinds of known wavelets, we have tried some other wavelets in our experiments. The precision of different wavelets is compared using both real (Figure 3.29) and synthetic (Figure 3.30) data. Daub4 corresponds to the Daubechies wavelets with 4 coefficients in wavelet filter while Coif6 corresponds to the Coiflet wavelets with 6 coefficients in wavelet filter. We observe that Haar

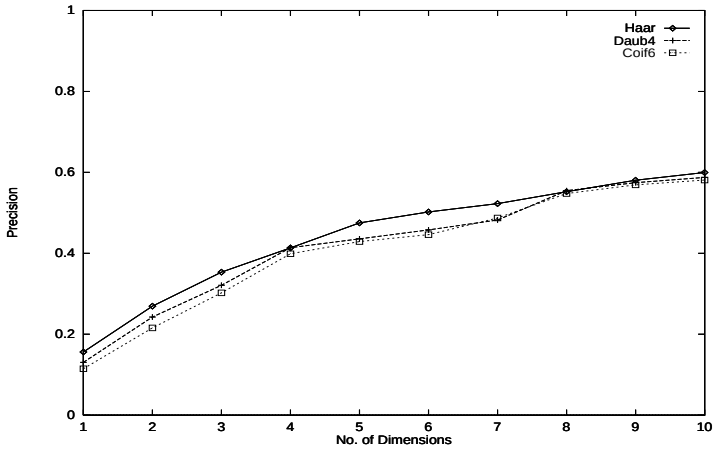

Figure 3.29: Precision of range query (Real data)

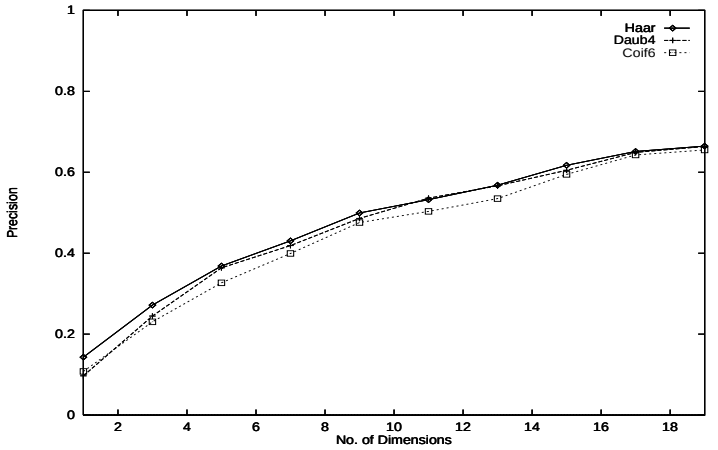

Figure 3.30: Precision of range query (Synthetic data)

wavelets performs better than the Daubechies and Coiflet wavelets. Moreover, it is computationally less expensive than the other wavelets.

We have discovered that not all the wavelets are suitable for dimensionality reduction for time series data. From our experiments, not all the wavelets are able to concentrate energy at the first few coefficients. Haar, Daub4, and Coif6 are the best wavelets we have found so far in their families. From experiments, we find that the other wavelets seem to also preserve Euclidean distances, however, so far we have a proof of this property only for the Haar wavelets. It will be interesting to see if we can apply different kinds of wavelets to different data series.

Chapter 4

Time Warping

Most of the time series similarity models are based on Euclidean distance between two time sequences. This linear matching process ignores the vertical (y-axis) and time (x-axis) shifts of sequences, which are indispensable to practical time series matching in reality. The problem of vertical shifts can be handled by the v-shift similarity model that we have proposed. On the other hand, time shifts of sequences can be coped with by means of time warping techniques [37, 13]. Time warping is widely used in speech and word recognition fields, in which human speech consists of varying durations and paces. The problem associated with sequence comparison for speech comes from the fact that different acoustic renditions, or tokens, of the same speech utterance (e.g., word, phrase, sentence) are seldom realized at the same speed (speaking rate) across the entire utterance. Thus, when comparing different tokens of the same utterance, speaking rate variations as well as duration variation should not contribute to the linguistic dissimilarity. Hence, there is a need to normalize speaking rate fluctuation in order for the utterance comparison to be meaningful before a recognition decision can be made.

Figure 4.1 shows two time series before time warping. When matched by time warping, the two sequences are aligned according to their peaks and valleys

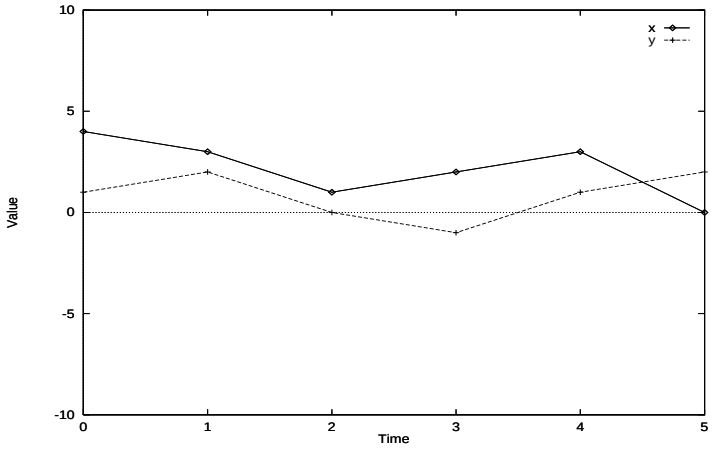

Figure 4.1: Time series $\vec{x}$ and $\vec{y}$ before time warping

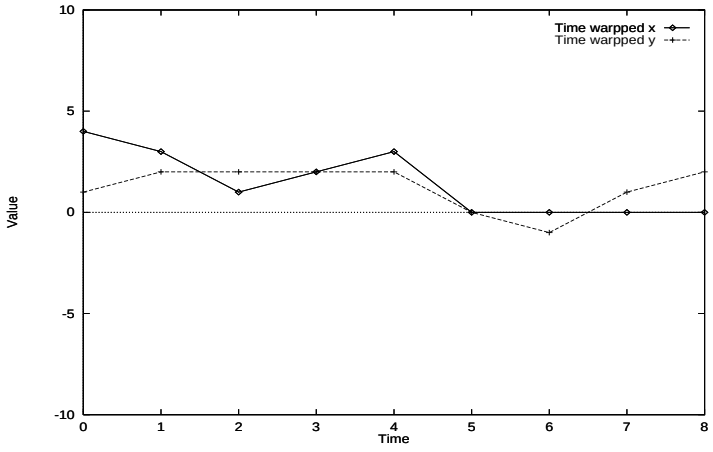

Figure 4.2: Time series $\vec{x}$ and $\vec{y}$ after time warping

Algorithm 4.1 TimeWarpDistance($\vec{x}$,$\vec{y}$)

```
1   x_len = ||x⃗||;
2   y_len = ||y⃗||;
3   C_matrix[0][0] = 0.0;
4   for (1 ≤ i ≤ x_len)
5       C_matrix[i][0] = ∞;
6   for (1 ≤ j ≤ y_len)
7       C_matrix[0][j] = ∞;
8   for (1 ≤ i ≤ x_len)
9       for (1 ≤ j ≤ y_len)
10          C_matrix[i][j] =
11          {D²(x_i, y_j) + min²(C_matrix[i−1][j], C_matrix[i][j−1], C_matrix[i−1][j−1])}^(1/2);
12  return C_matrix[x_len][y_len];
```

Figure 4.3: Algorithm for finding time warping distance

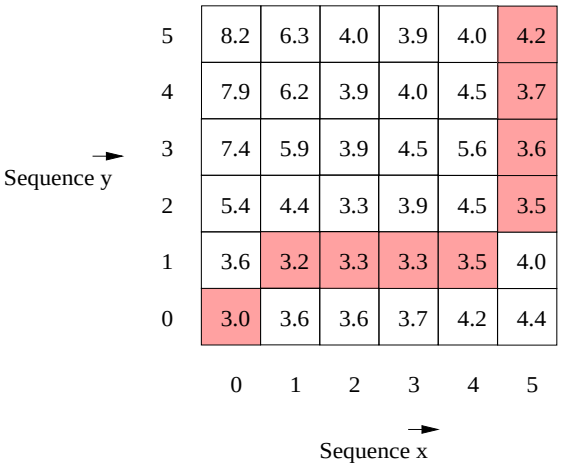

Figure 4.4: Cumulative distance matrix for template and data time series

by extending time values in order to minimize the distance between the series, this is shown in Figure 4.2. This is very different from simple Euclidean distance matching, where values of two sequences are matched based on the same time axis. Therefore, time warping is capable of extracting time series with similar shapes in different phases. Additional constraints [41] may also be applied to restrict the degree of freedom of the warping process for different applications.

An algorithm for time warping by dynamic programming method [47] can be found in Figure 4.3. The general principle is to find the shortest cumulative distance for each pair of time values between sequences $\vec{x}$ and $\vec{y}$, starting from the first pair (x_0, y_0), till the last pair $(x_{x_{len}}, y_{y_{len}})$. The time warping distance is actually the shortest cumulative distance of the whole sequences.

Example To illustrate, consider two sequences $\vec{x}$ = {4,3,1,2,3,0} and $\vec{y}$ = {1,2,0,-1,1,2} from Figure 4.1, the corresponding cumulative distance matrix is found using algorithm in Figure 4.3 and is shown in Figure 4.4. Each box corresponds to an entry in the cumulative distance matrix $C_{matrix}[i][j]$. Those pairs constituting the overall shortest cumulative distance are in grey, and the time warping distance is found to be $D_{timewarp}(\vec{x}, \vec{y}) = 4.2$ (upper right corner of the matrix). ■

Though time warping technique can accommodate time shifts of sequences, it is not as popular as Euclidean distance in time series matching. There are two limitations in using time warping distance. First, for length n sequences, the complexity of time warping distance function is $O(n^2)$ [1] as revealed from the distance matrix calculation, compared with $O(n)$ of Euclidean distance matching. This hinders the use of time warping distance for similarity searching in enormous time series databases where response time is a critical issue. Second, we cannot directly apply indexing techniques for time warping distance as in [23, 2] to speed up sequences retrieval. For multi-dimensional index trees like R-Tree,

[1] Strictly speaking, the complexity should be $O(\|\vec{x}\| \times \|\vec{y}\|)$

the distance function under consideration is assumed to be a *metric*, and time warping distance fails to fulfill this requirement. The definition of *metric space* is given as follows.

Definition 3 Given a nonempty set $\mathcal{X}$, a distance function or metric D on $\mathcal{X}$ is a function which assigns to each pair of points a non-negative real number satisfying the following for all x,y $\in \mathcal{X}$:

1. D(x,y) $\geq$ 0 and D(x,y) = 0 if and only if x = y;
2. D(x,y) = D(y,x);
3. For all x,y,z $\in \mathcal{X}$, D(x,y) $\leq$ D(x,z) + D(y,z), (triangle inequality).

The pair($\mathcal{X}$,D) is called a metric space. Different metrics defined on the same set can produce different metric spaces. ■

The most widely used distance function for similarity search in time series database is the Euclidean distance L_2 of the L_p metric family of distance function

$$D_p(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{4.1}$$

We can show that time warping distance violates the triangle inequality by the following example.

Example Given three time sequences $\vec{x}$ = {0,1,3}, $\vec{y}$={3,2,2}, and $\vec{z}$ = {2,3,2}. $D_{timewarp}(\vec{x}, \vec{y}) = 11 \geq D_{timewarp}(\vec{x}, \vec{z}) + D_{timewarp}(\vec{y}, \vec{z}) = 6 + 1 = 7$ ■

It is inappropriate to employ multi-dimensional index trees for direct indexing based on time warping distance, since it may give rise to false dismissals.

To deal with the problem of high time complexity, we propose approximation functions to time warping distance, which results in less computation by trading off tiny amount of accuracy. Before looking into our solution to the second problem of false dismissals, we would first elaborate on an index-based similarity search supporting time warping distance.

