Similarity Searching in Sequence Databases under Time Warping

Wong, Siu Fung

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Department of Computer Science & Engineering

December, 2003

Similarity Searching in Sequence Databases under Time Warping

submitted by

Wong, Siu Fung

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Finding similar patterns among time series data is very important, as many data are in the form of time series. Most of the research work uses Euclidean distance as their similarity metric. However, Dynamic Time Warping (DTW) distance is more robust than Euclidean distance in many situations, where sequences may be of different lengths or their patterns are out of phase in the time axis. Unfortunately, DTW does not satisfy the triangle inequality, so that spatial indexing techniques cannot be applied directly. We generalized an lower-bound technique for DTW to handle subsequence matching by two different indexing strategies. These methods takes full advantage of "sliding window" approach and R-tree. The first method uses a sliding window to extract multidimensional points for indexing. The second method, however, is to store multidimensional rectangles instead of points in the indexing structure.

Although it has been realized that Dynamic Time Warping (DTW) distance is more robust than Euclidean distance in the time series domain, DTW still cannot reflect users' subjective preferences, as it is a "fixed" distance metric. The perception of similarity may vary from user to user and from scope

to scope. Relevance Feedback, a technique proposed to "learn" user's requirement, is used to deal with this issue. We introduce a relevance feedback strategy, which is based on the DTW metric, to capture users' preferences. Instead of using a fixed warping path constraint, we suggest to use a reformable path constraint together with weighting factors embedded in the distance calculation. By refining the constraint and weights according to the user's feedback, his or her subjective perception can be captured during the retrieval process.

基於時間伸縮的序列數據的相似性查詢

作者　汪紹鋒

香港中文大學　二零零三年十二月

摘要

由於有很多數據都是以時間序列的形式表達，在時間序列數據中搜尋相似形狀是很重要的。大部份的研究工作都是以歐幾里德距離作為它們的相似公制。然而，在很多情況下動態時間伸縮距離比歐幾里德距離更健全，例如當序列的長度不一，或是大家的形狀在時間軸上出現相位差。遺憾地，動態時間伸縮距離並不符合三角不等式。因此，任何空間索引技術均不能被應用。我們由動態時間伸縮距離的下界技術歸納出兩種不同的索引策略，用以處理子序列匹配。這兩個策略利用了滑動窗方法和 R-樹。第一個策略使用滑動窗取出多維點，再以多維點建立索引。然而，第二個策略是存放多維矩形於索引之中，以之代替多維點。

雖然，大家意識到在時間序列的範疇之中，動態時間伸縮距離比歐幾里德距離更健全，但是動態時間伸縮距離還是不反映使用者的喜好，因為它是一種「固定」的距離公制。不同的使用者和不同的領域對相似的觀念都有不同。相關回饋是一種「學習」使用者要求的技巧，透過這種技巧可以應付上述的問題。我們根據動態時間伸縮公制，提出一種相關回饋的策略，從而獲得使用者的喜好。我們不使用固定的伸縮途徑限制，反而提出一種可更改的伸縮途徑限制，同時把加權因素嵌進距離計算之中。在檢索的過程中，我們透過使用者的回饋去改進伸縮途徑限制和加權。那麼，便能獲得使用者的主觀看法。

To my parents for their love and care

Acknowledgment

First of all, I would like to give my heartfelt thanks to Professor Man Hon Wong, my supervisor, for his systematic guidance, endless support and plentiful encouragement throughout my study. I am very grateful for his patience in improving my writing and presentation skills, which benefit the rest of my life. He taught me how to do research and gave insightful advice to my research. Without his guidance, this dissertation could not be done.

Furthermore, I would like to thank Professor Ada Fu and Professor Fung Yu Young, who are my marks, for their invaluable suggestions to my thesis.

Last but not least, I would like to express my gratitude to my colleagues. They are Mr. Ka Cheung Sia, Mr. Wai Man Szeto, Mr. Wing Ho Shum, Mr. Mi Zhou, Mr. Yat Chiu Law, Mr. Chi Wing Wong, Miss Ming Fei Jiang, Mr. Lok Hang Lee, Mr. Shing Ip Wong, Mr. Yung Hang Lau, Mr. Chi Hang Wong, Miss Ping Yan and Mr. Kut Fai Fong. They give me a lot of support and happiness in these two years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A time series $X = \langle x_1, x_2, \ldots x_n \rangle$ is a sequence of real numbers, where each number x_i represents a value at a point of time. In the real world, data always appear in the form of time series, e.g. temperature, atmosphere, stock prices, experimental results ... etc. Therefore, studying patterns of time series is very important in many areas. For example, in a financial application, we may need to find out some specific patterns in a daily stock price sequence, in order to predict the future trend. Extensive research has been done on similarity searching of time series [2, 15, 29, 11, 18, 9, 54, 21, 30, 19, 28, 49, 36, 22, 25, 48, 58, 37]

Most of the research work adopts Euclidean distance as the metric for sequence similarity [2, 15, 29, 11, 9, 54, 21]. Given two sequences $X =< x_1, x_2, \ldots, x_n >$ and $Y =< y_1, y_2, \ldots, y_n >$, the Euclidean distance is defined as follows:

$$D_2(X,Y) = \sqrt[2]{\sum_{i=1}^{n}(x_i - y_j)^2} \tag{1.1}$$

Dozens of papers proposed different modifications based on Euclidean distance [11, 3, 42], dimensional reduction [48, 55, 54] or transformation techniques [4, 40, 3].

In particular, a technique called Dynamic Time Warping (DTW), which is actually a well known technique being used in speech recognition [41], is proposed in [8]. DTW enables matching similar sequences which are out of phase, or even the sequences are of different lengths. Under dynamic time warping, two sequences can be stretched along the time axis. For example, given two sequences $S =< 4, 5, 6, 8, 9 >$ and $Q =< 3, 4, 7, 8, 9, 7 >$ of different lengths, we can stretch the sequences to $S' =< 4, 5, 6, \underline{6}, 8, 9, \underline{9} >$ and $Q' =< 3, \underline{3}, 4, 7, 8, 9, 7 >$, and measure the distance between S' and Q'. In such case, we say that the sequences S and Q "align" with each other. Apparently, there exists more than one way to "align" S and Q. The minimum distance among all possible alignments between sequences S and Q is defined as time warping distance, D_{tw}. Figure 1.1 delineates the difference between Euclidean distance and time warping distance.

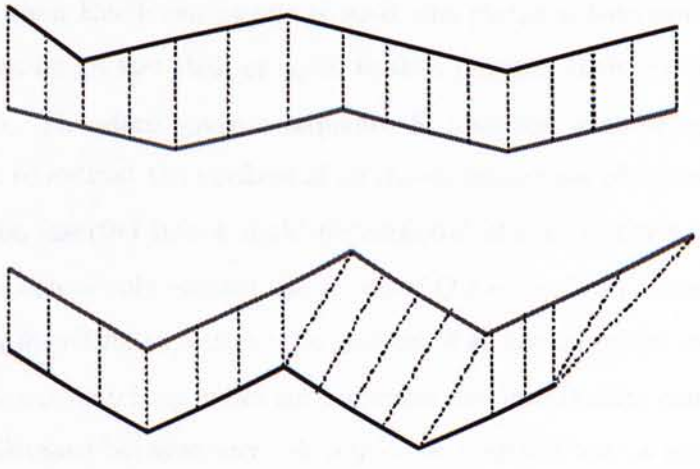

Figure 1.1: The first example is the alignment under Euclidean distance, and the second one is the alignment under DTW.

Searching the sequences sequentially can be slow. Obviously, the query process can be speeded up, if an indexing technique is used. When Euclidean

distance is used as the similarity measure, a sequence of length n can be transformed into a k-dimensional point, and the point can be indexed with any multi-dimensional indexing structure, such as the R-tree [17] or the R*-tree [6]. Similarly, a query sequence can also be transformed into a k-dimensional point, and then a range query search can be performed on the indexing structure. Unfortunately, DTW does not satisfy the triangle inequality, the above method cannot be applied directly.

Nevertheless, a great deal of work is proposed to perform efficient searching and indexing on DTW [57, 33, 27, 34, 26, 23, 35]. In [57], the authors introduce two techniques to speed up the searching process, the first one is FastMap, which results in false dismissals; the second one is lower bound technique. A lower bound function, which satisfies the triangle inequality, is defined by means of lower bound technique. This technique is further used in [27] and [23], in which, DTW can be indexed by any spatial indexing structure [17, 6], but the above approaches can only accommodate to whole sequence matching. However, subsequence matching is different from whole sequence matching. Note that, when Euclidean metric is used, the distance between any two sequences must be greater than or equal to that between their prefixes with the same length. Therefore, given a sequence S, a sliding window approach [15] can be used to extract the prefixes of all the subsequences of S, and then each prefix can be inserted into a multi-dimensional indexing structure. Given a query Q, we can simply extract the prefix of Q and perform a range query on the indexing structure to retrieve the prefixes of all the potential subsequences.

Subsequence matching under time warping distance is more complicated, as the DTW distance between any two sequences may not have a simple relation between their prefixes with the same length. Therefore, the above sliding window technique cannot be applied directly.

For subsequence matching, authors in [35] suggest a segment-based approach, but this approach again may cause false dismissals. Another indexing

method based on suffix tree is proposed in [33], which guarantees no false dismissals. Unfortunately, the index size of suffix tree is extremely large even for a small data set. As an extension of [27], the authors of [34] proposed to use the prefix querying technique together with the lower bound function suggested in [27] to handle subsequence matching. However, as pointed out in [23], this "lower bound is very loose, and many false alarms are generated".

In this dissertation, we are to utilize a better lower bound technique for subsequence matching under time warping distance. Based on the proposed approachs, an R-tree can be used to index the subsequences. Hence, the space overhead will be much smaller than that of the suffix tree approach.

There is less awareness on the difficulty to formulate a well designed query without detailed knowledge of the collection of the database in the time series domain. Moreover, the similarity between two sequences is subjective for different users and problems [39, 24]. For example, different users may have different preferences when choosing the sequences as delineated in Figure 1.2. One user may desire "shifting" the peak, while the others may desire the "stretch" one. Therefore, those "fixed" distance measures are not adaptive.

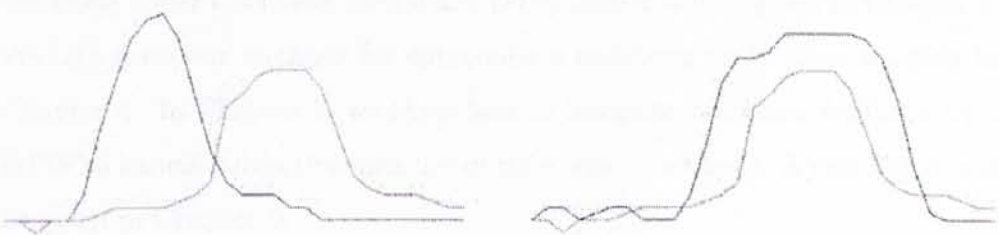

Figure 1.2: Some users may prefer the left result, while some may prefer the right one.

Unlike the time series domain, much effort has been done on this issue in the text and multimedia domains [5]. Relevance feedback [59, 12, 52, 31, 16, 1, 10] is a technique proposed to overcome these difficulties in the text and image domains [43, 44, 46]. The aim of relevance feedback is to capture users' need

during the retrieval process.

The authors in [24] introduce a method, which uses relevance feedback along with an approximation of Euclidean distance, to handle the subjective requirement of a user. However, Euclidean distance is a very brittle measure [23], and it has been realized that Dynamic Time Warping(DTW) distance is more robust and flexible [33, 56, 23, 35, 27, 8, 26] than it. For example, DTW allows matching similar patterns that are out of phase or of different lengths.

Nevertheless, DTW is still a "fixed" distance measure and is unable to reflect a user's requirement. Therefore, we introduce a relevance feedback strategy, which is based on the DTW metric, to capture users' preferences. Instead of using a fixed warping path constraint, we suggest to use a reformable path constraint together with weighting factors embedded in the distance calculation. By refining the constraint and weights according to the user's feedback, his or her subjective perception can be captured during the retrieval process.

The rest of this dissertation is organized as follows. In Chapter 2, we will briefly explain some basic techniques used in time series similarity searching, e.g. DTW, indexing. A literature review on different approaches for time series searching under Euclidean metric and DTW metric will be given in Chapter 3. We introduce our methods for subsequence matching under time warping in Chapter 4. In Chapter 5, we show how to integrate relevance feedback with DTW to handle subjective measure of time series searching. A conclusion will be given in Chapter 6.

Chapter 2

Preliminary

In this chapter, several techniques that used in time series searching are reviewed briefly. Firstly, we will explain the idea of *dynamic time warping*. Secondly, some spatial indexing techniques are studied. Finally, a skill called relevance feedback, which is used to learn users' preferences, is introduced.

2.1 Dynamic Time Warping (DTW)

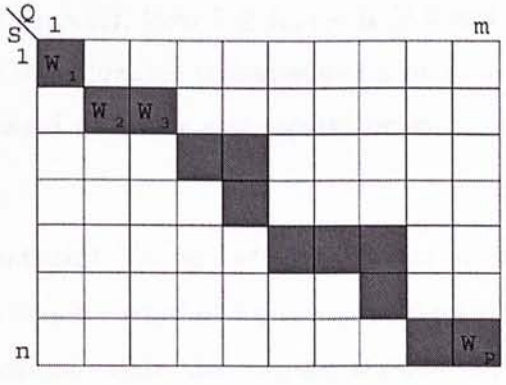

Figure 2.1: A warping path in an m-by-n grid

The idea of *dynamic time warping* is as follows. Given two sequences S and Q of lengths n and m respectively, where:

$$S =< s_1, s_2, \ldots, s_{n-1}, s_n >$$

$$Q =< q_1, q_2, \ldots, q_{m-1}, q_m >$$

We can develop an m-by-n grid, as illustrated in Figure 2.1. Each grid element, (i, j), represents an alignment between points s_i and q_j. A *warping path* W is a sequence of grid elements that define an alignment between S and Q.

$$W = (i_1, j_1), (i_2, j_2), \ldots, (i_p, j_p) \quad max(n, m) \leq p < m + n - 1 \quad (2.1)$$

,where (i_k, j_k) corresponds to the k^{th} grid element in the warping path. For example, (i_3, j_3) in Figure 2.1 represents the grid element $(2, 3)$, which implies that s_2 is aligned with q_3. For practical reasons, several types of constraints, which concern the warping path, are introduced in prevalent research work [41, 8, 50].

End point constraints The warping path should start at (1,1) and end at (n,m).

Monotonicity and Continuity Given two grid elements in a warping path, (i_k, j_k) and (i_{k+1}, j_{k+1}), then $1 \geq i_{k+1} - i_k \geq 0$ and $1 \geq j_{k+1} - j_k \geq 0$. This restricts the allowable transitions of a node to adjacent elements, which are located at either east, south, or south-east with respect to Figure 2.1.

Global Path Constraint The global path constraint defines the region of grid elements that are searched for the optimal warping path. The warping path is limited within the warping window [8], which is known as Sakoe-Chiba Band. For example, the grey area in Figure 2.2 refers to such a band. The constraint can be defined as follows:

$$\forall (i_k, j_k) \in W, \quad i_k - r \leq j_k \leq i_k + r, \quad (2.2)$$

where r is the width of the warping window. For example, in Figure 2.2,

$r = 2$.

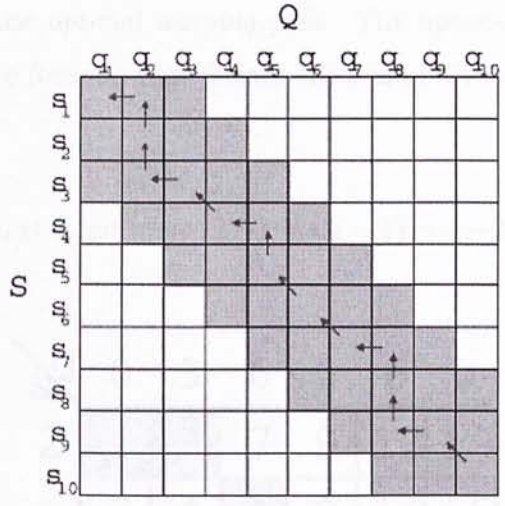

Figure 2.2: The warping path is restricted in the grey area (the warping window)

After aligning the sequences S and Q, their similarity can be measured by the cumulative distance of the warping path between them. Each element in the warping path is associated with a distance, e.g. $d(i_k, j_k) = |s_{i_k} - q_{j_k}|$. The cumulative distance of a warping path, e.g. $W = (i_1, j_1), (i_2, j_2), \ldots, (i_p, j_p)$, is defined as follows:

$$D_c(W) = \sum_{k=1}^{p} d(i_k, j_k) \tag{2.3}$$

There are possibly many warping paths. Among all the potential warping paths, we can always choose an *optimal warping path* such that its cumulative distance, D_c, is minimum. The corresponding distance is defined as the *time warping distance*, D_{tw}.

$$D_{tw}(S, Q) = \min_{\forall W}\{D_c(W)\} \tag{2.4}$$

As there are many warping paths, searching through all of them is computational expensive. Therefore, dynamic programming approach [57, 41] is proposed to find the *optimal warping path.* The approach is based on the following recurrence formula that defines the cumulative distance, $\gamma(i, j)$, for each grid element.

$$\gamma(i, j) = d(i, j) + \min\{\gamma(i-1, j), \gamma(i, i-1), \gamma(i-1, j-1)\} \tag{2.5}$$

S \ Q	0	3	6	0	6	0
2	2	3	7	9	13	15
5	7	4	4	9	10	15
2	9	5	8	6	10	12
5	14	7	6	11	7	12
2	16	8	10	8	11	9

Figure 2.3: A cumulative distance matrix for sequences Q and S

By applying the dynamic programming algorithm, we can construct a cumulative distance matrix as shown in Figure 2.3, which demonstrates such an algorithm by sequences $Q =< 0, 3, 6, 0, 6, 0 >$ and $S =< 2, 5, 2, 5, 2 >$. Each value in the cell represents the cumulative distance of that cell. The cumulative distance of a cell is the sum of the distance associated with it and the minimum of the cumulative distances of its neighboring cells. For example:

$$\gamma(3, 4) = d(3, 4) + \min\{\gamma(2, 4), \gamma(3, 3), \gamma(2, 3)\} = 6$$

After filling up the table, the optimal warping path can be found by tracing backward from the lower right corner, e.g. $(5, 6)$, towards the upper left corner,

e.g. $(1, 1)$. At each cell, we can choose the previous neighboring cell with minimum cumulative distance.

2.2 Spatial Indexing

Spatial indexing technique is widely used to speed up the searching process in sequence databases. In general, spatial indexing methods are used to store feature vectors of an object in a multi-dimensional space. R-tree [17], which is the foundation of spatial indexing technique, has been used in extensive research work. An R-tree is a height-balanced tree that use tuples to represent spatial objects. A leaf node in R-tree contains entries of the form (I,tuple-identifier), where tuple-identifier refers to a tuple representing an object and I is an n-dimensional rectangle which is a minimum bounding box of an indexed spatial object. I $= (I_0, I_1, \ldots, I_{n-1})$ where n is the number of dimensions of the object and I_i is an interval [a,b], which expresses the magnitude of the object along dimension i. Non-leaf node contains entries of the form (I, PTR_i) where PTR_i is a pointer pointing to the ith child node and I is the minimum bounding box that covers all the rectangles in the lower node's entries. Figure 2.4 shows the structure of a simple example of an R-tree and it's indexing mechanism.

Besides the R-tree, there are several spatial indexing methods, such as the R-tree variants [6, 47], X-tree [7] and SR-tree [20]. While the R*tree [6] uses the same structure of the R-tree [17], it outperforms the traditional one by reducing area, margin, and overlap of the data rectangles. The authors also proposed to use reinsertion to decrease the overlapping and splitting of the rectangles. Another variation of R-tree is the R^+-tree [47], in which, the authors suggests to obtain "zero overlap" among non-leaf node by decomposing the space into subregions when a node overflows. Both the X-tree [7] and SR-tree [20] use new indexing structures, which are based on that of the R-tree. The X-tree uses extendable directory nodes(supernodes) to avoid underfill during the process of

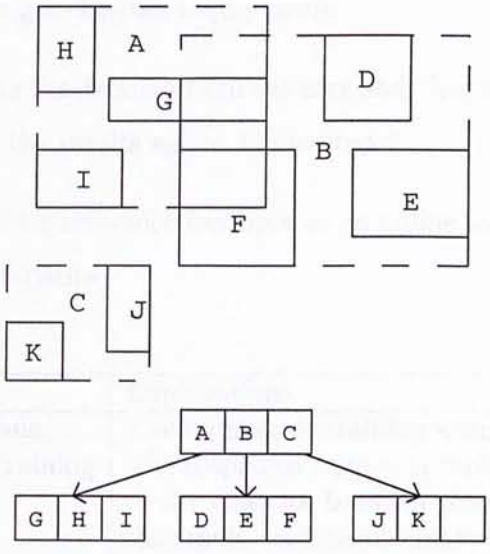

Figure 2.4: An Example of R-tree in a multi-dimensional space

node splitting. The SR-tree defines a region by the intersection of a bounding sphere and a bounding rectangle in order to enhance the disjointness between different regions.

2.3 Relevance Feedback

Relevance feedback is a powerful framework, which is initially designed for text retrieval systems and used in CBIR(Content-based image retrieval) afterward. It has been widely applied in various image retrieval systems [53, 51, 44, 32, 45].

A typical relevance feedback algorithm used in content-based image retrieval contains the following steps and characteristics as stated in [59]. In general, the feedback steps are as follows:

1. The user provides the initial query by sketch or example of keywords. Then the computer or system displays(output) the preliminary retrieval results.

2. The user decides the degree of relevance or irrelevance of the displayed

results according to his/her requirement.

3. The system gets the decision from the user and "learn" the requirements. Try to display the results again. Go to step 2.

We can also consider relevance feedback as an online learning process with the following characteristics:

Characteristics	Explanations
Small sample issue	The number of training samples is small.
Asymmetry in training sample	The displayed output is ranked according to the distance between the query and the result, instead of binary decision (i.e. relevant or irrelevant), and only the top-n results are displayed.
Real time requirement	The algorithm should be efficient and prevent massive computation, as the user is interacting with system in realtime.

Table 2.1: Characteristics of relevance feedback

Chapter 3

Literature Review

In this chapter, we discuss two main streams of research work of sequences searching. We first discuss different approaches of sequences searching under Euclidean metric. Secondly, sequences searching approaches under time warping metric are discussed.

3.1 Searching Sequences under Euclidean Metric

Various approaches, which are based on Euclidean distance, have been proposed for time series searching. In [2], the authors propose an efficient method, F-index method, for processing similarity queries. The time series are firstly transformed from time domain to frequency domain using Discrete Fourier Transform (DFT), and only the first few coefficients will be used. By using the first k coefficients, a time series is mapped to a k-dimensional space that has lower dimensions. The k-dimensional points are then indexed by the R*-tree. For a similarity query, the query sequence is also mapped to a k-dimensional space. Then the R*-tree indexing is being used to extract the feature points for which the Euclidean distance between the query and it is within the given threshold. The F-index method is enhanced in [15] to handle subsequence queies and another approach, ST-index, is proposed. The method

first defines a sliding window of size w and slides it over the time series. Then a set of trails are extracted. Each trail will be transformed by DFT to the frequency domain. Only the first few features (coefficients) are kept. Every trails are further divided into sub-trails and each of them is represented by its minimum bounding hyper-rectangle (MBR) in the R*-tree afterward. For the query process, if the query has the shortest allowable length (e.g. length = w), the sub-trails that intersect the query region are retrieved. If the query is longer than w, a searching algorithm called "MultiPiece" is proposed. The query is split into p pieces, and each piece is of length w. The final result is obtained by merging the results of the sub-queries.

In [14], the authors propose to use Singular Value Decomposition (SVD) for dimension reduction, and only the most significant components are kept. The authors point out that performing SVD on the given data results in small average error and small overhead on disk usage. Furthermore, the authors formulate an enhancement of the SVD algorithm, called SVDD, which can handle some gross errors that SVD cannot handle.

Besides DFT and SVD, [9] proposes to use Discrete Wavelet Transform (DWT) instead of DFT. The advantage of DWT over DFT is that DWT is multi-resolution representation and has the time-frequency localization property. The authors also give a new definition of similarity between two time series called "v-shift similar". The main idea of this similarity definition is to eliminate the effect of vertical offset on the Euclidean distance.

Another definition of similarity that use scaling and shifting are proposed in [11]. The authors regard a time series as a multi-dimensional position vector in $\Re^n$. So that, any scaling and shifting operation on the time series can be regarded as vector multiplication and vector addition respectively. If a data sequence can be scaled and shifted such that the distance between that and the query sequence within a threshold, then the two sequences are said to be similar. The problem is then reduced to a geometrical calculation. The

similarity between two sequences can be easily found by computing the shortest Euclidean distance between a scaled vector (the query) and a shifted vector (the time sequence). The authors propose to use an R-tree to index the data sequence. To facilitate the indexing structure, the shifted vector needs to perform an SE-transformation, and then store the resulting vector (point) into the MBR of the R-tree for searching.

Shifting transformation is also considered in [29], in which, the authors suggest a new definition on similar sequences. Two sequences are similar if the Euclidean distance between them is smaller than a given threshold ϵ, and the distance between each data pair is within another tolerance δ. To facilitate the second criterion, the authors propose to obtain the minimum attainable distance by shifting the sequences vertically. Moreover, the data sequences are projected onto a hyper-plane, and signatures are built for searching.

Furthermore, [13] proposes to use normalization, which is a widely adopted technique, to handle scaling and shifting on time series. Each element of a sequence X, say x_i, is transformed according to the following formula,

$$\frac{x_i - mean(X)}{std(X)}$$

where $mean(S)$ is the mean of S, and $std(X)$ is the standard deviation of S. After the transformation, the normalized sequences can be directly used in similarity searching.

In addition to scaling and shifting, [42] considers other transformations, moving average and time warping. They first discuss a wide range of examples and show that their proposed transformations can handle such cases. They propose an index strategy based on the framework of [2] to speed up the searching process. Firstly, an index I is built. For each query, a transformation is given. During the query process, a new index I' is constructed depending

on the given transformation and the query searches against I'.