4.1 Similarity Search based on K-L Transform

An approach [47] that makes use of K-L transform and lower bound distance function to support matching with time warping is described in Section 2.4. K-L transformed sequences is inserted into an index tree (Fastmap index), and lower bound distance function filters false alarms in the post-processing step. The main drawback of this technique is that the search range of time warping distance $\epsilon_{timewarp}$, when used in extracting sequences in the Fastmap index, i.e $D(\vec{x}, \vec{y}) \leq \epsilon_{fastmap} = \epsilon_{timewarp}$, is not effective in finding a relatively small candidate set with little false dismissals [2]. Therefore, large amounts of sequences have to be checked with the lower bound distance function D_{lb} and then the time warping distance function $D_{timewarp}$ to obtain the answer. To demonstrate the ineffectiveness of using $\epsilon_{timewarp}$ as the search range for the Fastmap index, we have conducted experiments based on precision and recall (excluding the pruning step by D_{lb}).

Precision and recall are defined as follows.

$$\text{Precision} = \frac{S_{RetrievedAndQualified}}{S_{Retrieved}} \tag{4.2}$$

$$\text{Recall} = \frac{S_{RetrievedAndQualified}}{S_{Qualified}} \tag{4.3}$$

[2] In fact, the value of $\epsilon_{fastmap}$ is increased to $\epsilon^2_{timewarp}$ in [47] to reduce the number of false dismissals.

where $S_{Retrieved}$ is the number of sequences retrieved from Fastmap index, $S_{Qualified}$ is the number of sequences qualified, and $S_{RetrievedAndQualified}$ is the number of sequences retrieved and qualified.

Totally 50 random sequences are queried on a database of 5k synthetic random walk sequences of length 256. The value of $\epsilon_{timewarp}$ is fixed such that the number of qualified sequences in the result set is 2.75% of the database size. On the other hand, the value of $\epsilon_{fastmap}$ is varied such that 1 qualified sequence is extracted at the minimum range and all qualified sequences are extracted at the maximum range. Result is shown in Figure 4.5.

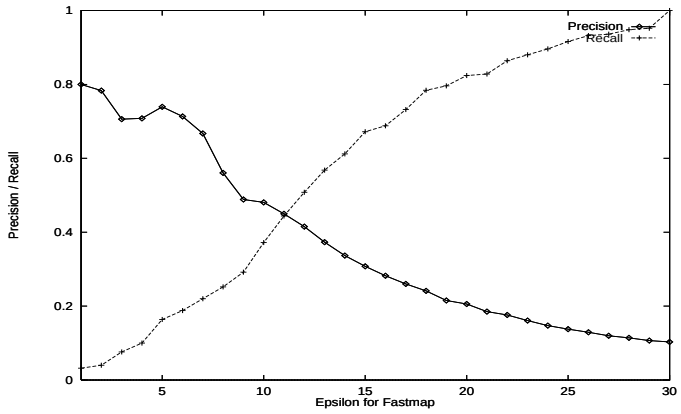

Figure 4.5: Precision and recall of range query on Fastmap index

It is observed that when we want to have higher recall, the value of $\epsilon_{fastmap}$ should be increased. However, precision drops rapidly with $\epsilon_{fastmap}$, which results in large amounts of false alarms. The consequence is that more processing time is devoted to matching of candidate sequences with D_{lb} and $D_{timewarp}$. As the complexity of $D_{timewarp}$ is $O(n^2)$, the performance drops drastically with lengthy time series, which is confirmed in [47].

Even worse, D_{lb} underestimates $D_{timewarp}$ to a great extent. For the same time series database we use, the fraction of distance estimated by D_{lb} is shown in

Figure 4.6. The pruning power of D_{lb} is low such that only 25% to 35% of time warping distance on the average can be estimated. Therefore large amounts of remaining sequences still should be checked with $D_{timewarp}$, which involves the most computations.

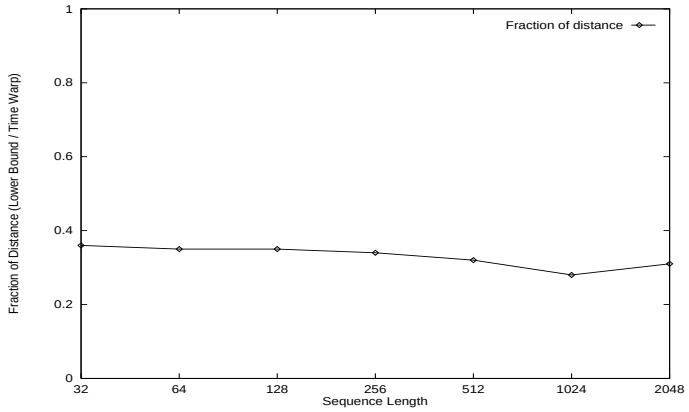

Figure 4.6: Fraction of distance estimated by D_{lb} (D_{lb} / $D_{timewarp}$)

We observe no simple solution to the second problem of false dismissals occurrence. Even for the K-L transform based index we have mentioned, the search range $\epsilon_{fastmap}$ should be enlarged with respect to $\epsilon_{timewarp}$ to avoid possible false dismissals. Rather than modifying Fastmap index to guarantee no false dismissals, we try to improve the overall performance of the similarity querying of Fastmap index, by replacing the lower bound distance function with our proposed approximation function as a more effective filter in the post-processing step, which is capable of pruning a large number of false alarms arising from large $\epsilon_{fastmap}$. Two approximation functions are suggested, which are *Low Resolution Time Warping* and *Adaptive Time Warping*. The former one can act as both an approximation and a filtering functions, while the latter one is solely for time warping distance approximation.

4.2 Low Resolution Time Warping

In order to reduce the time complexity of sequence matching in time warping distance, we propose to obtain a lower resolution version of the time sequences such that an approximation to the time warping distance can be found using an acceptable computation time. To achieve low resolution time warping, two steps are involved, resolution reduction and distance compensation.

4.2.1 Resolution Reduction of Sequences

To achieve different resolutions of sequences, we employ the technique in multi-resolution representation of Haar wavelets [3]. Upon application of Haar transform on time sequences, Haar coefficients can be obtained. Conversely, we may also reconstruct time sequences by applying an inverse Haar transform to Haar coefficients.

For a time sequence, its Haar transformation, or *decomposition* can be found by Equation (3.8) in Section 3.1.1. The inverse Haar transformation, or *reconstruction* goes in a similar manner, but actually reversing what we do in decomposition. It is shown in Equation (4.4),

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x'_0 \\ d^1_0 \\ x'_1 \\ d^1_1 \end{bmatrix} \tag{4.4}$$

with Haar coefficients $\vec{w} = [x'_0\ d^1_0\ x'_1\ d^1_1]^T$ as input and $\vec{x} = [x_0\ x_1\ x_2\ x_3]^T$ as

[3] Restriction is imposed on the length of time series. However, the problem of inconsistent sequence lengths was not addressed in [47] such that K-L transform can be applied properly.

output [4]. The number of iterations for recovering the original sequence in the reconstruction is exactly the same as needed in decomposition. Note that the inverse Haar transform matrix $\mathbf{H}'$ is equal to two times the Haar transform matrix $\mathbf{H}$, i.e. $\mathbf{H}' = 2\mathbf{H}$, $\mathbf{H} \times \mathbf{H}' = 2\mathbf{H} \times \mathbf{H} = \mathbf{I}$. Different resolutions of time series can be achieved by varying the number of iterations performed in Equation (4.4) of the reconstruction process. The more the iterations, the higher the resolution we can obtain.

Referring to Figure 4.7, we show different resolutions of an input sequence $\vec{z}$ = {3,1,0,2,-3,-4,1,2}. Decomposition and reconstruction correspond to downward and upward traversals of the tree respectively, with upper levels representing higher resolutions. By reconstruction of Haar coefficients of $\vec{z}$, we can obtain 4 different resolutions which are {0.25,0.25,0.25,0.25,0.25,0.25,0.25,0.25}, {1.5,1.5,1.5,1.5,-1.0,-1.0,-1.0,-1.0}, {2.0,2.0,1.0,1.0,-3.5,-3.5,1.5,1.5}, and $\vec{z}$ itself. Note that the lengths of sequences are preserved at different resolutions. The main drawback of this approach is the insufficiency of diversity of resolutions for a sequence. Even for a sequence of length 256, there are only 8 resolutions available. This is not very flexible and efficient for practical use in database index where diverse variations of resolutions are desired. Therefore, we allow the resolution reached by each branch to be different, i.e. the resolutions of sequence segments can vary. For instance, we can obtain a finer resolution of sequence {2.0,2.0,1.0,1.0,-3.5,-3.5,1.5,1.5} by expanding the last two values {1.5,1.5} to {1.0,2.0}, which becomes {2.0,2.0,1.0,1.0,-3.5,-3.5,1.0,2.0}. A systematic way to achieve this variety of resolutions is described below.

Instead of running different number of iterations of Equation (4.4), we obtain different resolutions of $\vec{z}$ by first truncating Haar coefficients $H(\vec{z})$, then performing a full reconstruction (a full iteration of matrix multiplication) of Equation (4.4). The number of coefficients truncated determines the resolution of sequence

[4] Normalization can be achieved by adding a scaling factor of $1/2^{\frac{1}{2}}$ in Equation (4.4).

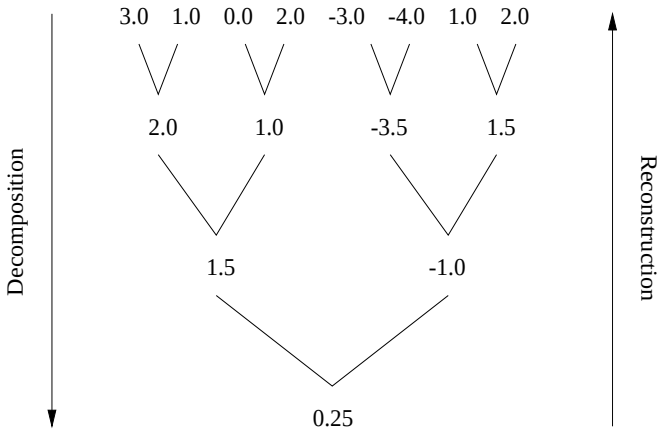

Figure 4.7: Resolution reduction by variations of iterations

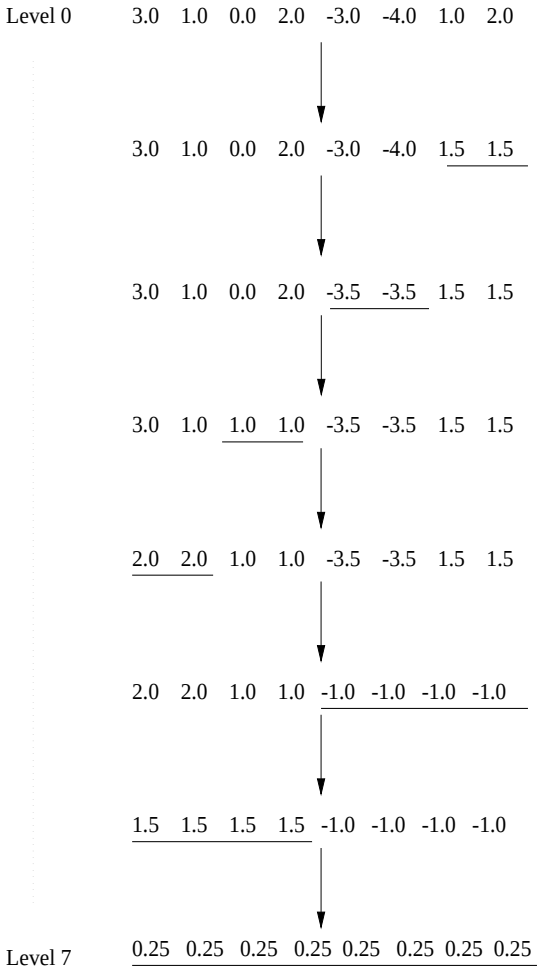

Figure 4.8: Resolution reduction by truncation and full reconstruction of Haar coefficients

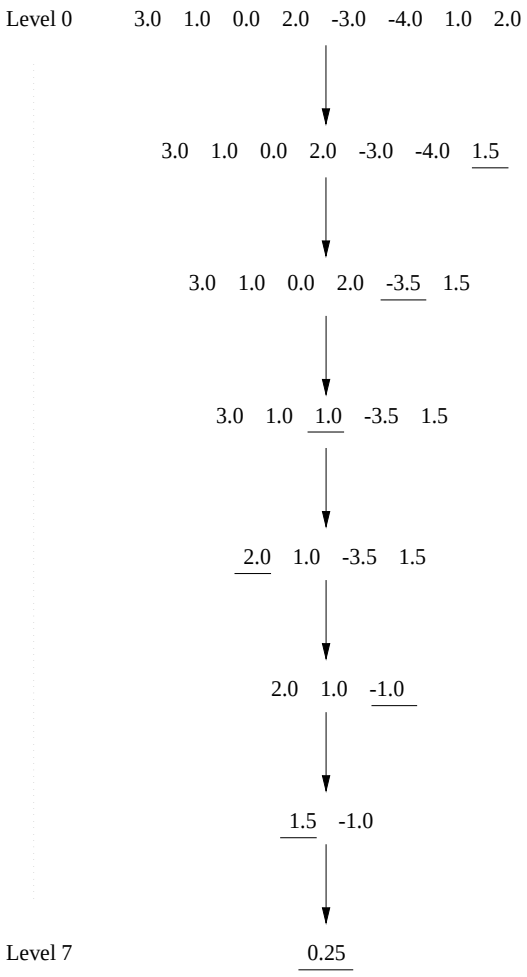

Figure 4.9: Removal of duplicated values by sampling

$\vec{z}$. Possible resolutions of sequence $\vec{z}$ are shown in Figure 4.8, with no truncation at Level 0, and 7 coefficients truncated at Level 7. The underlining signifies the value pair with resolution reduced.