[38] also considers various transformations other than scaling and shifting. The transformations are: Shifting, Uniform Amplitude Scaling, Uniform Time Scaling, Uniform Bi-scaling, Time Warping and Non-uniform Amplitude Scaling. Besides, the authors present the Landmark Model, which represents a time series by a set of turning points. Their idea is based on the fact that "people recognize patterns in charts by identifying important points". They also introduce a smoothing method, Minimal Distance/Percentage Principle (MDPP), for removing noise from the raw data without eliminating the peaks and bottoms from the time series pattern. Finally, they show by experiments that their Landmark Indexing is considerably fast.

The lattice structure proposed in [54] also make use of the concept of turning points. The authors propose to approximate the time series using all significant feature instead of involving all the data points. They measure the significance of a point, x_i, by the importance defined as below:

$$Importance(x_i) = \sum_{j=1}^{N}(|x_{i-j} - x_i| + |x_{i+j} - x_i|)$$

Each point in a time series is stored in appropriate layer of the lattice structure according to its importance. In other words, the control points in each layer form an approximation pattern of the time series, and different layers of the lattice structure have different levels of detail. To perform searching, a user needs to provide the query together with his requirements of level of detail.

In [24], the authors propose a method, which is based on the feedback of the user, to retrieve time series data. The similarity metric used in this paper is an approximation of Euclidean distance together with weights associated with the sequences. During the query process, the user gives different ratings to the sequences, and according to the user's feedback, the preference of the user

can be captured. Hence, the query can be refined accordingly. The authors also suggest using a profile to represent user's subjective similarity perception, which is the sensitivity to the distortions, such as offset translation, amplitude scaling, linear drift and discontinuities. Then this profile can be used in the data retrieval process.

3.2 Searching Sequences under Dynamic Time Warping Metric

Computing the time warping distance is time consuming, as the dynamic programming algorithm is quadratic on the length of the sequences. Various methods have been proposed to speed up the searching process.

An earlier work [57] proposes two techniques to achieve this goal. The first one is FastMap technique. Given sequences with length n, the authors suggest to map the sequences into $k - d$ points at first. Therefore, the calculation will be much faster, as k is much smaller than n. Another one is lower bound technique. The idea is to use another cheaply computed distance, which underestimates the time warping distance. This technique is also used in [27] and [23], in which, an index can be used under the lower bound distance measure. Here, we give a brief explanation for each lower bound.

The intuitive idea of the lower bound function introduced in [57] is shown in Figure 3.1. This lower bound distance is defined as the squared difference of the shaded regions. This boundary is based on the following observations: An element of a sequence X, say x_i, can align with any element of another sequence Y, say y_i[1], but,

$$d(x_i, max(Y)) \leq d(x_i, y_j) \quad \forall x_i > max(Y)$$

[1]The role of X and Y can be switched

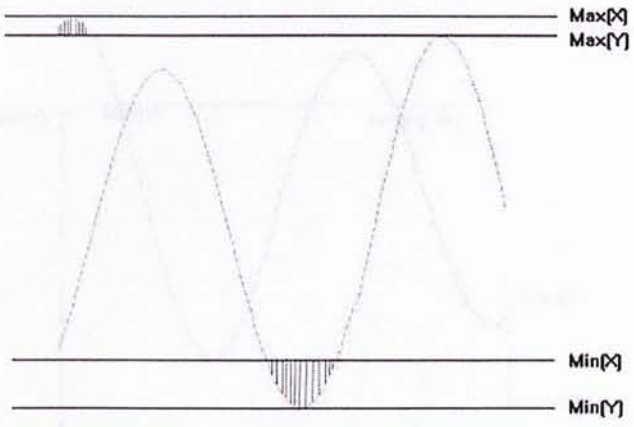

Figure 3.1: An illustration of lower bound proposed in [57]

$$d(min(X), y_i) \leq d(x_i, y_j) \quad \forall y_j < min(X)$$

Figure 3.2 depicts the intuitive idea of the lower bound function introduced in [27]. This lower bound distance does not accumulate the time warping distance between the elements of the sequences. It only considers the first, last, maximum, and minimum values of the two sequences. It takes the advantage of the fact that the time warping distance must be greater than the maximum of the corresponding distance of the four features.

The lower bound function introduced in [23] can be illustrated in Figure 3.3. An envelope, which specifies how much a sequence can be shrunk and stretched, is defined to enclose one of the sequences. The lower bound distance is the squared difference of the shared regions, which are the parts of another sequence not falling into the envelope.

In contrast with the above work, which concentrates on whole sequence matching, the authors in [35] suggest a method to handle subsequence matching. The sequences are divided into piece-wise and the distance between any

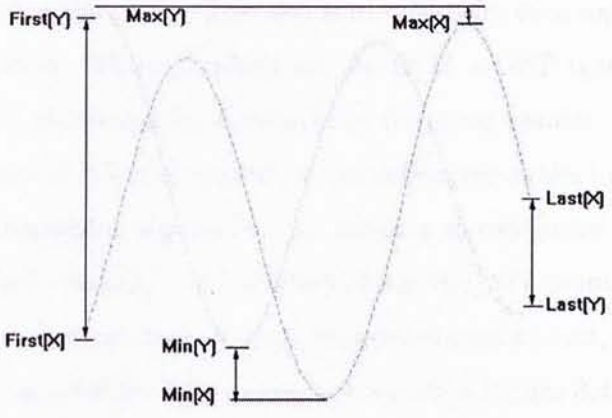

Figure 3.2: An illustration of lower bound proposed in [27]

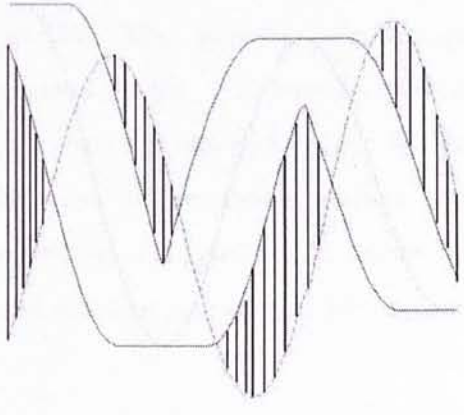

Figure 3.3: An illustration of lower bound proposed in [23]

two sequences are the sum of the time warping distance between each segment pair. A lower bound function together with a suffix tree indexing method have been proposed in the paper. The idea is to categorize each segment and denote them by symbols. Those symbols are stored in a GST (general suffix tree), which actually stores similar sequences in the same branch.

The authors of [33] also suggest to use suffix tree as the indexing structure. Instead of categorizing segments, they propose to categorize each point of the sequences. Each category is a boundary (Max,Min) of a point. When calculate the cumulative distance matrix, the distance between a point, p, and a category is the minimum possible distance as delineated in Figure 3.4.

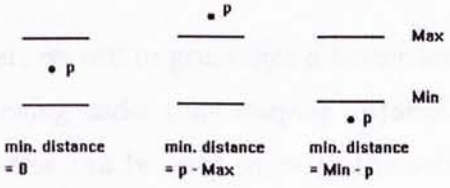

Figure 3.4: An illustration of lower bound proposed in [33]

As the memory usage of suffix tree is huge, another approach using spatial index is presented in [34]. They introduce the concept of maximum warping ratio and minimum query length. The $maxWarpRatio$ stands for how many times an element of a sequence can be repeated by time warping. Firstly, an R-tree index is built based on the sliding window approach. Secondly, a set of subsequences are determined based on the $maxWarpRatio$ and $minQLen$. Then the lower bound function proposed in [27] are used to perform searching.

Chapter 4

Subsequence Matching under Time Warping

In this chapter, we are to generalize a better lower bound technique for subsequence matching under time warping distance. Based on the proposed approach, an R-tree can be used to index the subsequences. Hence, the space overhead will be much smaller than that of the suffix tree approach. In general, similarity queries can be divided into two main categories:

(a) Whole sequence matching: Given a query sequence of length n, and a time series database, we want to find out all the time series from the database that are similar to the query sequence; i.e., given a set of time series $S = \{S_1, ..., S_m\}$ and a query sequence $Q = \langle q_1, q_2, \ldots, q_n \rangle$, we want to ask if there is any time series S_i, where $1 < i < m$, such that S_i and Q are similar.

(b) Subsequence matching: Given a query sequence of length n, and a time series database, we want to retrieve all subsequences from the database that are similar to the query sequence; i.e., given a set of time series $S = \{S_1, ..., S_m\}$ and a query sequence $Q = \langle q_1, q_2, \ldots, q_n \rangle$, we want to ask if there is any subsequences S' within any S_i, where $1 < i < m$, such that S' and Q are similar.

4.1 Subsequence Matching

In this section, we introduce the problem of subsequence matching. The main symbols used in this chapter and their definitions are summarized in Table 4.1. The problem is formally defined as follows:

Symbol	Definition
Q	A query sequences.
S_i	The i-th data sequence
$S[i:j]$	Subsequence of S, begin at position i and end at j.
$Len(S)$	The length of a sequence S.
l	The width of the sliding window.
r	The width of the warping window.
$D_{tw}(S,Q)$	The time warping distance between sequences S and Q.
$D_{tw-lb}(S,Q)$	The lower bound distance between sequences S and Q.

Table 4.1: Summary of Symbols and Definitions

(1) Given a set of time series $\{S_1, S_2, \ldots, Sn\}$ of arbitrary lengths.

(2) A query sequence Q together with a threshold ε are given.

(3) Find all the subsequences $S_k[i:j]$, for a given global path constraint, such that $D_{tw}(S_k[i:j], Q) \leq \varepsilon$, where $S_k[i:j]$ and Q may be of different lengths.

4.1.1 Sequential Search

The most straightforward method is sequential search, by which, all the possible subsequences of every sequence will be examined. Consider the following example:

Given a sequence $S =< 1, 2, 5, 2, 5, 3, 10 >$ and a query $Q =< 0, 3, 6, 0, 6 >$, we want to find all the subsequences, $S[i:j]$, of S such that $D_{tw}(S[i:j], Q) \leq 8$. In particular, let's consider the subsequence started at the 2^{nd} offset of S, i.e.

$S[2:7] =< 2,5,2,5,3,10 >$. By using the dynamic programming approach, we can construct the matrix as shown in Table 4.2.

S \ Q	0	3	6	0	6
2	2	3	7	9	**13**
5	7	4	4	9	**10**
2	9	5	8	6	**10**
5	14	7	6	11	**7**
3	17	7	9	9	**10**
10	27	14	11	19	**13**

Table 4.2: An example of subsequence matching

From the last column of Table 4.2, we know that $D_{tw}(S[2:2], Q) = 13$, $D_{tw}(S[2:3], Q) = 10$, $D_{tw}(S[2:4], Q) = 10$, $D_{tw}(S[2:5], Q) = 7$, $D_{tw}(S[2:6], Q) = 10$, $D_{tw}(S[2:7], Q) = 13$. Therefore, the subsequence $S[2:5]$ satisfies our requirement and will be included in the answer set. The same procedure has to be used to examine each subsequence at every offset in order to find out all the qualifying answers. We call this method as "Sequential Search".

However, there are two major problems concerning sequential search:

- The I/O cost of examining all the possible subsequences is very high. In the above example, seven subsequences have to be examined.
- The process of dynamic programming is computational costly($O(m * n)$, where n and m are the lengths of the sequences).

4.1.2 Indexing Scheme

Before introducing the indexing scheme of subsequence matching under DTW, we need to review the scheme under the Euclidean distance metric first.

Under the Euclidean metric, the sliding window approach [15] can be used to avoid examining each possible subsequence of a given sequence S. The

approach is described as follows. A sliding window of length l is placed and slid over each offset of S. Then a prefix of length l of each possible subsequence is extracted. Each extracted prefix can be regarded as an l-dimensional point and indexed in a multidimensional indexing structure, e.g. R-tree. Note that the following lemma holds under the Euclidean metric.

Lemma 1 If two sequences S and Q of length n agree within a threshold ϵ, then the subsequences $S[i : j]$ and $Q[i : j]$ agree within the same threshold too, i.e.

$$D(S, Q) \leq \epsilon \Rightarrow D(S[i : j], Q[i : j]) \leq \epsilon \tag{4.1}$$

From Lemma 1, the distance between the prefix of S and the corresponding prefix of Q is a lower bound of that between the whole sequences. Therefore, we can search for all prefixes, which match $Q[1 : l]$ within ϵ, from the index. The results are prefixes of the subsequences that will potentially match the query sequence Q within threshold ϵ. Then, the candidate set will be filtered by postprocessing to obtain the final answers.

Unfortunately, under the time warping metric, Lemma 1 does not hold. Let's consider the example in Section 4.1.1. If the length of the sliding window is 4, the prefix of length 4 of each possible subsequence of S will be extracted and indexed. $Q[1 : 4]$ will be used to perform searching. Consider the subsequence at the 2^{nd} offset of S again. From Table 4.2,

$$\begin{aligned} D_{tw}(S[2 : 5], Q[1 : 4]) &= 11 \\ D_{tw}(S[2 : 6], Q[1 : 5]) &= 10 \end{aligned}$$

Therefore, $D_{tw}(S[2 : 5], Q[1 : 4]) > D_{tw}(S[2 : 6], Q[1 : 5])$, and Lemma 1 does not hold. It is clear that we extract a "wrong" prefix of Q to compare with $S[2 : 5]$. If we use $Q[1 : 4]$ to perform index searching, it will cause false

dismissal. Instead, from Table 4.2, we have the following observations:

$$\begin{aligned} D_{tw}(S[2:5], Q[1:5]) &= 7 \\ D_{tw}(S[2:6], Q[1:5]) &= 10 \\ D_{tw}(S[2:5], Q[1:5]) &< D_{tw}(S[2:6], Q[1:5]) \end{aligned}$$

Hence, $Q[1:5]$ is the "correct" prefix to be used. The remaining problems are:

- We need to find the "correct" prefix of Q to perform searching.
- Time warping distance does not satisfy triangle inequality, so that we cannot perform searching in any spatial index.
- The time complexity for the computation of dynamic programming is high.

To overcome these problems, lower bound technique [27, 23, 57] is used. We first discuss some existing lower bound techniques for whole "sequence matching". Then we will extend the idea for subsequence searching in the rest of the chapter.

4.2 Lower Bound Technique

The computation process of dynamic programming has very high complexity. Performing such process on each data sequence in the database can be slow. Therefore, the lower bound technique is introduced [27, 23, 57]. A new distance measure, D_{tw-lb}, is defined ,which is lower-bounding the actually DTW distance.

4.2.1 Properties of Lower Bound Technique

Algorithm 1 lower-bound-search

```
1: C := {}
2: for each possible subsequence S' in S do
3:    if D_tw-lb(S', Q) ≤ ε then
4:       C ← S'
5:    end if
6: end for
7: for each S' in C do
8:    if D_tw(S', Q) ≤ ε then
9:       return(S')
10:   end if
11: end for
```

Algorithm 1 describes how range query can be performed using D_{tw-lb}. Note that Algorithm 1 does not reduce the number of subsequences to be scanned. Instead, the speed up is contributed by the distance calculation. Therefore, a lower-bound function should have the following properties:

Property 1 (Correctness) It must return all the qualifying subsequences, but may cause false alarms, which can be discarded afterwards using post-processing, i.e.

$$D_{tw}(S, Q) \leq \varepsilon \Rightarrow D_{tw-lb}(S, Q) \leq \varepsilon \tag{4.2}$$

Property 2 (Efficiency) The time complexity for the computation should be low, e.g. O(n).

Property 3 (Tightness) With a relative tight lower bound, the number of candidate results for post-processing can be greatly reduced.

Moreover, if the lower bound function satisfies triangle inequality, it can be further used as the filtering function in indexing search.

4.2.2 Existing Lower Bound Functions

To our best knowledge, two lower bound functions, which guarantee no false dismissals and support indexing, have been proposed to handle "whole sequence" matching.

In [27], the authors propose to extract a 4-tuple feature vector, ⟨ First(S), Last(S), Greatest(S), Smallest(S) ⟩, from each sequence S. The features are the first, last, greatest and smallest elements of a sequence S respectively. Given two sequences S and Q, their lower bound distance, D_{lb-kim}, is defined as follows:

$$D_{lb-kim} = \max \begin{cases} |First(S) - First(Q)| \\ |Last(S) - Last(Q)| \\ |Greatest(S) - Greatest(Q)| \\ |Smallest(S) - Smallest(Q)| \end{cases} \tag{4.3}$$

Another lower bound function is proposed in [23], in which the indexing method is called an "exact indexing" of DTW. For any two sequences S and Q of *the same length* n, for any global path constraints of the form $i-r \leq j \leq i+r$, the lower bound distance, $D_{lb-keogh}$, is defined as follows:

$$\begin{aligned} U_i &= \max\{q_{i-r} \ldots q_{i+r}\} \\ L_i &= \min\{q_{i-r} \ldots q_{i+r}\} \\ D_{lb-keogh} &= \sqrt{\sum_{i=1}^{n} \begin{cases} (s_i - U_i)^2 & if s_i > U_i \\ (s_i - L_i)^2 & if s_i < L_i \\ 0 & otherwise \end{cases}} \end{aligned} \tag{4.4}$$

Here, we want to utilize the lower bound function proposed in [23], which is designed for "whole sequence" matching, to handle subsequence matching. By the deductions and observations in Sections 4.3 and 4.4, we can utilize

the lower bound function in the sliding window approach via two indexing methods.

4.3 Point-Based indexing

In this section, we introduce our first method. The idea is to determine the possible subsequences of the query that we need to examine and form a bounding box (hyper-rectangle) based on those subsequences. Then the hyper-rectangle will be searched against a point-based index.

4.3.1 Lower Bound for subsequences matching

To understand how we can form a bounding box, let's consider the following example.

An example: Given two sequences $S =< 2, 5, 2, 5, 3, 7, 1, 8, 4, 10 >$ and $Q =< 7, 0, 3, 6, 0, 6, 5, 8, 4 >$, we can construct two matrixes as shown in Figure 4.1. The first matrix is the same as the grid illustrated in Figure 2.1 and each value in (i, j) represents the distance between points s_i and q_j, e.g. $d(1, 5) = |s_1 - q_5| = |2 - 0| = 2$. The second matrix is the cumulative distance matrix, which is constructed by dynamic programming. There is an optimal warping path $\{(1, 1), (2, 1), (3, 2), (4, 3), (5, 3), (6, 4), (7, 5), (8, 6), (8, 7), (8, 8), (9, 9)\}$, such that the time warping distance is 18.

In particular, $S[5 : 8]$ must align with one of the subsequences of Q within $Q[3 : 9]$ and the warping window, i.e. $Q[3 : 6]$, $Q[3 : 7]$, $Q[3 : 8]$, $Q[3 : 9]$, $Q[4 : 6]$, $Q[4 : 7]$, $Q[4 : 8]$, $Q[4 : 9]$, $Q[5 : 6]$ ... etc.

According to the above observation, we have the following Lemma.

Lemma 2 If the time warping distance, for any global path constraints, between two sequences S and Q is within a threshold ϵ, then for any given subsequence $S[i : j]$ of length l, there must exist at least one subsequence

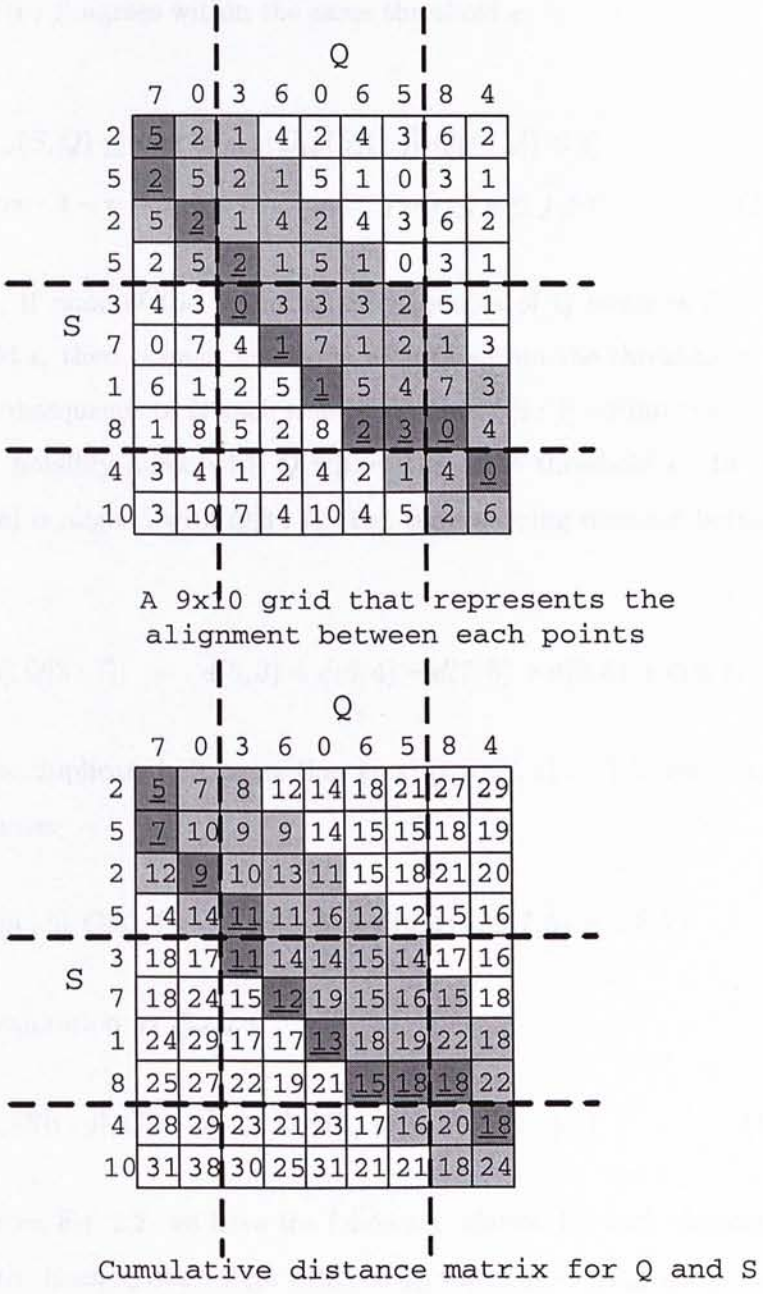

Figure 4.1: An illustration of a 9-by-10 grid and a cumulative distance matrix for Q and S. The warping path is underlined and it is restricted in a warping window with width 2(grey area)

$Q[x:y]$, where $i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, whose time warping distance with $S[i:j]$ agrees within the same threshold ϵ.

$$D_{tw}(S,Q) \leq \epsilon \Rightarrow \exists_{i,j,x,y}(D_{tw}(S[i:j],Q[x:y]) \leq \epsilon)$$
$$where \quad i-r \leq x \leq i+r \quad and \quad j-r \leq y \leq j+r \tag{4.5}$$

In other words, if none of the potential subsequences of Q matches $S[i:j]$ within threshold ϵ, then S must not agree with Q within the threshold ϵ. If there exists a subsequence of Q such that it matches $S[i:j]$ within threshold ϵ, then S may possibly agree with Q within the same threshold ϵ. In this example, $S[5:8]$ is aligned with $Q[3:7]$. The time warping distance between them is:

$$D_{tw}(S[5:8],Q[3:7]) \;=\; d(5,3)+d(6,4)+d(7,5)+d(8,6)+d(8,7)$$

If we remove the duplicated elements (i.e. (i,x),(i,y),(i,z) ...) in every row, e.g. $(8,6)$, we have:

$$D_{tw}(S[5:8],Q[3:7]) \;\geq\; d(5,3)+d(6,4)+d(7,5)+d(8,7)$$

We define this operation as F_{rd}, i.e.

$$D_{tw}(S[i:j],Q[x:y]) \geq F_{rd}(D_{tw}(S[i:j],Q[x:y])) \tag{4.6}$$

Furthermore, from Eq. 2.2, we have the following relation for each element of the warping path. Each s_u must align with one q_v, where $q_v \in \{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\}$. For example, each element of $S[5:8]$ must align with at least one element of

$Q[3:7]$ within the warping window, i.e. s_7 must align with q_5, q_6, or q_7. Therefore, the distance between s_u and q_v must satisfy the following inequality.

$$\begin{aligned} d(u,v) &\geq \min\{d(u, max(u-r,x)) \ldots d(u, min(u+r,y))\} \\ &\geq \min\{|s_u - q_{max(u-r,x)}| \ldots |s_u - q_{min(u+r,y)}|\} \\ &\geq \begin{cases} |s_u - \max\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\}|, \\ if s_u > \max\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} \\ |s_u - \min\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\}|, \\ if s_u < \min\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} \\ 0 \quad otherwise \end{cases} \\ &\equiv d_{lb}(u,v) \end{aligned} \tag{4.7}$$

We define the lower bound for each element as $d_{lb}(u,v)$. From Eq. 4.6 and Eq. 4.7, we derive the following inequality:

$$D_{tw}(S[i:j], Q[x:y]) \geq F_{rd}(\sum_{u=i}^{j} d_{lb}(u,v)) \tag{4.8}$$

Then, we construct a bounding rectangle for subsequences of Q based on the above observations. Given two subsequences $S[i:j]$ and $Q[x:y]$, where $l = Len(S[i:j])$ and $i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, we treat $S[i:j]$ as an l-dimensional point. For any global path constraints, a query bounding hyper-rectangle (referred as QBR hereafter) of l-dimension is defined by two endpoints, BL and BH, of its major diagonal, where $BL = (bl_1, bl_2, \ldots, bl_l)$ and $BH = (bh_1, bh_2, \ldots, bh_l)$ and $bl_i < bh_i$ for $1 < i < l$. An example of QBR is illustrated in Figure 4.2, where

$$\left.\begin{aligned} bl_{u-i+1} &= \min\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} \\ bh_{u-i+1} &= \max\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} \end{aligned}\right\} for \quad u = i \ldots j \tag{4.9}$$