Although we obtain lower resolution versions of a sequence, their lengths are still equal to the original sequence length. This does not constitute any improvement on time complexity as the lengths of sequences remain unchanged. Therefore, sampling is introduced for sequence length reduction.

From Figure 4.8, there exists some repeated values in different resolutions of sequences. One sample value can thus be taken out of those repeated ones. Consider the same figure, the sequence at Level 2 is sampled from {3.0,1.0,0.0,2.0,-3.5,-3.5,1.5,1.5} to {3.0,1.0,0.0,2.0,-3.5,1.5}, with the last two value pairs sampled down to one value each. By the same rule, the sampled sequence at the last level will be of length 1 which is {0.25}. Figure 4.9 shows the result after sampling of sequences in Figure 4.8.

4.2.2 Distance Compensation

With reduced sequence length, the computations involved in time warping distance can be drastically reduced. However, it is still inappropriate to estimate the original distance by the distance of lower resolution sequences, as distances arose from discarded value pairs are lost owing to the down sampling process, leading to severe underestimation of original time warping distance. The aim of using lower resolution sequences, down sampling, and then time warping is to first pair up as rapidly as possible the peaks and valleys of the two sequences accordingly. Afterwards, we should compensate for the distances lost owing to the down sampling process.

The compensation process is done as follows. Given two sequences $\vec{x}$ and $\vec{y}$ of

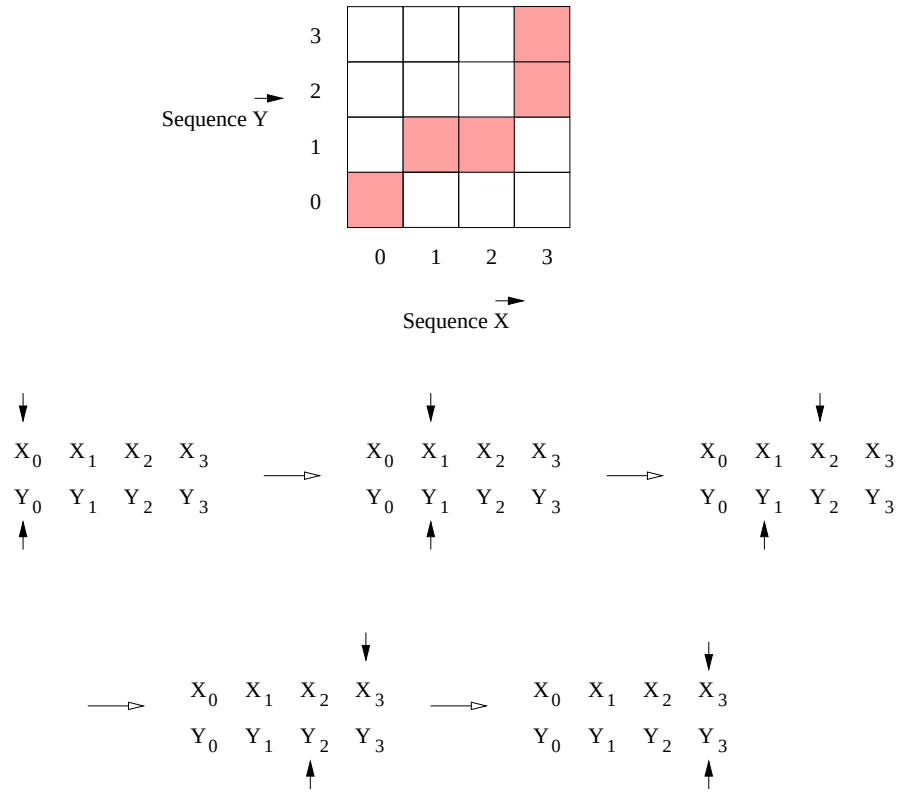

Figure 4.10: State transitions in finding time warping distance

length > 4, without loss of generality, we denote their low resolution versions as $\vec{X} = \{X_0, X_1, X_2, X_3\}$ and $\vec{Y} = \{Y_0, Y_1, Y_2, Y_3\}$, and assume that they have a time warping path shown in grey on the distance matrix in Figure 4.10. The length of the path corresponds to the number of pairs of matching values in time warping, which are five pairs in this case, namely, (X_0, Y_0), (X_1, Y_1), (X_2, Y_1), (X_3, Y_2), and (X_3, Y_3). They are visualized in lower part of the figure consisting of five different states, with arrows pointing to the matched pairs. The positions of these arrows are able to indicate where and how we make the distance compensation.

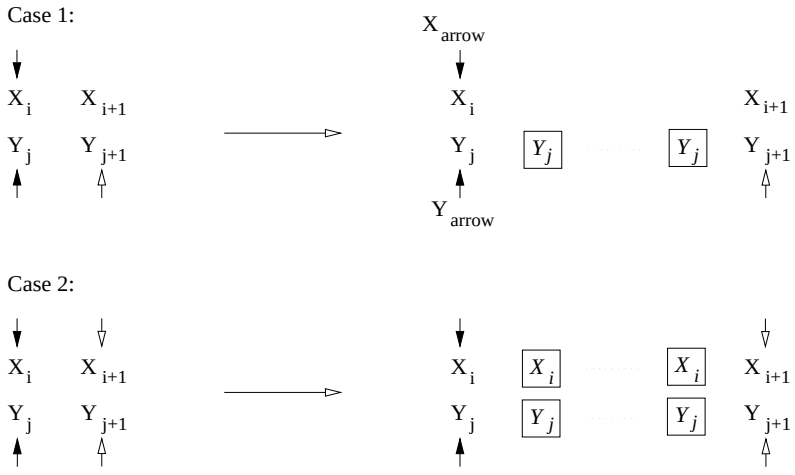

Figure 4.11: Distance compensation

There are two possible movements of the arrow pair when proceeding from one state to the next: either one arrow moves a unit forward, or both arrows move a unit forward. These situations are shown in Figure 4.11. For any consecutive elements $\mathbf{X_i}$, $\mathbf{X_{i+1}}$ and $\mathbf{Y_j}$, $\mathbf{Y_{j+1}}$ of sequences $\vec{X}$ and $\vec{Y}$ respectively, where i need not equal j, there exists some repeated values or elements which are in-between (removed by down sampling), represented as X_i and Y_j. Denote the arrows pointing sequences $\vec{X}$ and $\vec{Y}$ as X_{arrow} and Y_{arrow} respectively.

- *Case 1*

  Either X_{arrow} or Y_{arrow} moves forward. Without loss of generality, we assume Y_{arrow} moves a step forward pointing $\mathbf{Y_{j+1}}$ (shown in hollow arrow).

As the only movement is Y_{arrow}, we just need to compensate for distances between point $\mathbf{X_i}$ and subsequence $\{Y_j, \ldots, Y_j\}$, which is

$$DC = \{(\mathbf{X_i} - Y_j)^2 \times \|\{Y_j, \ldots, Y_j\}\|\}^{\frac{1}{2}}$$

such that the transition of Y_{arrow} from $\mathbf{Y_j}$ to $\mathbf{Y_{j+1}}$ is in continuity.

- *Case 2*

  Both X_{arrow} and Y_{arrow} move a unit forward. Distances from both subsequences $\{X_i, \ldots, X_i\}$ and $\{Y_j, \ldots, Y_j\}$ should be considered. Euclidean distance between $\{X_i, \ldots, X_i\}$ and $\{Y_j, \ldots, Y_j\}$ could be used, however, a better estimation involves the use of time warping distance. It is computed as follows.

$$\begin{aligned} DC \quad = \quad & \{D^2_{timewarp}(\{\mathbf{X_i}, X_i, \ldots, X_i, \mathbf{X_{i+1}}\}, \{\mathbf{Y_j}, Y_j, \ldots, Y_j, \mathbf{Y_{j+1}}\}) \\ & -(\mathbf{X_i} - \mathbf{Y_j})^2 - (\mathbf{X_{i+1}} - \mathbf{Y_{j+1}})^2\}^{\frac{1}{2}} \end{aligned}$$

  Using time warping distance is a better estimation to compensate for distances as it gives closer approximation to the original distance between the two time series. As the length of the repeated segment for a sample point is relatively short, the amount of computation involved is small.

Knowing how to compensate distance may lead us to the formula for finding the overall low resolution time warping distance between two time series $\vec{x}$ and $\vec{y}$ which is shown in the following equation

$$D_{lowresTW}(\vec{x}, \vec{y}) = \left\{ D^2_{timewarp}(\vec{X}, \vec{Y}) + \sum_{s=1}^{No.ofstates} DC_s^2 \right\}^{\frac{1}{2}} \tag{4.5}$$

where $\vec{X}$ and $\vec{Y}$ are the lower resolution versions of sequences $\vec{x}$ and $\vec{y}$ respectively, and DC_s is the distance compensated at state s. We can thus use $D_{lowresTW}$ to approximate $D_{timewarp}$ closely. The procedures in finding low resolution time warping distance are summarized in Figure 4.12.

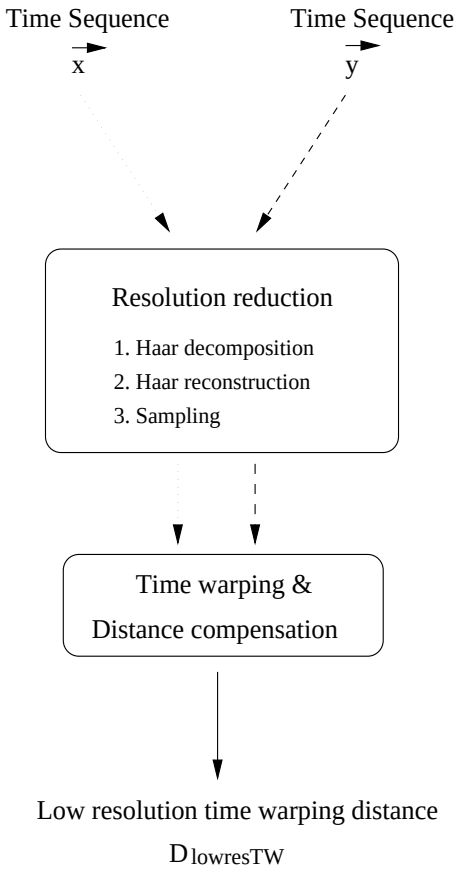

Figure 4.12: Procedures in finding low resolution time warping distance

Example To obtain the low resolution time warping distance between two sequences $\vec{x}$ = {3,1,0,2,-3,-4,1,2} and $\vec{y}$ = {2,3,1,-3,-4,1,0,1} of length $n = 8$, the following procedures are taken.