Given a QBR, the *ϵ-enlargement* of the QBR, denoted by ϵ-QBR, is a

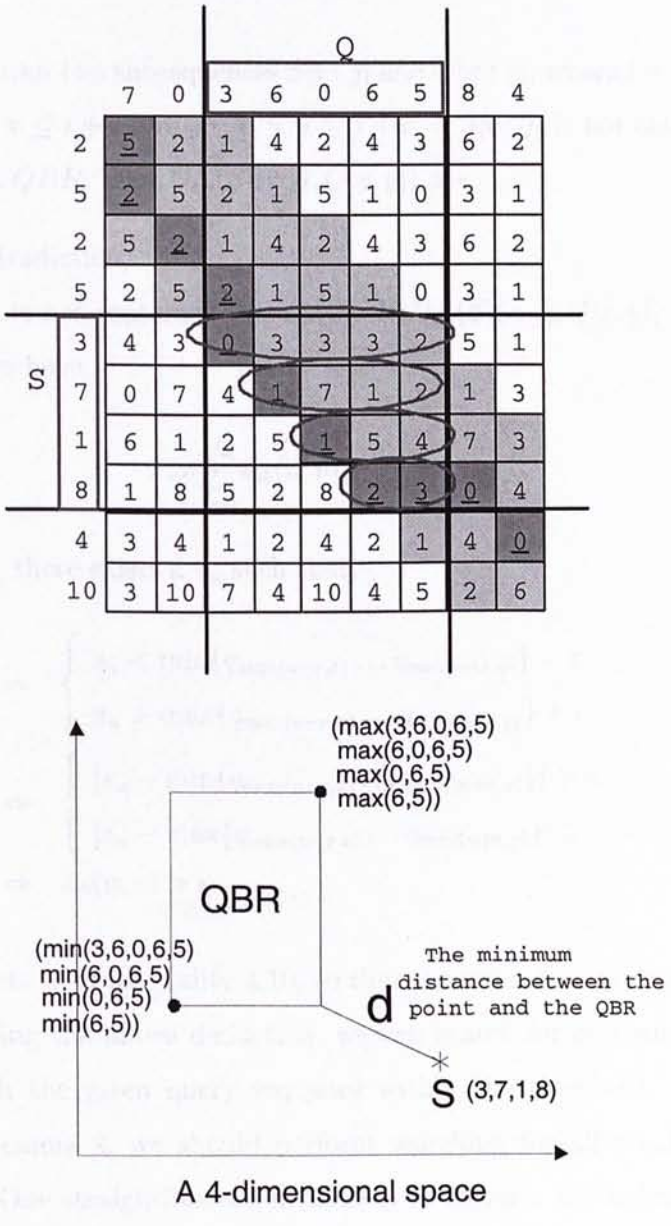

Figure 4.2: An example of the QBR

QBR defined by $BL = (bl_1 - \epsilon, bl_2 - \epsilon, \ldots, bl_l - \epsilon)$ and $BH = (bh_1 + \epsilon, bh_2 + \epsilon, \ldots, bh_l + \epsilon)$. To make use of the ϵ-QBR on searching, the following theorem is needed.

Theorem 1 Given two subsequences $S[i:j]$ and $Q[x:y]$, where $l = Len(S[i:j])$ and $i - r \leq x \leq i + r$ and $j - r \leq y \leq j + r$, if $S[i:j]$ is not contained in the ϵ-QBR of a QBR, then $D_{tw}(S[i:j], Q[x,y]) > \epsilon$.

Proof: By contradiction,
Assume $S[i:j]$ is not contained in ϵ-QBR. If $D_{tw}(S[i:j], Q[x,y]) \leq \epsilon$, then from Eq. 4.8, we have,

$$F_{rd}(\sum_{u=i}^{j} d_{lb}(u,v)) \leq \epsilon \tag{4.10}$$

By assumption, there exists a s_u such that,

$$\begin{aligned}
&\Rightarrow \begin{cases} s_u < \min\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} - \epsilon \\ s_u > \max\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\} + \epsilon \end{cases} \\
&\Rightarrow \begin{cases} |s_u - \min\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\}| > \epsilon \\ |s_u - \max\{q_{max(u-r,x)} \cdots q_{min(u+r,y)}\}| > \epsilon \end{cases} \\
&\Rightarrow \; d_{lb}(u,v) > \epsilon
\end{aligned}$$

which contradicts with inequality 4.10, so that the proof is completed.

By completing the above deduction, we can search for any subsequences that agree with the given query sequence within the threshold. However, according to Lemma 2, we should perform searching for all possible query subsequences. One straightforward method is to traverse the index for every possible ϵ-QBR, but it is not efficient. To reduce the number of indexing search, we can merge all the QBRs into one single QBR. That is to define a minimum bounding box, which contains all the $QBRs$. Refer to Eq. 4.9, as

$i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, we have:

$$u-r = \min \begin{cases} \max\{u-r, i-r\} \\ \max\{u-r, i+1-r\} \\ \vdots \\ \max\{u-r, i+r\} \end{cases} \tag{4.11}$$

$$u+r = \max \begin{cases} \min\{u+r, j-r\} \\ \min\{u+r, j+1-r\} \\ \vdots \\ \min\{u+r, j+r\} \end{cases} \tag{4.12}$$

Therefore, the large QBR is defined as follows:

$$\left.\begin{array}{l} bl_{u-i+1} = \min\{q_{u-r} \ldots q_{u+r}\} \\ bh_{u-i+1} = \max\{q_{u-r} \ldots q_{u+r}\} \end{array}\right\} for \quad u = i \ldots j \tag{4.13}$$

Actually, it is the QBR of the longest possible subsequences. Figure 4.3

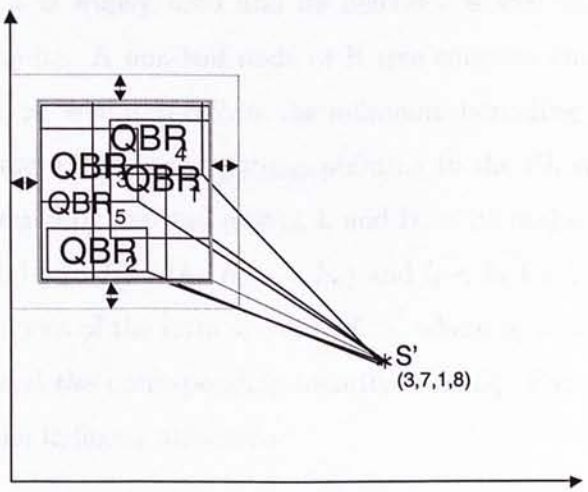

Figure 4.3: An illustration of merging all the QBRs

depicts this idea. By deducing this large QBR and Lemma 2, we generalize the "exact indexing" method [23], to handle subsequence queries of varying

lengths as described in the following section.

4.3.2 Algorithm

In this section, we describe a subsequence searching method that is fast, guarantees no false dismissal and can handle query sequences of arbitrary lengths.

Index Construction

Given a set of data sequences $S = \{S_1, S_2, \ldots, S_m\}$ of different lengths, a sliding window of length l is placed and slid over each offset of all data sequences, i.e. $S_i \in S$. Thus, a set of prefixes of length l will be extracted. The total number of subsequences is $\Sigma(Len(S_i) - l + 1)$. Every prefix with length l will be regarded as an l-dimensional point. To index a set of multi-dimensional points, we can use any spatial indexing technique, such as the R-tree [17], and the R*-tree [6]. In the following, we sketch the idea of indexing by R-tree, which uses hyper-rectangles as its minimum bounding volume. We choose R-tree because it is widely used and its behavior is well understood in the database community. A non-leaf node of R-tree contains entries of the form $< MBR_i, PTR_i >$, where MBR_i is the minimum bounding rectangle of the ith child node and PTR_i is the pointer pointing to the ith child node. Each MBR is represented by two end points, L and H, of its major diagonal, where $L = (l_1, l_2, \ldots, l_n)$ and $H = (h_1, h_2, \ldots, h_n)$ and $l_i < h_i$ for $1 \leq i \leq n$. A leaf node contains entries of the form $< SID'_i, S'_i >$, where S'_i is one of the prefixes of sequence S_i and the corresponding identity is SID'_i. Figure 4.4 illustrates an example of the indexing structure.

Searching subsequences of arbitrary lengths

Here we show how to search for subsequences that match the query sequence Q within threshold ϵ. We call this method 'PB-SSM'. If the length of the sliding

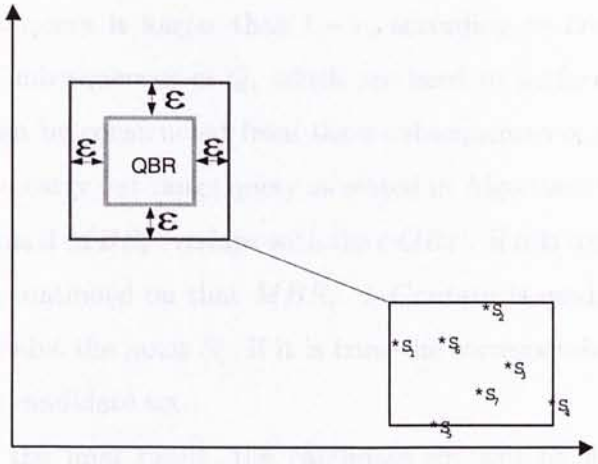

Figure 4.4: An illustration of the indexing structure.

window is l, the minimum query length is $l - r$. The proof is as follows:

Proof: Assume that we have a query Q, such that $Len(Q) < l - r$. If there exist a subsequence that matches Q within threshold ϵ. By Lemma 2, we have:

$$D_{tw}(S, Q) \leq \epsilon \;\Rightarrow\; (D_{tw}(S[1:l], Q[1:y]) \leq \epsilon)$$
$$where \quad l - r \leq \quad y \quad \leq l + r$$

which contradicts the assumption that $y < l - r$.

Algorithm 2 RangeQuery(T,ϵ-QBR)

```
1: C := {}
2: if T is not a leaf node then
3:    for each child node of T do
4:       if IsOverlap(MBR_i, ε-QBR) then
5:          RangeQuery(PTR_i,ε-QBR)
6:       end if
7:    end for
8: else
9:    if IsContain(S'_i, ε-QBR) then
10:      C ← SID'_i
11:   end if
12: end if
```

If the input query is longer than $l - r$, according to Lemma 2, we can derive a set of subsequences of Q, which are used to perform searching. A large ϵ-QBR can be constructed from these subsequences of Q. The ϵ-QBR are then used to carry out range query as stated in Algorithm 2. *IsOverlap* is used to determine if MBR_i overlaps with the ϵ-QBR. If it is true, the searching process will be continued on that MBR_i. *IsContain* is used to determine if the ϵ-QBR contains the point S'_i. If it is true, the corresponding SID'_i will be included in the candidate set.

To acquire the final result, the candidate set will be filtered by post-processing. For each SID'_i in the candidate set, we retrieve the corresponding subsequence (which has prefix S'_i) from the database, e.g. $S_i[j : k]$. If $D_{tw}(S_i[j : k], Q) \leq \epsilon$, then $S_i[j : k]$ is a qualified answer.

4.4 Rectangle-Based indexing

In contrast with point-based indexing, we propose another indexing structure, which stores hyper-rectangle in the index, in this section. Therefore, the query will become a point instead of a bounding box. Hence, the query volume is independent of the warping window.

4.4.1 Lower Bound for subsequences matching

To understand this method, we use the same scenario stated in Section 4.3.1, but we consider the subsequence $Q[3 : 6]$ this time. From Figure 4.1, $Q[3 : 6]$ must align with one of the subsequences of S within the warping window, i.e. $S[1 : 4]$, $S[1 : 5]$, $S[1 : 6]$, $S[1 : 7]$, $S[1 : 8]$, $S[2 : 4]$, $S[2 : 5]$, $S[2 : 6]$, $S[2 : 7]$, $S[2 : 8]$, $S[3 : 4]$ ...etc. According to the above observation, we have the following Lemma, which is similar to Lemma 2.

Lemma 3 If the time warping distance, for any global path constraints, between two sequences S and Q is within a threshold ϵ, then for any given subsequence $Q[i:j]$ of length l, there must exist at least one subsequence $S[x:y]$, where $i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, whose time warping distance with $Q[i:j]$ agrees within the same threshold ϵ.

$$D_{tw}(S,Q) \leq \epsilon \Rightarrow \exists_{i,j,x,y}(D_{tw}(S[x:y],Q[i:j]) \leq \epsilon) \qquad (4.14)$$
$$where \quad i-r \leq x \leq i+r \quad and \quad j-r \leq y \leq j+r$$

In other words, if none of the potential subsequences of S matches $Q[i:j]$ within threshold ϵ, then Q must not agree with S within the threshold ϵ. If there exists a subsequence of S such that it matches $Q[i:j]$ within threshold ϵ, then Q may possibly agree with S within the same threshold ϵ.

Given two subsequences $S[x:y]$ and $Q[i:j]$, where $l = Len(Q[i:j])$ and $i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, we construct a bounding rectangle for subsequences of S using the deduction stated in Appendix A. For any global path constraints, a data bounding hyper-rectangle (referred as DBR hereafter) of l-dimension is defined by two endpoints of its major diagonal, where $BL = (bl_1, bl_2, \ldots, bl_l)$ and $BH = (bh_1, bh_2, \ldots, bh_l)$ and $bl_i < bh_i$ for $1 < i < l$, as stated in Eq. 4.15. Figure 4.5 illustrates the DBR.

$$\left.\begin{aligned} bl_{v-i+1} &= \min\{s_{max(v-r,x)} \ldots s_{min(v+r,y)}\} \\ bh_{v-i+1} &= \max\{s_{max(v-r,x)} \ldots s_{min(v+r,y)}\} \end{aligned}\right\} for \quad v = i \ldots j \qquad (4.15)$$

The *ϵ-enlargement* of a query Q, denoted by ϵ-Q, is a minimum bounding box defined by $QL = (q_1-\epsilon, q2-\epsilon, \ldots, ql-\epsilon)$ and $QH = (q_1+\epsilon, q_2+\epsilon, \ldots, q_l+\epsilon)$. The relation between the DBR and *ϵ-enlargement* is described below.