First we obtain the lower resolution versions of both $\vec{x}$ and $\vec{y}$ by finding their Haar coefficients using Equation (3.8), which are $H(\vec{x})$ = {0.71,3.54,1.0,-5.0,1.41,-1.41,0.71,-0.71} and $H(\vec{y})$ = {0.35,1.77,3.5,-2.0,-0.71,2.83,-3.54,-0.71} respectively. In order to determine the best k_{sam}, we employ both Equation (4.7) and Equation (4.10) and then take the average, $k_{sam} = (n^{\frac{1}{2}} + \left(\frac{n^2}{2}\right)^{\frac{1}{3}})/2 = 3$. Therefore, we replace the (8-3)=5 right most Haar coefficients each in $H(\vec{x})$ and $H(\vec{y})$ with zeros, resulting in $H(\vec{x})$ = {0.71,3.54,1.0,0,0,0,0,0} and $H(\vec{y})$ = {0.35,1.77,3.5,0,0,0,0,0}. Haar reconstructions are then performed using the modified $H(\vec{x})$ and $H(\vec{y})$ to obtain lower resolution sequences of $\vec{x}$ and $\vec{y}$, which are {2.0,2.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0} and {2.5,2.5,-1.0,-1.0,-0.5,-0.5,-0.5,-0.5} respectively. They are then reduced to $\vec{X}$ = {2.0,1.0,-1.0} and $\vec{Y}$ = {2.5,-1.0,-0.5} separately after sampling of duplicated time values.

Next, we find the value of $D_{timewarp}(\vec{X}, \vec{Y})$ which is 1.66, with time warping path $(\vec{X_0}, \vec{Y_0})$, $(\vec{X_1}, \vec{Y_0})$, $(\vec{X_2}, \vec{Y_1})$, and $(\vec{X_2}, \vec{Y_2})$. Afterwards, we perform the distance compensation. According to this path, two Case 1 and one Case 2 distance compensations are required. The last pair of time values $(\vec{X_2}, \vec{Y_2})$ also needs compensation as duplicated values follow after them, i.e. {-1.0,-1.0,-1.0,-1.0} of $\vec{x}$ and {-0.5,-0.5,-0.5,-0.5} of $\vec{y}$. The procedure for this compensation is nearly the same as for those described in Case 2. Finally, by substituting $D_{timewarp}(\vec{X}, \vec{Y}) = 1.66$, $DC_1 = 0.5$, $DC_2 = 1.5$, $DC_3 = 0$, and $DC_4 = 0.87$ into Equation (4.5), we obtain $D_{lowresTW} = \{1.66^2 + 0.5^2 + 1.5^2 + 0.87^2\}^{\frac{1}{2}} = 2.45$. ∎

4.2.3 Time Complexity

Since the computation time of low resolution time warping depends on the number of samples taken in resolution reduction, we are able to estimate the optimal number of sample that leads to minimum computation. Denote n as the length of time series and we reduce the resolution of sequences to only k_{sam} sample points. Hence, $\frac{n}{k_{sam}}$ corresponds to the length of repeated segment for each sample point. Since it is impossible to exhaust the state space for different paths in the cumulative distance matrix, we consider two representative paths that produce close approximations to the lower and upper bounds of the true computation, which are denoted as $P_{optimistic}$ and $P_{pessimistic}$ respectively.

- *Path* $P_{optimistic}$

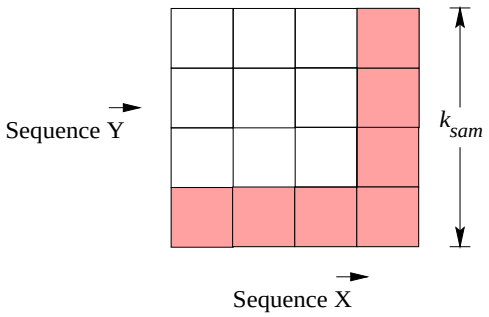

Figure 4.13: Cumulative distance matrix of $P_{optimistic}$

The first path $P_{optimistic}$ is shown in Figure 4.13. It composes of two shorter linear paths perpendicular to the x-axis and y-axis respectively. The trace of $P_{optimistic}$ leads to faster computation time in distance compensation, since only compensation of the first case (Figure 4.11) is involved which has a linear complexity. With $2n$ computations in resolution reduction of time sequence, k_{sam}^2 computations in finding the time warping path of the resolution reduced time series, and $2k_{sam} \times \frac{n}{k_{sam}} + \left(\frac{n}{k_{sam}}\right)^2$ computations

involved in distance compensation, the overall computation λ is expressed in Equation (4.6).

$$\begin{aligned} \lambda &= 2n + k_{sam}^2 + 2k_{sam} \times \tfrac{n}{k_{sam}} + \left(\tfrac{n}{k_{sam}}\right)^2 \\ &= 4n + k_{sam}^2 + \left(\tfrac{n}{k_{sam}}\right)^2 \end{aligned} \tag{4.6}$$

Differentiating λ with respect to k_{sam},

$$\frac{d\lambda}{d(k_{sam})} = 2k_{sam} - \frac{2n^2}{k_{sam}^3}$$

$$\begin{aligned} 0 &= 2k_{sam} - \tfrac{2n^2}{k_{sam}^3} \\ k_{sam}^4 &= n^2 \\ k_{sam} &= n^{\frac{1}{2}} \end{aligned} \tag{4.7}$$

Moreover,

$$\frac{d^2\lambda}{d(k_{sam})^2} = 2 + \frac{6n^2}{(k_{sam})^4} > 0$$

Hence $\lambda\,|_{k_{sam}=n^{\frac{1}{2}}}$ gives the minimum computation, and

$$\begin{aligned} \lambda\,|_{k_{sam}=n^{\frac{1}{2}}} &= 4n + n + (\tfrac{n}{n^{\frac{1}{2}}})^2 \\ &= 6n \\ &\leq n^2 \quad \text{for } n \geq 6 \end{aligned} \tag{4.8}$$

■

- *Path $P_{pessimistic}$*

For the second path $P_{pessimistic}$ in Figure 4.14, it runs along the diagonal of the cumulative distance matrix, which leads to slower computation in distance compensation, since all compensations involved are of Case 2. With the same computations involved in resolution reduction and time

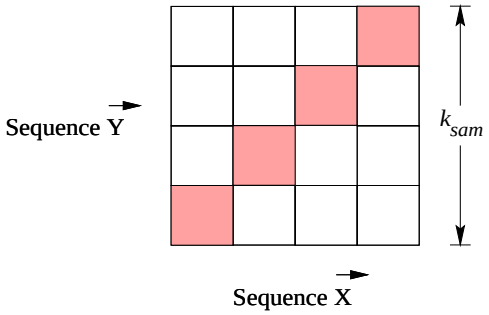

Figure 4.14: Cumulative distance matrix of $P_{pessimistic}$

warping path selection as for $P_{optimistic}$, and $k_{sam} \times \left(\frac{n}{k_{sam}}\right)^2$ computations for distance compensation, the overall computation is shown in Equation (4.9).

$$\begin{aligned} \lambda &= 2n + k_{sam}^2 + k_{sam} \times \left(\tfrac{n}{k_{sam}}\right)^2 \\ &= 2n + k_{sam}^2 + \tfrac{n^2}{k_{sam}} \end{aligned} \tag{4.9}$$

Differentiating λ with respect to k_{sam},

$$\frac{d\lambda}{d(k_{sam})} = 2k_{sam} - \left(\frac{n}{k_{sam}}\right)^2$$

$$\begin{aligned} 0 &= 2k_{sam} - \left(\tfrac{n}{k_{sam}}\right)^2 \\ k_{sam}^3 &= \tfrac{n^2}{2} \\ k_{sam} &= \left(\tfrac{n^2}{2}\right)^{\frac{1}{3}} \end{aligned} \tag{4.10}$$

Moreover,

$$\frac{d^2\lambda}{d(k_{sam})^2} = 2 + \frac{2n^2}{(k_{sam})^3} > 0$$

Hence $\lambda \mid_{k_{sam}=(\frac{n^2}{2})^{\frac{1}{3}}}$ gives the minimum computation, and

$$\begin{aligned}
\lambda\,|_{k_{sam}=(\frac{n^2}{2})^{\frac{1}{3}}} &= 2n + (\tfrac{n^2}{2})^{\frac{2}{3}} + \tfrac{n^2}{(\frac{n^2}{2})^{\frac{1}{3}}} \\
&= 2n + \tfrac{n^{\frac{4}{3}}}{2^{\frac{2}{3}}} + 2^{\frac{1}{3}} \times n^{\frac{4}{3}} \\
&= 2n + 2^{-\frac{2}{3}} \times 3 \times n^{\frac{4}{3}} \\
&= 2n + 1.9 \times n^{\frac{4}{3}} \\
&< 2(n + n^{\frac{4}{3}}) \\
&< n^2 \quad \text{for } n \geq 6
\end{aligned} \tag{4.11}$$

■

From both Equation (4.8) and Equation (4.11), we discover that the time complexity of low resolution time warping is rather linear, $\lambda = 6n$ for $k_{sam} = n^{\frac{1}{2}}$ and $\lambda = 2n + 1.9 \times n^{\frac{4}{3}}$ for $k_{sam} = (\frac{n^2}{2})^{\frac{1}{3}}$. In Table 4.1, the estimated and experimental values of K_{sam} are compared for various sequence lengths [5].

Sequence length	k_{sam}			
n	$n^{\frac{1}{2}}$	$\left(\frac{n^2}{2}\right)^{\frac{1}{3}}$	Experimental values	$(n^{\frac{1}{2}} + \left(\frac{n^2}{2}\right)^{\frac{1}{3}})/2$
64	8	13	8 - 20	11
128	11	20	16 - 17	16
256	16	32	26 - 32	24
512	23	51	34 - 37	37

Table 4.1: Optimal number of K_{sam} in low resolution time warping

In the table we show our estimated lower and upper bounds of optimal K_{sam} for a variety of sequence lengths. On the other hand, the experimental results are tabulated and they show the values of k_{sam} that give rise to minimum CPU time. We observe that the experimental values of k_{sam} are within the range of the bounds, except for sequence length of 64, where the maximum value of k_{sam} = 20 exceeds the upper bound value which is 13.

[5] The same experimental setup is used as in Section 4.4.

From another point of view, we are making use of these bounds to find optimal k_{sam}, therefore, we try to use the averages of these bounds as an estimation, which are shown in the last column. Excluding the third case, all the average values coincide with the experimental values. Though 24 is not the optimal k_{sam} for sequence length of 256, it is close enough to 26 such that the CPU time is nearly at the minimum.

4.3 Adaptive Time Warping

We propose another method of estimation to time warping distance by splitting the original time series into subsequences. The overall time warping distance is then estimated by summations of the partial time warping distances of these subsequences. The way of breaking sequence $\vec{x} = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ is shown in Figure 4.15.

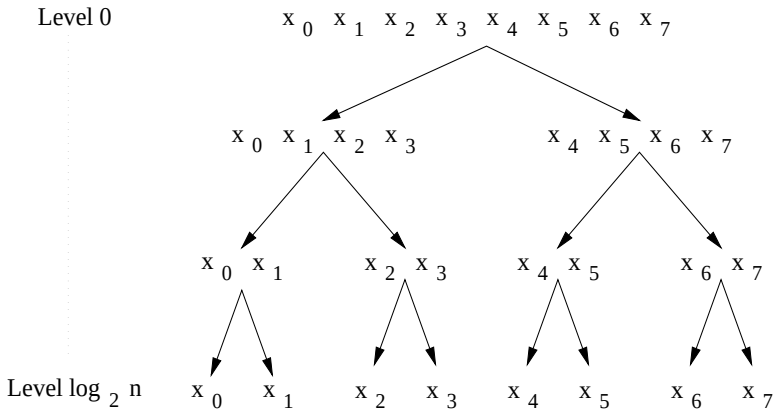

Figure 4.15: Sequence partition in adaptive time warping

At each level, each sequence is being partitioned into two equal halves. The last level is reached when $\vec{x}$ is splitted into separate time points, $\{x_0\}$,$\{x_1\}$,...,$\{x_7\}$. Denote the subsequences of $\vec{x}$ as $s\vec{x}$, $\{\ s\vec{x}_i | s\vec{x}_i \in \vec{x}$ and $s\vec{x}_i \cap s\vec{x}_j = \emptyset\ \}$. The adaptive time warping distance is defined as

$$D_{adaptiveTW}(\vec{x},\vec{y}) = \left\{ \sum_{s\vec{x}_i \in \vec{x}, s\vec{y}_i \in \vec{y}} D^2_{timewarp}(s\vec{x}_i, s\vec{y}_i) \right\}^{\frac{1}{2}} \tag{4.12}$$

Prior to any partition (Level 0), $D_{adaptiveTW} = D_{timewarp}$ obviously. If we partition the sequence into separate time points (Level $\log_2 n$), then $D_{adaptiveTW} = D_{Euclidean}$. Physically, splitting sequences into pieces to achieve adaptive time warping corresponds to the imposition of restrictions on the original time warping path in cumulative distance matrix. For splitted sequences at Level 1, the situation is shown in Figure 4.16.

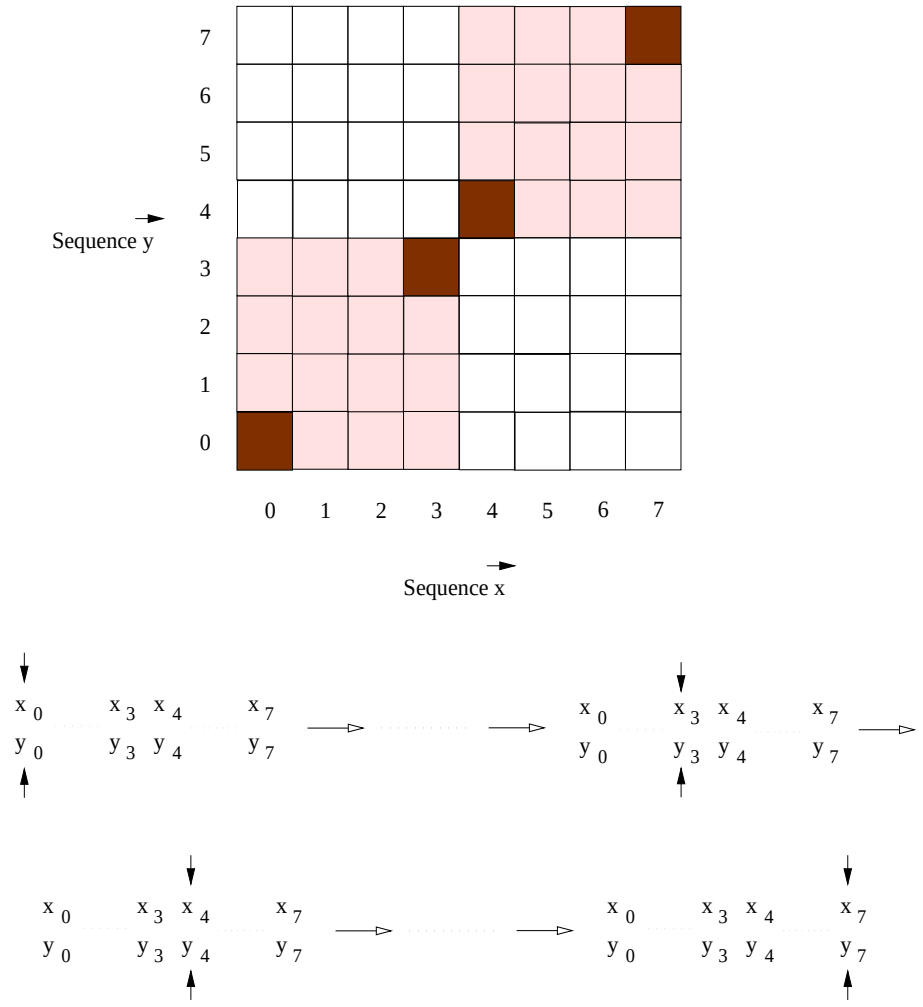

Figure 4.16: Pair restrictions in adaptive time warping

We show the cumulative distance matrix of sequences $\vec{x}$ and $\vec{y}$ in full length for better explanation. As a matter of fact, the time warping path should always pass through (x_0, y_0) and (x_7, y_7) of the cumulative distance matrix, as they are the starting and ending points of the two sequences. Partitions at the middle of $\vec{x}$ and $\vec{y}$ restrict this warping path to pass through (x_3, y_3) and (x_4, y_4). The reason is that (x_3, y_3) becomes the new ending point of the first subsequences, whilst (x_4, y_4) is the new starting point of the second subsequences. These four restriction pairs correspond to four different states depicted in the diagram of Figure 4.16. With these restriction pairs, the search space now has been reduced by half, which is shown as light grey regions. Therefore, only half of the computations are needed. The more the partitions of the sequences, the faster the computation time, however, the larger deviation from the real time warping distance. This is a trade off between accuracy in distance estimation and time complexity.

4.3.1 Time Complexity

The real time warping distance $D_{timewarp}$ always gives the shortest cumulative distance in the matrix. With more pair restrictions, the cumulative distance will increase. Therefore, $D_{adaptiveTW}$ would upper bound $D_{timewarp}$ all the time. Denote $D_{adaptiveTW}$ obtained at partition Level i as DA_i,

$$D_{timewarp} \leq DA_i \quad \text{for } 0 \leq i \leq \log_2 n \tag{4.13}$$

From the facts that $DA_0 = D_{timewarp}$ and $DA_{\log_2 n} = D_{Euclidean}$, hence the time complexity of adaptive time warping is

$$O(n) \leq O_{DA_i} \leq O(n^2) \qquad \text{for } 0 \leq i \leq \log_2 n$$

Note that we can achieve more flexibility of sequence partition by allowing *unsynchronized* splitting of subsequences, the methodology is similar to the ones introduced in Section 4.2.1 for low resolution time warping in achieving a diversity of resolutions.

4.4 Performance Evaluation

Experiments using synthetic data have been carried out. All experiments are conducted on a Sun UltraSPARC-1 workstation with 592MBytes of main memory. Synthetic random walk data consisting of 5k time sequences are generated. The length of sequences ranges from 64 to 512. All results are obtained from the average of 50 trials.

4.4.1 Accuracy versus Runtime

Since we aim to show the accuracy and effectiveness of low resolution and adaptive time warpings for real time warping approximation, query is raised following by an exhaustive match of each time series in the database. Results are shown from Figure 4.17 to Figure 4.21.

In Figure 4.17, the fraction of distance to real time warping against sample length / number of partitions is investigated (original length of time series is 256). To low resolution time warping, sample length refers to the number of sample values taken, that is K_{sam}. To adaptive time warping, number of partitions refers to the number of subsequences resulting from partition process.

The approximation of low resolution to real time warping distance is accurate as observed, the fraction of estimation is close to 1.0 and is bounded between 1.0 and 1.3 on average. The larger the value of K_{sam} of low resolution time warping,

the more accurate the approximation to the real time warping distance. Larger K_{sam} means fewer time values of the original time series are being sampled. Moreover, the number of distance compensations are also reduced. Both of them lead to a closer approximation to the actual time warping distance. $D_{lowresTW}$ will be equal to $D_{timewarp}$ in the extreme when K_{sam} is the same as the full length of the original time series.

For adaptive time warping, the distance estimated ranges from 1.0 to 1.75 on average and should always be greater than or equal to 1.0 by the upper bounding property in Equation (4.13). With no partition, the fraction is equal to 1.0, and rapidly stabilizes to around 1.75 with increasing number of partitions, as more restrictions are imposed on the time warping path. We observe that the saturation occurs around 60 partitions and it finally converges to Euclidean distance of time series.

The distance estimated by the lower bound distance function D_{lb} mentioned in Section 4.1 is also shown for reference. As the lower bound distance function makes use of the full time series, the distance estimated is invariant to the sample length and is found to be around 0.3, i.e 30% of the real time warping distance.

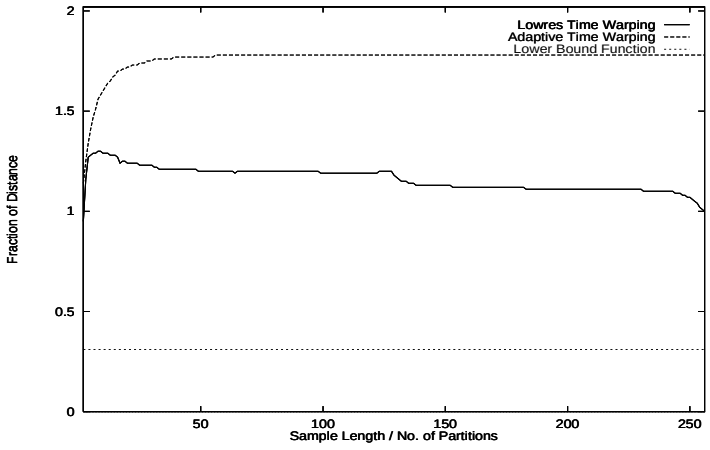

Figure 4.17: Fraction of distance estimated (Sequence length = 256)

The CPU time required by various time warping methods is shown in Figure

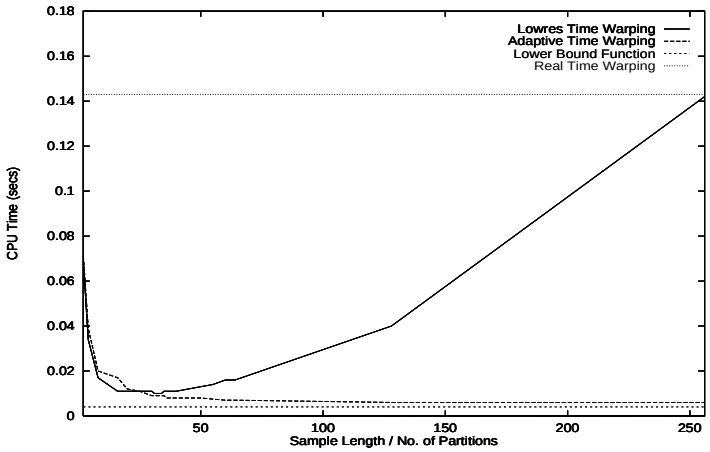

Figure 4.18: CPU time of different time warping methods (Sequence length = 256)

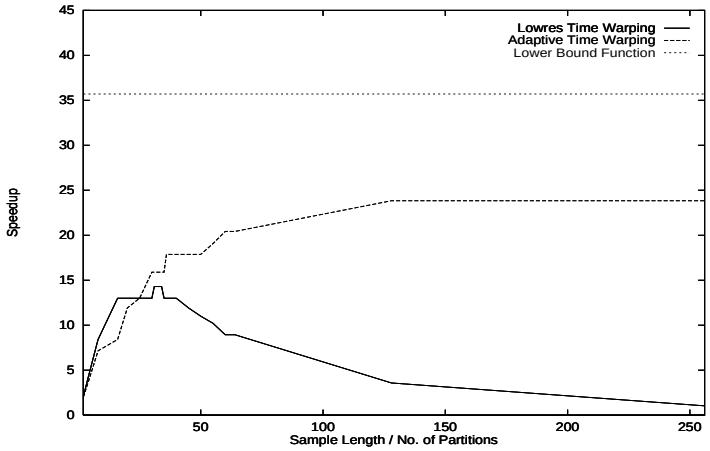

Figure 4.19: Speedup (Sequence length = 256)