Theorem 2 Given two subsequences $S[x:y]$ and $Q[i:j]$, where $l = Len(Q[i:j])$ and $i-r \leq x \leq i+r$ and $j-r \leq y \leq j+r$, if ϵ-Q does not overlapped

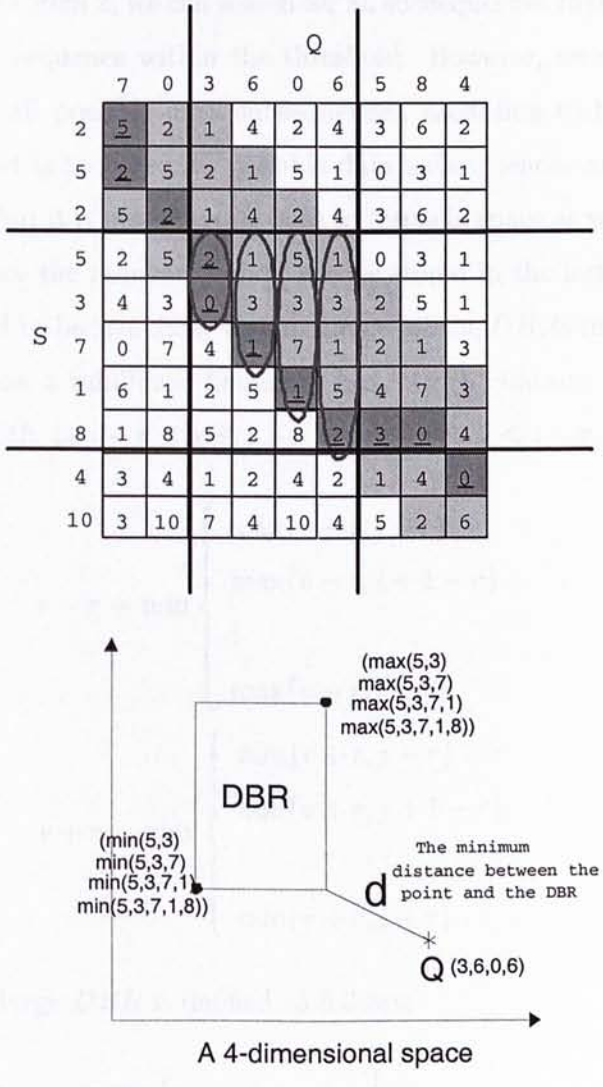

Figure 4.5: An example of the DBR.

with the DBR, then $D_{tw}(S[x:y], Q[i,j]) > \epsilon$.

Proof: See Appendix B.

Based on Theorem 2, we can search for all subsequences that agree with the given query subsequence within the threshold. However, we should perform comparison on all possible data subsequences, according to Lemma 3. The simplest method is to index all possible data subsequences and traverse the index directly, but it is not efficient, both in terms of space as well as searching speed. To reduce the number of data entries stored in the index and number of nodes needed to be searched, we can merge all the DBRs into a single one. That is to define a minimum bounding box, which contains all the DBRs. Consider Eq. 4.15, as $i - r \leq x \leq i + r$ and $j - r \leq y \leq j + r$, we have:

$$v - r = \min \begin{cases} \max\{v - r, i - r\} \\ \max\{v - r, i + 1 - r\} \\ \vdots \\ \max\{v - r, i + r\} \end{cases} \tag{4.16}$$

$$v + r = \max \begin{cases} \min\{v + r, j - r\} \\ \min\{v + r, j + 1 - r\} \\ \vdots \\ \min\{v + r, j + r\} \end{cases} \tag{4.17}$$

Therefore, the large DBR is defined as follows:

$$\left.\begin{array}{l} bl_{u-i+1} = \min\{s_{u-r} \ldots s_{u+r}\} \\ bh_{u-i+1} = \max\{s_{u-r} \ldots s_{u+r}\} \end{array}\right\} for \quad u = i \ldots j \tag{4.18}$$

An example of DBR is depicted in Figure 4.6. Based on this large DBR, we propose another indexing structure to handle subsequence queries of various lengths. This method stores multidimensional rectangles (DBRs) in the index

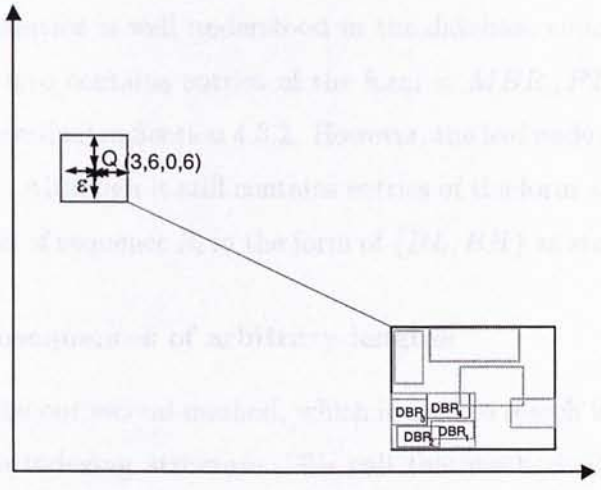

Figure 4.6: An illustration of merging all the DBRs

instead of points. The following section describes the index construction and searching algorithm based on the above strategy.

4.4.2 Algorithm

In this section, we present another subsequence searching method which can also handle query sequences of arbitrary lengths.

Index Construction

Given a set of data sequences $S = \{S_1, S_2, \ldots, S_m\}$ of different lengths, a sliding window of length $l + r$ is placed and slide over each offset of each data sequence $S_i \in S$. If the length of a subsequence is shorter than $l + r$, then adjust the length of the sliding window to $l + r - 1$, $l + r - 2$ and so on. The minimum length of the sliding window is $l - r$. Every prefix will be transformed to an l-dimensional DBR according to Eq. 4.18. Instead of indexing multidimensional points, we index the DBRs here. In the following, we also sketch our idea using R-tree, as we have mentioned that it is widely

used and its behavior is well understood in the database community. A non-leaf node of R-tree contains entries of the form $< MBR_i, PTR_i >$, which is same as that described in Section 4.3.2. However, the leaf node is different from the normal one. Although it still contains entries of the form $< SID'_i, S'_i >$, S'_i is now the DBR of sequence S_i in the form of $\{BL, BH\}$ as stated in Eq. 4.18.

Searching subsequences of arbitrary lengths

Here, we describe our second method, which is used to search for subsequences from the above indexing structure. We call this method 'RB-SSM'. If the length of the query is l, we can use Algorithm 3 directly.

Algorithm 3 RangeQuery(T,ϵ-Q)

```
1: C := {}
2: if T is not a leaf node then
3:    for each child node of T do
4:       if IsOverlap(MBR_i, ε-Q) then
5:          RangeQuery(PTR_i,ε-Q)
6:       end if
7:    end for
8: else
9:    if IsOverlap(S'_i, ε-Q) then
10:      C ← SID'_i
11:   end if
12: end if
```

For non-leaf node, check for each child node, if MBR_i overlaps with the ϵ-Q. If it is true, we continue to search that subtree. For leaf node, check if the DBR overlaps with the ϵ-Q. If it is true, SID'_i will be included in the candidate set.

It is more complicated to deal with longer queries, as the DBRs comprehend only the subsequences with length between $l - r$ to $l + r$. If the input query is longer than l, then by Lemma 2, we can select a subsequence with length l from Q(i.e. the prefix with length l), and use the DBR index to search for $DBRs$ that match the prefix of Q within the threshold ϵ. Then,

the unqualified answers are removed by post-processing. For each SID'_i in the candidate set, we get the corresponding subsequence from the database, e.g. $S_i[j:k]$. If $D_{tw}(S_i[j:k], Q) \leq \epsilon$, then $S_i[j:k]$ will be included in the final answers.

4.5 Experimental Results

In this section, we present the experimental results of our proposed methods. The data in our experiments are from the historical data of S&P 500 stocks, which were extracted from http://kumo.swcp.com/stocks/. We used the close price of 500 stock sequences with average length of 231. Each stock will generate about 215 subsequences by the sliding window approach. The system is written in 'C' and an R-tree is used as the index. We performed the experiments on Sun Ultra 5/360 with 256MB of main memory.

We carried out experiments to measure the performance of 'PB-SSM' and 'RB-SSM'. For each experiment, the query sequences were generated by selecting random subsequences from the database. For each query sequence, a threshold ϵ was given and searching was performed against that threshold. In each experiment, 10 queries were performed and the average results are collected. We evaluated our methods in different aspects:

1. The filtering power of both methods.
2. The performance of our proposed methods ('PB-SSM' and 'RB-SSM') and sequential search.
3. The effect of the width of the warping window on both methods.
4. The effect of the threshold value on both methods.

4.5.1 Candidate ratio vs Width of warping window

We measured the filtering power of both indexing methods by candidate ratio. Candidate ratio is the fraction of sequences, which requires post-processing to get the final answer. It is defined as follows [27]:

$$\text{candidate ratio} = \frac{\text{Number of candidate sequences that need to compute DTW}}{\text{Number of sequences in database}}$$

In this experiment, the number of stocks was fixed at 500. We tested our methods against various widths of warping window. The aim of this experiment is to analyzes the effect of the width of the warping window on the filtering power. The results are presented in Figure 4.7. The candidate ratio varies from 7% to 14%. Both 'PB-SSM' and 'RB-SSM' have similar filtering power using different widths of warping window. Although their filtering power is

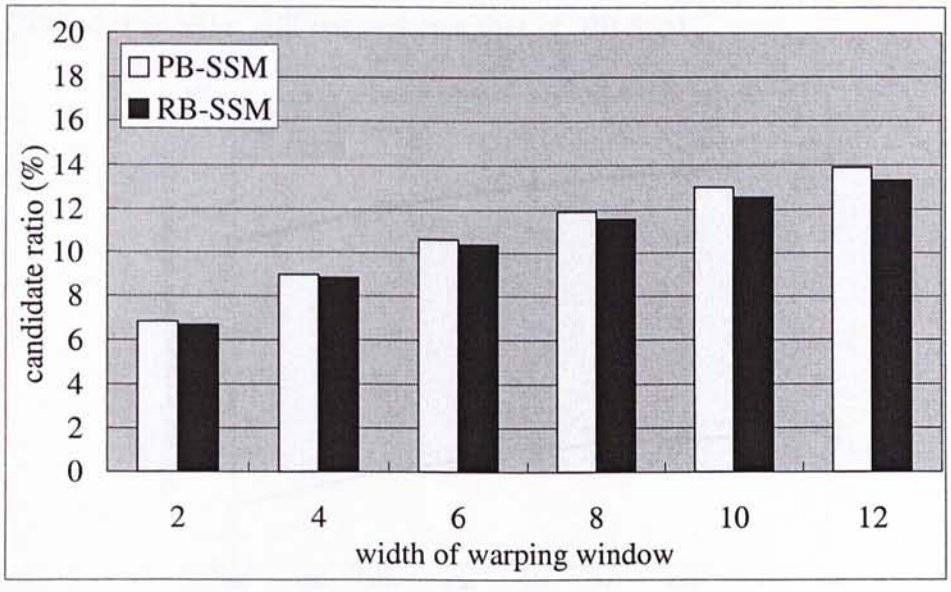

Figure 4.7: Comparison of the candidate ratio between 'PB-SSM' and 'RB-SSM'

similar, their performance may be different. Since they use different indexing strategies, the number of nodes needed to be traversed or the time spent on

filtering may be different during the query process. Therefore, we evaluated the average CPU time of both methods by the following experiments.

4.5.2 CPU time vs Number of subsequences

We evaluated the average CPU time by increasing the number of stocks from 100 to 500. The query length is 16 and ϵ is 16. In each experiment, the width of the warping window is 6. Figure 4.8 depicts the results. It can be observed that the 'RB-SSM' outperforms both 'PB-SSM' and sequential search. For sequential search, the query sequence needs to compare with all data sequences, so the CPU time increases as the number of data sequences increases. For our proposed method, the CPU time also increases as the number of data sequences increases. This is because more branches of the R-tree are needed to be traversed as the number of data points increases. However, the average CPU of 'RB-SSM' still outperforms that of 'PB-SSM'.

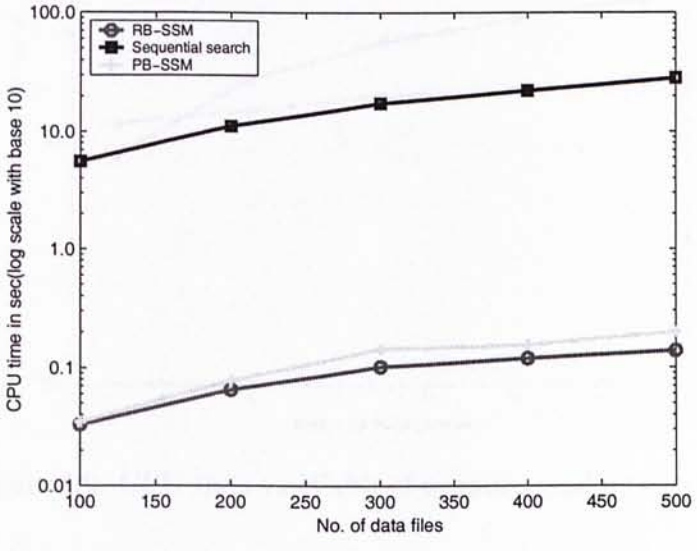

Figure 4.8: CPU time vs Number of stocks

4.5.3 CPU time vs Width of warping window

We examined our methods with different widths of warping window. The query length is 16 and we used 500 stocks to perform our experiments. Figure 4.9 shows the CPU time of both methods by varying the width of the warping window. The wider the warping window, the longer the CPU time is required. According to the results, 'RB-SSM' outperforms 'PB-SSM' when the width of the warping window is over 5. This is because the query of 'PB-SSM' is a hyper-rectangle with its volume depending on the width of the warping window. However, the query of 'RB-SSM', which is a point, is independent of the width of the warping window. Nevertheless, the width of the warping window still affects the 'RB-SSM' method, this may probably due to the increasing of overlapping region in the index.