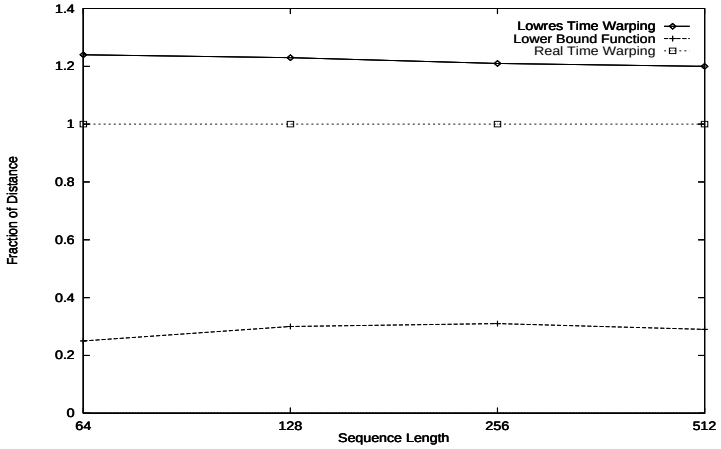

Figure 4.20: Fraction of distance estimated at best CPU time

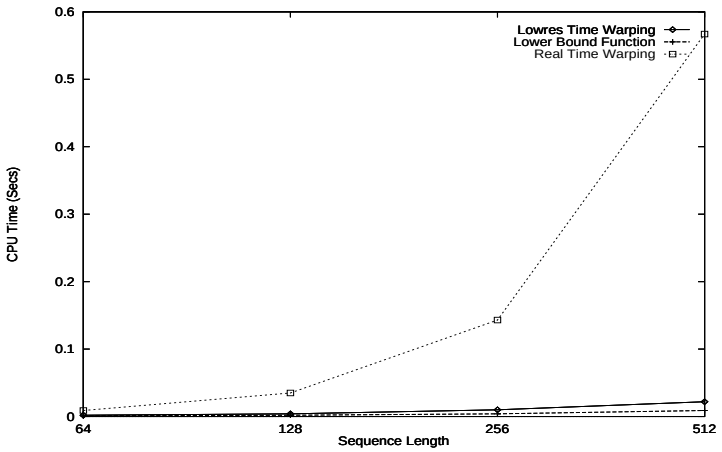

Figure 4.21: Best CPU time at various sequence lengths

4.18. The real time warping consumes enormous CPU time compared with other three methods due to its $O(n^2)$ complexity. Initially low resolution time warping experiences a drop in CPU time to a minimum, followed by a gradual increase until it reaches the same CPU time required by real time warping. Low resolution time warping using full sample length is equivalent to real time warping, thus, they have the same CPU time and the fraction of distance estimated would be equal to 1.0 which is depicted in Figure 4.17. Minimum CPU time of low resolution time warping could be obtained by determining the optimal sample length k_{sam} for time series being described in Section 4.2.3. In this experiment, the optimal dimension ranges from 26 to 32 for sequences of length 256, which follows the prediction in Table 4.1 and Figure 4.18.

On the other hand, the CPU time required by adaptive time warping decreases with the number of partitions. The initial drop in CPU time is similar to low resolution time warping, and it continues to decrease upon further partitions on the sequence. Meanwhile, the complexity goes from $O(n^2)$ to $O(n)$. The minimum CPU time would be the time taken to find the Euclidean distance between two time series, which corresponds to the point with number of partitions = full length of time sequence. Lower bound distance function requires the least CPU time by having $O(n)$ complexity and the simplicity in finding D_{lb}.

The speedup in terms of ratio of CPU time is found in Figure 4.19. The maximum speedups attained by low resolution and adaptive time warpings are 14 and 24 times respectively. At these speedup factors, low resolution time warping achieves an estimation of distance at 1.21, while adaptive time warping attains 1.7 on average. The accuracy of adaptive time warping can be improved by trading off CPU time. Though the lower bound distance function achieves a speedup of 36 times, the distance estimated is too low (30%) to be effective for distance approximation or filtering purpose. Moreover, the inability to trade off CPU time for accuracy makes it prohibitive.

In order to ensure performance at different sequence lengths, we carry out the experiment as in Figure 4.17 with different lengths of sequences. The fraction of distance estimated at best CPU time for various sequence lengths is shown in Figure 4.20. The fraction of distance that low resolution time warping estimated is relatively constant around the value 1.2, while the estimation by lower bound distance function is close to 0.3. In the figure, it is obvious that low resolution time warping is able to give a better approximation to real time warping distance (an +0.2 overestimation) compared with the lower bound distance function (an -0.7 underestimation), and this phenomenon seems to persist in a variety of sequence lengths. The persistence of fraction of distance is important in the sense that different lengths of sequences share nearly the same factor, thus other sequence lengths will produce likely the same amount of overestimation, which is used in turn to estimate the real time warping distance. There is no best CPU time exists for adaptive time warping as CPU time is a direct trade off for accuracy of distance estimation. Therefore, it is inappropriate to make comparison in this case.

Figure 4.21 shows the best CPU time attained for various sequence lengths. Having high complexity, real time warping scales bad in CPU time with sequence length. Doubling in sequence length gives rise to quadruple of CPU time. In the

meantime, both low resolution time warping and lower bound function maintain a linear increase in CPU time and the speedup could be enormous for lengthy sequences. The linearity of low resolution time warping is consistent with the estimations in both Equation (4.8) and Equation (4.11).

4.4.2 Precision versus Recall

The following experiments are carried out to evaluate the performance of low resolution time warping when acting as a filtering function in the post-processing step of similarity search. Performance is described in terms of precision and recall defined in Equation (4.2) and Equation (4.3) respectively. The number of false alarms as well as the number of false dismissals generated are also studied. The experimental setup is all kept the same with only two exceptions. First, we want to perform a search [6] to look for candidate sequences, rather than exhaustively *match* all sequences in the database with the query as in Section 4.4.1. Therefore, an appropriate epsilon range $\epsilon_{timewarp}$ should be employed to obtain a reasonable amount of candidate sequences. In our experiment, the values of epsilon range from 0.5% to 5% of the database size, and results are drawn from the average of these epsilon ranges.

Moreover, the original epsilon range $\epsilon_{timewarp}$ should be adjusted in accordance with the fraction of distance obtained in previous experiments (Figure 4.20) when matching sequences using low resolution time warping in the post-processing step, the adjusted epsilon is denoted as $\epsilon_{lowresTW}$. After all, more false alarms but fewer false dismissals appear for an increase in epsilon range, while a decrease in the range reverses the effect. Our task is to obtain a modest range that results in a tiny number or none of false dismissal, and at the same time suppresses the number of false alarms.

[6] Note that we emphasize on the performance comparison of filtering functions, hence we bypass the use of Fastmap index for the sake of simplicity.

The fraction of distances estimated for various sequence lengths are from Figure 4.20 and tabulated in Table 4.2 accompanied with their standard deviations. The value of fraction of distance estimated and its associated standard deviation act effectively as an indicator for the adjustment of the epsilon range $\epsilon_{lowresTW}$ in low resolution time warping. The adjustment can be expressed by the following equation

$$\epsilon_{lowresTW} = dist\_frac \times \epsilon_{timewarp} + c \times s.d. \quad \text{for } c \geq 0 \tag{4.14}$$

where $\epsilon_{timewarp}$ is the epsilon range used in real time warping distance, $dist\_frac$ represents the fraction of distance estimated, $s.d.$ is the standard deviation at a particular $dist\_frac$. By varying the value of c, the range $\epsilon_{lowresTW}$ can be systematically adjusted. The best dimensions used in the experiments for low resolution time warping are listed in the same table for reference.

Sequence length n	Fraction of Distance $dist\_frac$	S.D.	Best K_{sam}
64	1.25	0.11	20
128	1.24	0.13	17
256	1.21	0.10	32
512	1.21	0.12	37

Table 4.2: Fraction of distance estimated and associated standard deviation

Figure 4.22 to Figure 4.25 show the performance with different epsilon ranges $\epsilon_{lowresTW}$. In Figure 4.22, the number of false alarms and false dismissals are shown for querying a database of sequence length of 256. We observe that when we scale the value of the epsilon only by a factor of $dist\_frac$ with zero $s.d.$, i.e. $\epsilon_{lowresTW} = 1.21 \times \epsilon_{timewarp}$, there are 85 false dismissals [7] being generated. On

[7] Number of false alarms and dismissals should be compared to the average number of qualified sequences in the answer set, which is = database size $\times$ average epsilon $\epsilon_{timewarp}$ = $5000 \times 2.75 = 137.5$.

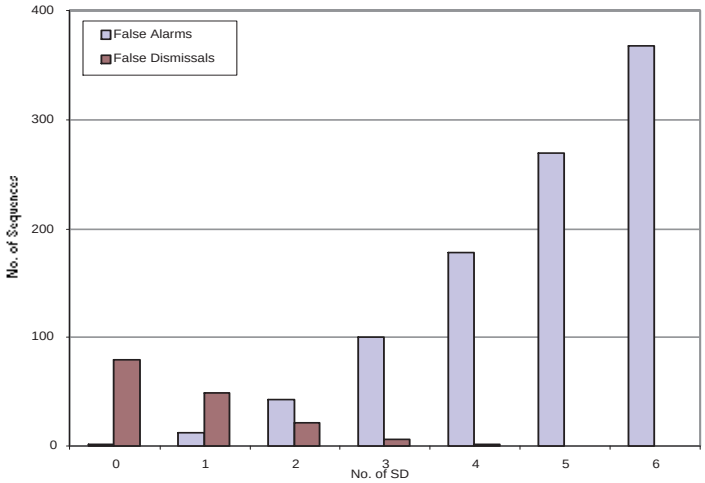

Figure 4.22: Number of false alarms and false dismissals (Sequence length = 256)

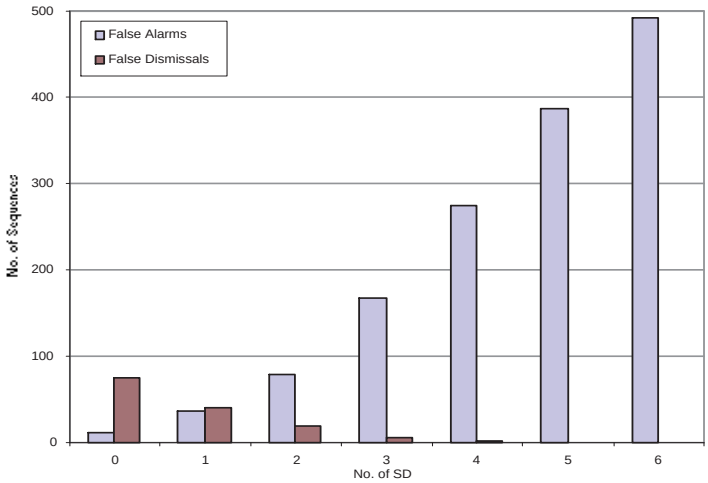

Figure 4.23: Average number of false alarms and false dismissals

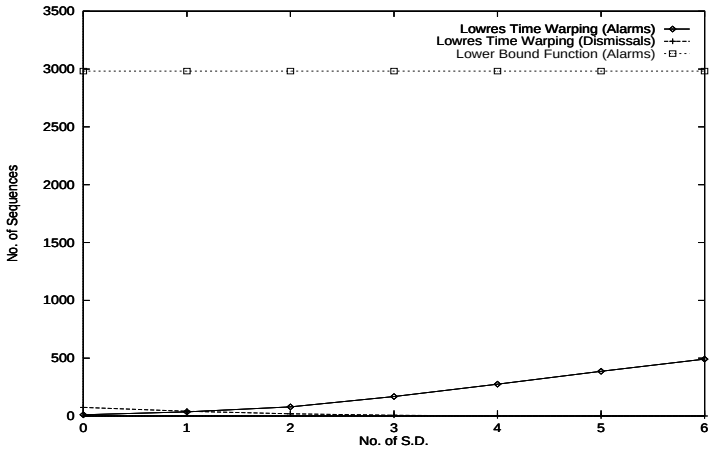

Figure 4.24: Average number of false alarms compared with D_{lb}

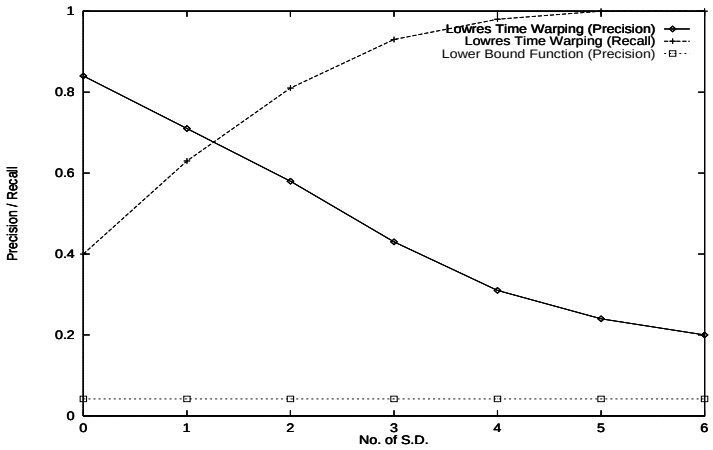

Figure 4.25: Average precision and recall

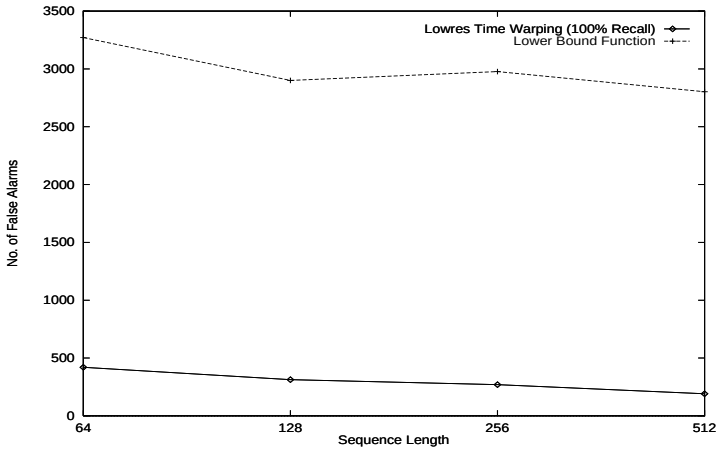

Figure 4.26: Number of false alarms at various sequence lengths

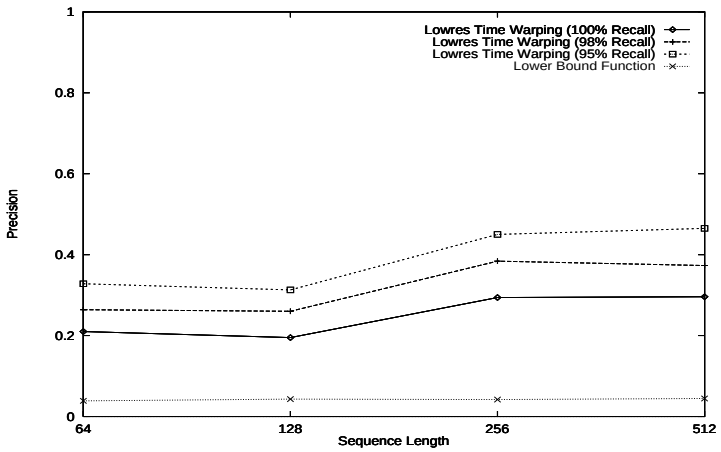

Figure 4.27: Precision at various sequence lengths

the contrary, the number of false alarms being generated is small. This seems to be natural since we are using solely $dist\_frac$ as the scaling factor, which measures the overestimation *on average*. As reflected in the figure, the larger the value of $\epsilon_{lowresTW}$ (through increment of c in Equation (4.14)), the fewer the number of false dismissals, but the more the number of false alarms. For $c = 5$, no false dismissal is recorded while the number of false alarms increased to 390 sequences.

The same experiment is carried out separately for databases of sequence length of 64, 128, and 512. Their results are averaged and shown in Figure 4.23. A similar trend is observed as in Figure 4.22. Moreover, we notice that for the same value of c, there are more false alarms in this case than that for 256 units long sequence database. It is because more false alarms appearing for shorter sequence lengths (64 and 128), hence more false alarms are observed in the figure. This phenomenon is explained later with Figure 4.26.

We have demonstrated in Section 4.1 that the fraction of real time warping distance estimated by the lower bound distance function is quite small, although no false dismissal will be generated. The enormous amounts of false alarms produced by D_{lb} shown in Figure 4.24 confirms this fact. There are 3k false

alarms being generated by D_{lb} compared with 0.35k (recall = 1.0), i.e. a 8.5 times improvement. The amount of false alarms produced by low resolution time warping is still tiny for large S.D. value with respect to D_{lb}.

This enables a significant performance gain by low resolution time warping technique, since most computations are involved in the matching between the candidate sequences and the query in the post-processing step. Having $O(n^2)$ complexity, the matching in real time warping distance will consume enormous CPU time if the filtering function fails to prune away false alarms effectively, which is the case for lower bound distance function. In contrast, the filtering power of low resolution time warping is overwhelming which results in a conceivable significant outperformance.

In Figure 4.25, the precision and recall of low resolution time warping are shown. The precision of lower bound function is also included for reference. For low resolution time warping, precision decreases while recall increases with $\epsilon_{lowresTW}$. Their trends correspond to the rise of false alarms and the drop of false dismissals respectively in Figure 4.23. The value of recall recorded is 0.4 at $c = 0$, and gradually increases to 1.0 at $c = 5$, where no false dismissal occurs. Meanwhile, precision drops from 0.82 to 0.25. On the other hand, low bound function offers a precision of 0.04, meaning that 96% of the retrieved sequences go to false alarms, which require a great deal of computations to get rid of.

The performance of the two techniques are compared for a variety of sequence lengths in Figure 4.26 and Figure 4.27. For Figure 4.26, we show the number of false alarms generated by both filtering functions (recall = 1.0 for low resolution time warping). Both methods experience a decline in the number of false alarms with lengthy sequences. We observe that for lengthy sequences, the alignments of those peaks and valleys are rather localized, since the time series of financial data consist of time values fluctuating around some levels, which are relatively con-

stant locally without abrupt change. Therefore, it is very unlikely that the head of a sequence will align with its tail. This leads to a more uniform distribution of $D_{lowresTW}$ and D_{lb}. In addition, both $D_{lowerTW}$ and D_{lb} will distribute in neater proportion to the real time warping distance $D_{timewarp}$, such that large $D_{timewarp}$ is less likely to end up with relatively small $D_{lowresTW}$ and D_{lb}. Thus, more non-qualified sequences can be filtered yet without appearing as false alarms.

The performance gap is maintained for different sequence lengths. Compared to low resolution time warping, at most 15 times more false alarms are recorded for lower bound function for sequence length of 512, and at least 8 times improvement is recorded for sequence length of 64.

The precision at various sequence lengths is depicted in Figure 4.27. While lower bound function maintains a low precision around 0.04 for different sequence lengths, low resolution time warping offers tremendous improvements in precision, ranging from 0.2 to 0.3 at recall = 1.0 (or 100% recall), which is 6 times better on average. The rise in precision for longer sequences directly corresponds to the drop in the number of false alarms in Figure 4.26.

In addition, we show the precision of low resolution time warping at recall = 0.98 and 0.95. Smaller recall value provides better precision, since the epsilon range $\epsilon_{lowresTW}$ is reduced, which can suppress the generation of further false alarms. The results are encouraging, the improvements of low resolution time warping over lower bound distance function are 8 and 10 times at 98% and 95% recall respectively.

4.4.3 Overall Runtime

In this experiment we measure the overall CPU time of similarity search based on our strategy. The time measurement commences once the query is raised

and ends when all result sequences are returned from the post-processing step. Experimental setup is kept the same as in Section 4.4.2. Results are shown in Figure 4.28 and Figure 4.29.

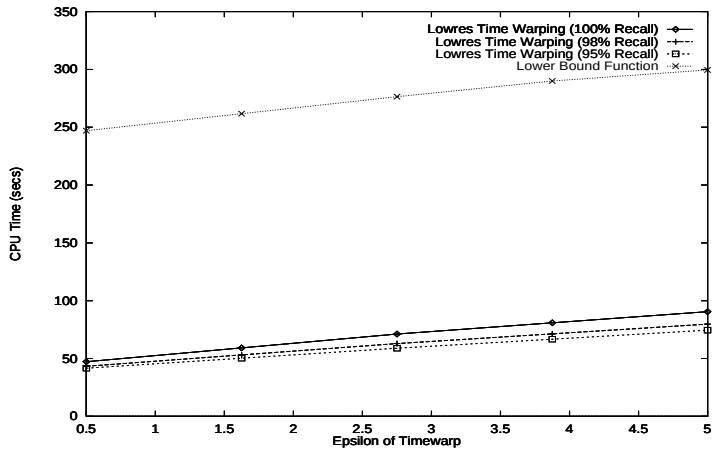

Figure 4.28: Overall CPU time (Sequence length = 256)

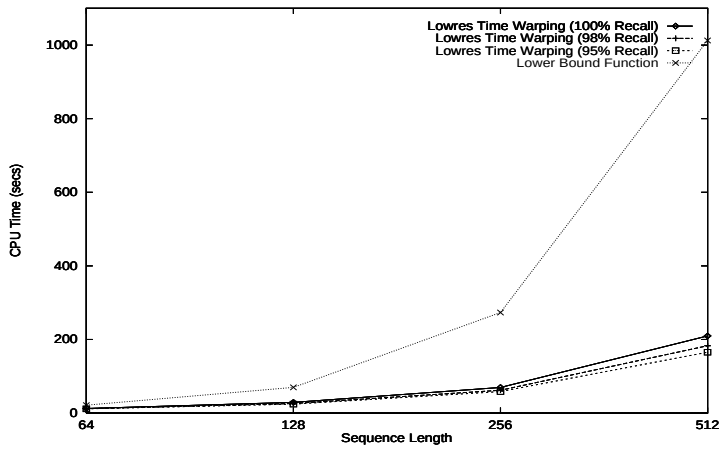

Figure 4.29: Overall CPU time at various sequence lengths

For Figure 4.28, we consider the CPU time with varying epsilon ranges $\epsilon_{timewarp}$ at sequence length of 256. We observe 5 times and 3 times improvements of low resolution time warping (recall = 1.0) over lower bound function at search range $\epsilon_{timewarp} = 0.5\%$ and 5% respectively. The smaller the $\epsilon_{timewarp}$, the greater the outperformance. Roughly 50 seconds more CPU time are recorded for both methods when $\epsilon_{timewarp}$ is widen from 0.5% to 5%. Hence, they share

a similar linear scaling with epsilon range increase, though a performance gap is maintained between the two methods. The CPU time of low resolution time warping at recall = 0.98 and 0.95 is shown in addition. Lower recall values require less computation time since fewer qualified time sequences are retrieved, thus fewer sequences are being matched using real time warping distance. This accounts for the reduction in CPU time for lower recall values.

The CPU time of different time series lengths is shown in Figure 4.29. Low resolution time warping outperforms lower bound distance function in tremendous amounts, especially for lengthy sequences. It has a much better scaling with sequence length increase. The CPU time is kept relatively low for different sequence lengths by employing low resolution time warping, compared with the drastic increase for the lower bound distance function. Numerically, a 6 times improvement is achieved for sequence length of 512, and at least 2 times improvement is recorded for sequence length of 64. Being consistent with the previous experiment, low recall value of low resolution time warping consumes less overall CPU time at different lengths of sequences.

4.4.4 Starting Up Evaluation

Note that in our performance evaluation, the two important indicators including the fraction of distance estimated in Figure 4.17, and the best epsilon range $\epsilon_{lowresTW}$ in Equation (4.14) are obtained *statistically*. These results are valid for our particular time series database. However, for other sets of time series data, results may vary accordingly. Therefore, before starting up a new similarity querying database based on our strategy, we advise the two kinds of experiments being carried out as in our evaluation to find the values of these indicators. In case for enormous database, sampling of sequences could be adopted to reduce the running time of the evaluation process. Upon incremental update of new

time series, evaluation could be carried out again, depending on the number of new sequences added and the nature of time series. For database composed of heterogeneous time series, more re-evaluations are anticipated upon incremental updating. In fact, these evaluations could be performed at the same time whenever Fastmap index is reorganized (Section 2.2), to keep up the performance of similarity search.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

As time series data are of growing importance, we need to manage sequence data in database systems. For many applications such as prediction, decision making, the system is given a query sequence and we should return similar time sequences efficiently and precisely.