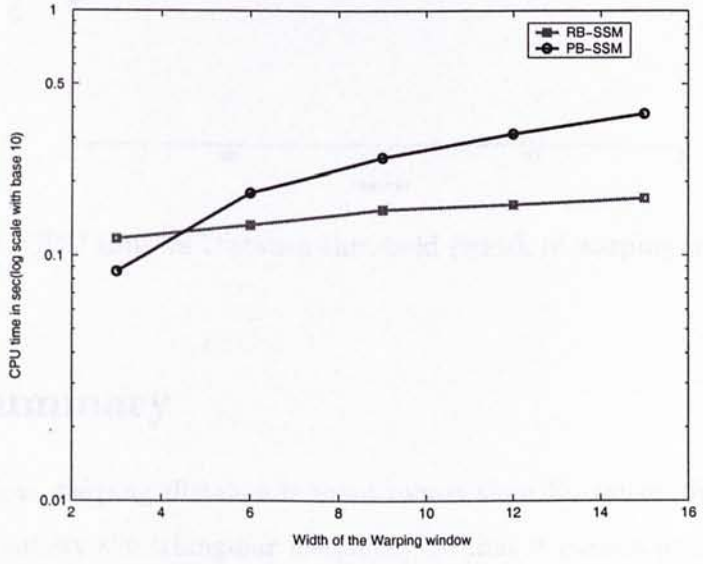

Figure 4.9: CPU time vs Width of warping window ($\epsilon = 16$)

4.5.4 CPU time vs Threshold

As for both 'PB-SSM' and 'RB-SSM', the query regions depend on the threshold, we performed experiments to compare the effect of the threshold on them.

We again used 500 stocks to perform our experiments and used warping window of width 6. As shown in Figure 4.10, the tendency of the performance is different from that in Figure 4.9. The effect of threshold on both methods are quite similar. The is because the query volume of both methods depend on the threshold, but that of 'PB-SSM' also depends on the width of the warping window.

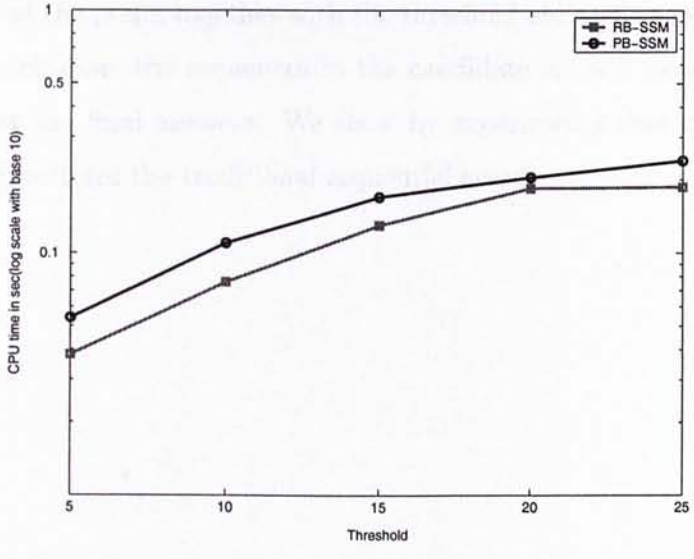

Figure 4.10: CPU time vs Distance threshold (width of warping window = 6)

4.6 Summary

Dynamic time warping distance is more robust than Euclidean distance, but it does not satisfy the triangular inequality, so that it cannot be speed up by indexing. Dozens of research work proposes to use a lower bound function to facilitate the indexing structure. However, most of them focus on whole sequence matching or use a loose lower bound function. We propose two approaches which can handle subsequence matching of arbitrary lengths under DTW. Our approaches guarantee no false dismissal and use small space overhead. Both of our methods are based on the sliding window approach and the

warping window constraint.

The first method indexes each prefix of all possible subsequences. For the query process, a query bounding rectangle, which is based on the potential subsequences of the query, is formed to perform range query. The second approach maps each prefix of all possible subsequences to a hyper-rectangle and indexed. For the query process, the corresponding prefix is extracted from the query and the prefix together with the threshold are used to perform range query. In each case, the sequences in the candidate set will be examined by DTW to get the final answers. We show by experiments that our proposed methods outperform the traditional sequential search.

Chapter 5

Relevance Feedback under Time Warping

In this chapter, we present a strategy to capture user's subjective requirement over the query process. The process is an interactive procedure between the user and computer. First, the user needs to provide an initial query. During the retrieval process, the system can automatically refine the existing query or the weights according to the user's feedback. To hide the low-level details of the calculation, the user only needs to indicate which answers are relevant in his or her point of view. Based on the information given by the user, the weights embedded in the query, that means the query, will be refined. As a result, the preference of the user can be captured by the updated query. Then the retrieval process will be continued with the "new" query.

5.1 Integrating Relevance Feedback with DTW

To capture the user's requirement, we refine the DTW by embedding the *dynamic path constraint* and *weights* in the distance calculation to facilitate our proposed feedback strategy in this section.

As defined before, the *global path constraint* controls the warping path in

a global scale. However, a static path constraint may not satisfy the requirements for all users. The examples below, which represent the preferences of different users, describe such case. The left hand side of Figure 5.1 shows two sequences and the corresponding warping path is depicted on the right hand side. There is a global path constraint, which is the grey area in the grid, to restrict the warping path. However, we can see that the path constraint misjudges the user's preference. Another example using the same path constraint is illustrated in Figure 5.2. In this example, the path constraint is fit for the warping path.

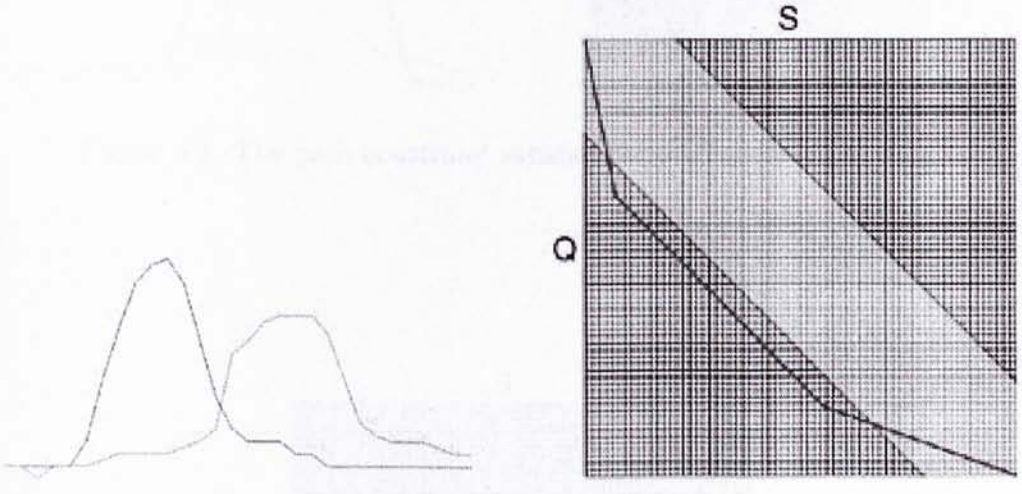

Figure 5.1: The path constraint misjudges the user's preference

Obviously, we need different path constraints for different users. We propose to update the path constraint (warping window) during the query process based on the feedback of the user. As a result, the constraints will be refined and look like that in Figures 5.3 and 5.4. The detail of the update mechanism will be described in Section 5.2.1.

Unlike the traditional path constraint, we do not restrict the warping path inside the warping window as we need to 'learn' the user's preference dynamically. Instead, we use different weighting factors for the elements inside and

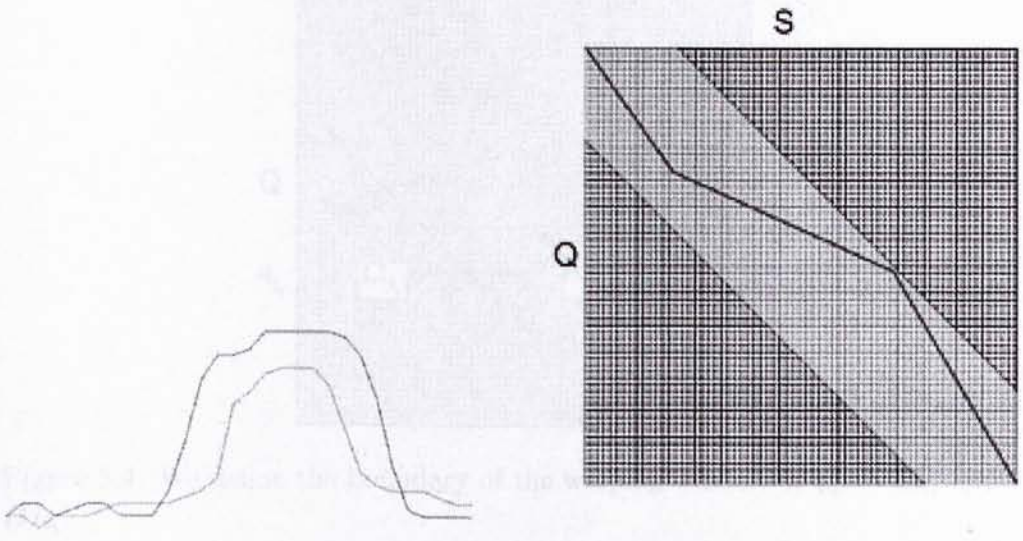

Figure 5.2: The path constraint satisfies the preference of the user

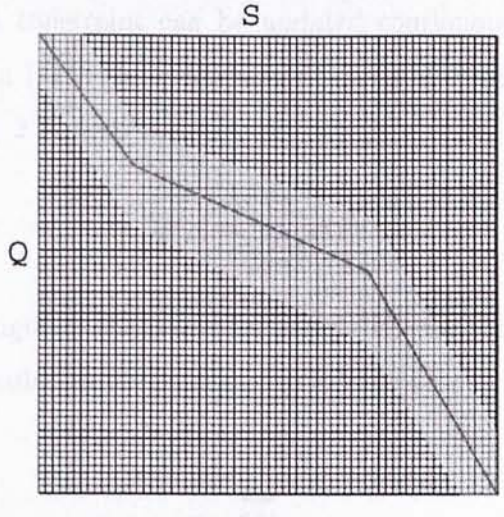

Figure 5.3: An example of path constraint which captures the preference of the user.

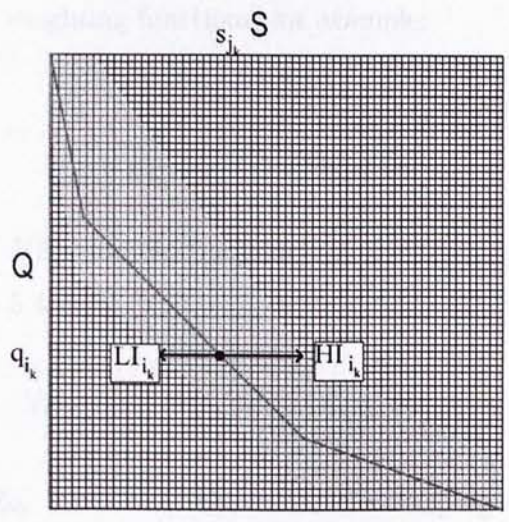

Figure 5.4: We define the boundary of the warping window of q_{i_k} as LI_{i_k} and HI_{i_k}.

outside the warping window. By updating the weights, the warping path can still pass through the region outside the warping window, but with different weighting factors. Thus the system can display some ambiguous results, which have various warping paths, for the user to choose. Based on the feedback of the user, the path constraint can be updated continuously. To embed the weighting factors in DTW, let's consider the typical cumulative distance formula of DTW (Eq. 2.3) first:

$$D_c(W) = \sum_{k=1}^{n} d(i_k, j_k) \tag{5.1}$$

, where n is the length of the query sequence. The weights can be embedded in the distance calculation by generalizing Eq. 5.1 as

$$D_{wc}(W) = \sum_{k=1}^{n} d(i_k, j_k) m(i_k) \tag{5.2}$$

, where $m(i_k)$ is a non-negative weighting function associated with q_{i_k}. There

are many possible weighting functions, for example:

$$m(i_k) = \begin{cases} 1 & if LI_{i_k} \leq j_k \leq HI_{i_k} \\ 1 + \frac{1}{HI_{i_k} - LI_{i_k} + 1} & otherwise \end{cases} \tag{5.3}$$

, where LI_{i_k} and HI_{i_k} are the boundary of the warping window of q_{i_k} as depicted in Figure 5.4, i.e.

$$\forall (i_k, j_k) \in W, \quad LI_{i_k} \leq j_k \leq HI_{i_k} \tag{5.4}$$

The meaning of Eq. 5.3 is that the wider the warping window is, the less important of q_{i_k} is. Without loss the generality of Eq. 2.4, the DTW distance becomes

$$D_{tw}(S, Q) = \min_{\forall W} \{D_{wc}(W)\} \tag{5.5}$$

We call this a "weighted DTW" distance. Accordingly, the dynamic programming recurrence formula (Eq. 2.5) becomes

$$\gamma(i, j) = d(i, j)m(i_k) + \min\{\gamma(i-1, j), \gamma(i, i-1), \gamma(i-1, j-1)\} \tag{5.6}$$

In next section, we will present our query reformulating strategy in detail.