First we propose an efficient time series matching technique through dimension reduction by Haar Wavelet Transform. The first few coefficients of the Haar transformed sequences can be indexed in an R-Tree or other similar indices for similarity search. Experiments show that our method outperforms the F-index (Discrete Fourier Transform) method in terms of pruning power, number of page accesses and complexity. In addition, a new similarity model is introduced to deal with vertical shifts of sequences. The proposed v-shift model has a better description of similarity between two sequences and is also shown to have better performance when compared with the non-v-shift model of DFT. Furthermore, an efficient two-phase nearest neighbor search is proposed and its effectiveness is demonstrated by experiments.

Our time series matching strategy is capable of handling Euclidean distance or v-shift based similarity search effectively. For similarity search of time shifted sequences, we turn to the extensively used time warping techniques. However, the high complexity involved hinders its use. Even worse, false dismissals are produced when we directly apply indexing techniques for time warping distance. Therefore, we suggest the low resolution and adaptive time warpings to approximate the real time warping. In addition, low resolution time warping can act as a filtering function in the post-processing step in index-based similarity search. Experiments show that both techniques provide close approximation and achieve significant speedup. Moreover, low resolution time warping is shown to be effective in suppressing the number of false alarms generated in the post-processing step, this in turn consumes fewer computations in matching with real time warping distance that follows. It outperforms the lower bound distance function with an order of magnitude improvement.

5.2 Future Work

We have some suggestions for future work.

5.2.1 Application of Wavelets on Biomedical Signals

We are seeking for opportunity to apply families of wavelets that do not work well with stock data in biomedical signals, such as electrocardiographs (ECGs) and electrogastrograms (EGGs) for dimension reduction. Stock data are more stationary in the sense that the trends of stocks are gentle without abrupt change. This is totally different from biomedical signals which may be full of chirps. An example of ECG is shown in Figure 5.1.

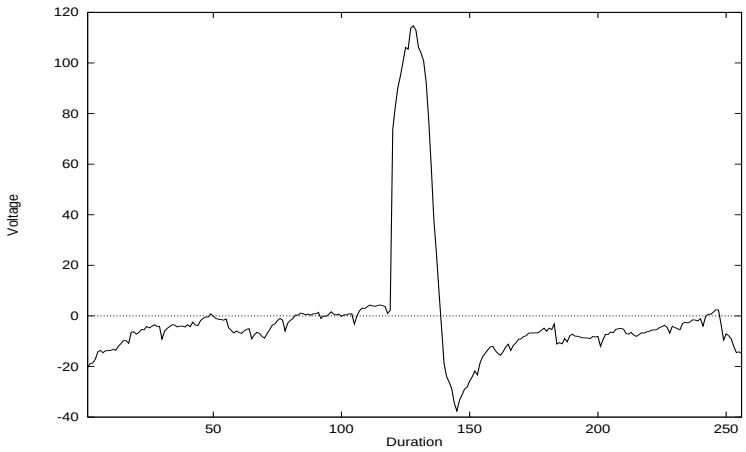

Figure 5.1: An example of ECG signal

The effects on various kinds of signals using different wavelet families may vary. Haar wavelets are shown to be effective in extracting features for stock sequences. But they may fail when applying to ECG sequences. However, the wide choice of wavelet family enables us to select the tailored one for a particular kind of signal, that can maximize the extraction of features. Using DFT or Piecewise Fourier Transform on biomedical signals, is thus less appealing and effective.

In fact, wavelets have been widely used as analyzing tools in biomedical signals [7, 6] for signal detection, de-noising, compression, and feature extraction. Although wavelets are found to be useful in many areas of biomedical signals, nothing has been done related to signal retrieval. Therefore, we are going to evaluate wavelet families for these kinds of signals and try to develop mathematical property on Euclidean distance preservation.

5.2.2 Moving Average Similarity

Moving average on time series eliminates short term fluctuations by averaging adjacent time values. While the transformation by Haar wavelets is equivalent to averaging the original sequence of adjacent pairs to achieve multi-resolution representation. This close relationship between moving average and Haar transformation suggests us to match Haar coefficients in order to retrieve similar moving-averaged sequences. Though Haar coefficients provide good estimation of moving-averaged distance of two sequences, false dismissals will result as we can not guarantee the distance of Haar coefficients lower bounds the moving-averaged distance. We will evaluate the effectiveness of this matching strategy by considering both the precision and the recall of query experimentally.

5.2.3 Clusters-based Matching in Time Warping

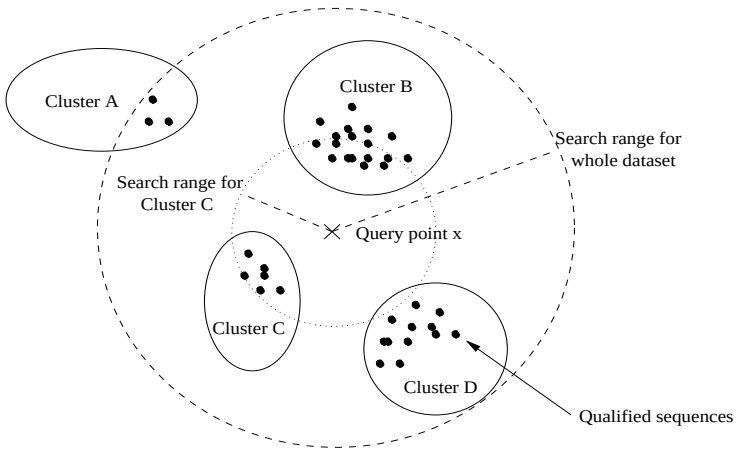

Figure 5.2: Querying on clustered time series database

It has been revealed in Section 4.1 that time warping matching based on K-L transform is inefficient in terms of precision and recall. High precision leads to low recall, while high recall gives rise to low precision. To mitigate this problem,

we suggest to partition the time series database into different clusters in a pre-processing step using one of the many clustering methods like K-Mean [30] , which aims at grouping similar-shaped time series into clusters prior to the building of Fastmap index. Given a query sequence $\vec{x}$, we try to extract sequences from these clusters, C_i where $1 \leq i \leq N$ and N is the total number of clusters. Search ranges for clusters are independent. A heuristic to determine the search range for cluster i maybe comparing the time warping distance between the centroid of cluster i and the query point, which is $D_{timewarp}(\vec{x}, \vec{Centroid_i})$. In addition, the search range should be chosen by taking into the consideration of $\epsilon_{timewarp}$. The advantage of querying in terms of clusters is demonstrated in Figure 5.2.

Without loss of generality, we assume the time series database is partitioned into four clusters. For the given query point, a large search range (outer dotted-lined circle) should be used in the original approach to attain high recall. However, this high recall sacrifices the precision, so that almost all time series in the database are retrieved and results in large amounts of false alarms. On the contrary, if we query time series cluster by cluster, the search range (inner dotted-lined circle) can be reduced tremendously and fewer false alarms will appear from each cluster.

The performance of this clusters-based approach is enhanced with the replacement of the lower bound function, which is shown to be inefficient in previous experiment, with the proposed low resolution or adaptive time warping functions as described before.

Bibliography

[1] Johnson Ihyeh Agbinya. Discrete wavelet transform techniques in speech processing. *IEEE TENCON - Digital Signal Processing Applications*, pages 514–519, 1996.

[2] Rakesh Agrawal, Christos Faloutsos, and Arun swami. Efficient similarity search in sequence databases. In *Procs. of the Fourth International Conference on Foundations of Data Organization and Algorithms*, 1993.

[3] 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 *Procs. of the 21st VLDB Conference*, pages 490–501, 1995.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. *IEEE*, pages 3–14, 1995.

[5] Ali N. Akansu and Richard A. Haddad. *Multiresolution Signal Decomposition.* Academic Press, 1992.

[6] Metin Akay. *Detection and Estimation Methods for Biomedical Signals.* Academic Press, 1996.

[7] Metin Akay, editor. *Time Frequency and Wavelets in Biomedical Signal Processing.* IEEE Press, 1998.

[8] Ricardo Baeza-Yates and Gaston H. Gonnet. A new approach to text searching. *Communication of the ACM*, 35, 1992.

[9] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 322–330, 1990.

[10] John J. Benedetto and Michael W. Frazier. *Wavelets – Mathematics and Applications.* CRC, 1994.

[11] Stefan Berchtold, Christian Bohm, and Hans-Peter Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *Procs. of ACM SIGMOD Conference on Management of Data*, 1998.

[12] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. An index structure for high-dimensional data. In *Procs. of the 22nd VLDB Conference*, 1996.

[13] Donald J. Berndt and James Clifford. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1995.

[14] Bela Bollobas, Gautam Das, and Dimitrios Gunopulos. Time-series similarity problems and well-separated geometric sets. In *Procs. of the 21st VLDB Conference*, 1995.

[15] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms, A Primer.* Prentice Hall, 1997.

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

[17] King Lum Cheung and A. Fu. Enhanced nearest neighbor search on the R*-tree. *ACM SIGMOD Record, to appear*, Sept, 1998.

[18] K. W. Chu, S. K. Lam, and M. H. Wong. An efficient hash-based algorithm for sequence data searching. Master's thesis, The Chinese University of Hong Kong, 1997.

[19] Tzi cker Chiuen. Content-based image indexing. In *Procs. of the 20th VLDB Conference*, pages 582–593, 1994.

[20] Gautam Das, Dimitrios Gunopulos, and heikki Mannila. Finding similar time series. In *Procs. of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, 1997.

[21] Tim Edwards. Discrete wavelet transforms: Theory and implementation. Technical report, Stanford University, 1991.

[22] C. Faloutsos, H. V. Jagadish, and A. O. Mendelzon. A signature technique for similarity-based queries. In *Procs. of Compression & Complexity of Sequences*, 1997.

[23] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 419–429, 1994.

[24] Dennis Gabor. Theory of communication. *Journal of the Institute of Electrical Engineers*, 93(22):429–257, 1946.

[25] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison Wesley, 1992.

[26] Amara Graps. An introduction to wavelets. IEEE, 1995.

[27] K. Grochenig and W. R. Madych. Multiresolution analysis, haar bases, and self-similar tilings of r^n. *IEEE Trans. of Information Theory*, 38(2):556–568, 1992.

[28] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 47–57, 1984.

[29] Alfred Haar. Theorie der orthogonalen funktionen-systeme. *Mathematische Annalen*, 69:331–371, 1910.

[30] John A. Hartigan. *Clustering Algorithms.* Wiley, 1975.

[31] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. In *Procs. of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1995.

[32] K. V. Ravi Kanth, Divyakant Agrawal, and Ambuj Singh. Dimensionality reduction for similarity searching in dynamic databases. In *Procs. of ACM SIGMOD Conference on Management of Data*, 1998.

[33] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Procs. of ACM SIGMOD Conference on Management of Data*, 1997.

[34] Cory S. Myers and Lawrence R. Rabiner. A level building dynamic time warping algorithm for connected word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 29(2), 1981.

[35] Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. Walrus: A similarity retrieval algorithm for image databases. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 395–406, 1999.

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

[37] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[38] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 13–25, 1997.

[39] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent. Nearest neighbor queries. In *Procs. of ACM SIGMOD Conference on Management of Data*, pages 71–79, 1995.

[40] Hiroaki Sakoe. Two-level DP-matching – a dynamic programming-based pattern matching algorithm for connected word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 27(6), 1979.

[41] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Readings in Speech Recognition*, 1990.

[42] Hagit Shatkay and Stanley B. Zdonik. Approximate queries and representations for large data sequences. In *Procs. of International Conf. on Data Engineering*, 1996.

[43] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann, 1996.

[44] Alexander Thomasian, Viitorio Castelli, and Chung-Sheng Li. Clustering and singular value decomposition for approximate indexing in high dimensional spaces. In *Procs. of Conf. on Information and Knowledge Management*, 1998.

[45] Karl Weierstrass. *Mathematische Werke, Volume II.* Mayer & Muller, Berlin, 1895.

[46] Daniel Wu, Divyakant Agrawal, Amr El Abbadi, Ambuj Singh, and Terrence R. Smith. Efficient retrieval for browsing large image databases. In *Procs. of Conf. on Information and Knowledge Management*, 1996.

[47] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Procs. of International Conference on Data Engineering*, 1998.