5.2 Query Reformulation

In this section, we describe the constraint update scheme and the weighting function that we use. The main symbols used in this chapter and their definitions are summarized in Table 5.1.

5.2.1 Constraint Updating

To update the path constraint during the query process, we define an operation named "reform_constraint". The constraint is refined according to the warping

Symbol	Definition
Q	A query sequence.
S	A data sequence.
HI_{i_k}	The upper bound of the warping window of q_{i_k}.
LI_{i_k}	The lower bound of the warping window of q_{i_k}.
R	A set of relevant results that are marked by the user, i.e. $\{R_1 \dots R_n\}$.
$m_p(i_k)$	The weighting function of q_{i_k} in the p^{th} iteration of feedback.
$c(p)$	Number of relevant results that are marked by the user in the p^{th} iteration of feedback.
$W_{Q,S}$	An optimal warping path between S and Q.
$D_{tw}(Q,S)$	The time warping distance between sequences S and Q

Table 5.1: Summary of Symbols and Definitions

path between the query Q and a data sequence S, which is relevant to Q. The basic idea is that the new path constraint should enclose the warping path, for which the sequence is relevant to Q. Given a set of sequences, which are relevant to Q, we repeat "reform_constraint" by those sequences to update the path constraint continuously.

The algorithm stated in Algorithm 4 describes our constraint refinement operation. Initially, there is no path constraint, and all HI_{i_k}s are initialized to $-\infty$ and LI_{i_k}s to $+\infty$. After the first feedback, the path constraint becomes a line. If the path constraint is a line, the constraint will be updated like merging two warping paths as depicted in Figure 5.5. After the first two iterations of feedback, a path constraint is defined. Figure 5.6 shows an example of the refinement of path constraint according to another warping path afterward.

Algorithm 4 reform_constraint(S,Q)

```
1: for (i_k, j_k) in W_{S,Q} do
2:    if j_k > HI_{i_k} then
3:       HI_{i_k} = j_k
4:    else if j_k < LI_{i_k} then
5:       LI_{i_k} = j_k
6:    end if
7: end for
```

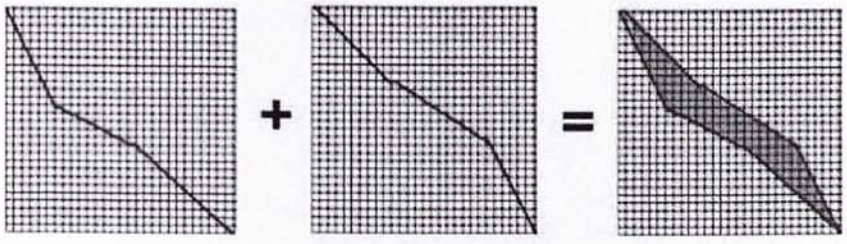

Figure 5.5: An example of path constraint update at the initial stage.

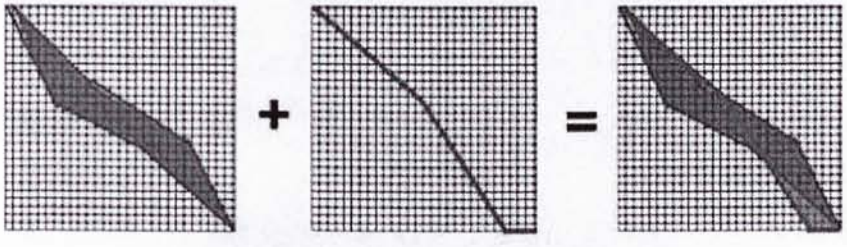

Figure 5.6: An illusion of Algorithm 4

5.2.2 Weight Updating

After updating the path constraint, we can refine the weights accordingly. As mentioned in Section 5.1, the weight $m(i_k)$ is associated with q_{i_k}. To define the weight of q_{i_k} and the refinement methodology, we first introduce two factors, the importance of q_{i_k} and the importance of current iteration.

Intuitively, if the width of the warping window of q_{i_k} is small, it means that q_{i_k} is more significant, as q_{i_k} has similar alignments against all relevant sequences. On the other hand, if the alignments of q_{i_k} with the relevant sequences are very different, then q_{i_k} is less important. According to this analysis, we propose to use the inverse of the width of the warping window as the importance of q_{i_k}. In other words, the wider the warping window is, the smaller the importance is.

Definition 1 (Importance of q_{i_k}) *Given a query sequence and its path constraint, which is constructed based on the relevant sequences, the importance,*

$\Omega(i_k)$, *of each* q_{i_k} *is defined as:*

$$\Omega(i_k) = \frac{1}{HI_{i_k} - LI_{i_k} + 1} \tag{5.7}$$

Here, we assume that there is at least one sequence which can match the user's preference.

Figure 5.7 delineates the intuitive idea of $\Omega(i_k)$. The bar chat on the left hand side illustrates the importance of each q_{i_k}, which is contributed by the warping path on the right hand side.

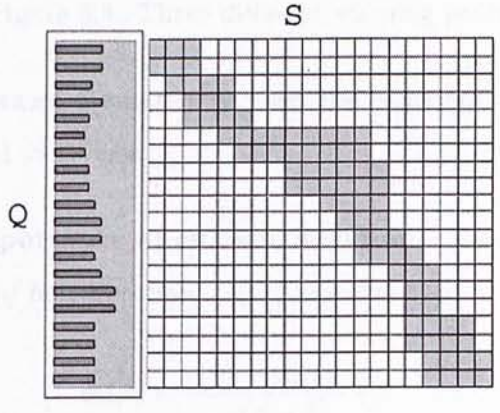

Figure 5.7: An example of warping path constraint together with the importance of q_{i_k}

Using the importance of q_{i_k} as the only one quantitative measure of the weighting function is not enough since the implication of some relevant sequences may be hidden. Such case is shown in Figure 5.8. The path constraint constructed by paths A, B, and C is the same as that constructed by paths A and B only. As a result, the sequence, which has path C against the query, is "ignored". Therefore, we propose to use the importance of the current iteration as another weighting factor.

The importance of the current iteration is actually the changing rate of the number of relevant sequences in each iteration, p. If the number of relevant

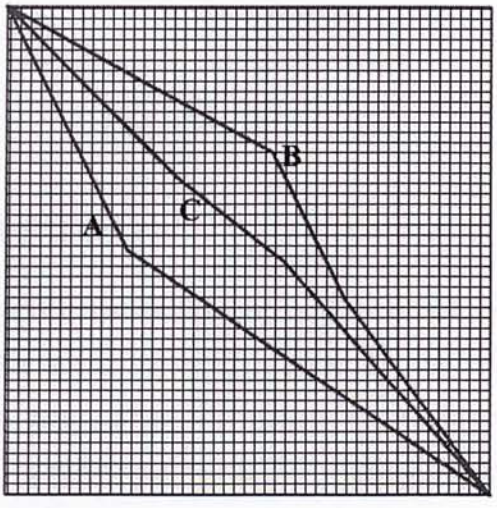

Figure 5.8: Three different warping paths.

answers, $c(p)$, increases dramatically, then the contribution of that iteration should be large and vice versa.

Definition 2 (Importance of current iteration) *For a particular iteration, p, the importance of that iteration, $\Delta(p)$, is defined as:*

$$\Delta(p) = \frac{c(p) - c(p-1)}{c(p-1)} \tag{5.8}$$

After introducing the above two kinds of importance, we now define the weight, $m_p(i_k)$, of q_{i_k}. We first initialize $m_0(i_k) = 1$. At each iteration, the weights are refined by "inheriting" from the previous weight and "plus" the current influence as stated in Eq. 5.9. The current influence is composed of the two factors (importance)introduced before. After the refinement, if $m_p(i_k)$ is smaller than 1, then set it to 1.

$$m_p(i_k) = \begin{cases} 1 & if LI_{i_k} \leq j_k \leq HI_{i_k} \\ m_{p-1}(i_k) + f_p(i_k) & otherwise \end{cases} \tag{5.9}$$

where,

$$f_p(i_k) = \Omega(i_k) \times \Delta(p) \tag{5.10}$$

5.2.3 Overall Strategy

The overall query reformulation strategy and feedback procedure are stated in Algorithm 5. An initial query Q is given by the user and the weights are all initialized to one. Then the system will rank the sequences in the database according to the "weighted DTW" distance between the query and the sequences. The best n sequences will be displayed for the user to choose. The user only needs to mark those sequences relevant to the query according to his or her view point. After that, the path constraint and weights associated with the query will be updated by the procedures stated in Section 5.2.1 and Section 5.2.2 respectively. The query process will be iterated until the user finish querying.

Algorithm 5 Overall

```
User given the initial query Q
Initialize all HI = −∞
Initialize all LI = +∞
Initialize all m_0(i) = 1
repeat
    Find the best n results
    Show the sequences to the user
    User states which sequences are relevant, e.g R
    for each sequence R_i in R do
        refrom_constraint(R_i,Q)
    end for
    for i = 1 to n do
        update m(i) used Eq. 5.9
    end for
until User finish query
```

5.3 Experiments and Evaluation

To evaluate the performance of our proposed method, we conducted experiments on four similar groups of sequences. Each group contains 500 data sequences. We generated the sequences using $\sin(x)$ and Gaussian noise described as follows:

Group A are sin waves shift to left, with Gaussian noise of $\sigma = 0.2$.

Group B are sin waves shift to right, with Gaussian noise of $\sigma = 0.2$.

Group C are sin waves in the middle, with Gaussian noise of $\sigma = 0.2$.

Group D are shrunken sin waves, with Gaussian noise of $\sigma = 0.2$.

All sequences are of length 40. Figure 5.9 shows an example from each group. The initial query was constructed by averaging all the 2,000 sequences.

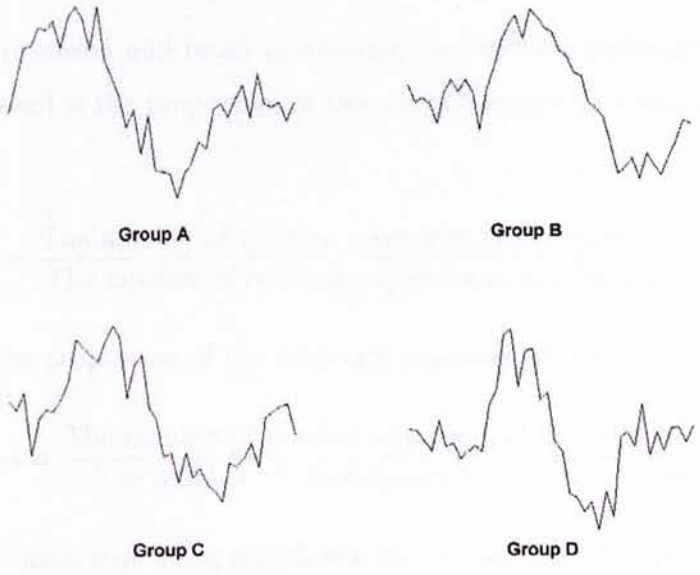

Figure 5.9: An example of each group of sequences.

It is assumed in all the experiments that the user will not change his or her goal during the retrieval process, so the feedback process can be simulated by a computer. For every experiments, the feedback procedure was simulated by the computer using the following steps:

1. One of the groups(A,B,C, or D) was chosen to be the objective.
2. The initial query was given.
3. The best n results were returned by the computer.
4. Among the best n results, the sequences, which are the same group of the objective, were marked as relevant.
5. The path constraint and weights were updated as stated in Sections 5.2.1 and 5.2.2.
6. The feedback procedure was repeated m times, which is the testing parameter of our experiment.

We used precision and recall to evaluate the retrieval performance of our approach. Recall is the proportion of the relevant sequences which have been returned, i.e.

$$Recall = \frac{\text{The number of relevant sequences in the answer set}}{\text{The number of relevant sequences in the databases}}$$

Precision is the proportion of the retrieved sequences which are relevant, i.e.

$$Precision = \frac{\text{The number of relevant sequences in the answer set}}{\text{The number of the sequences in the answer set}}$$

The performance of an approach can be measured by the precision at a particular recall level. For example, we want to measure the precision at 50% recall level. We can choose the answer set, which contains 50% of the relevant sequences in the databases. Note that we should choose the answer

set according to the ranking of the sequences. Among the answer set, the proportion of the relevant sequences is the precision.

5.3.1 Effectiveness of the strategy

The aim of relevance feedback is to return what the users want. Therefore, it is important to measure how good the results are. We first ran our experiments using different numbers of top n results as feedback and measured the precisions of the first two iterations. The experimental results are illustrated by using precision recall graphs. We tested our proposed strategy by giving the top 10, the top 20 and the top 30 results as feedback and the statistics are shown in Figures 5.10, 5.11 and 5.12 respectively. For each case, we measured the precisions of our approach varying from 0% recall to 100% recall, and the precisions of first two iterations were recorded in the figures. From the figures, it can be observed that the precisions of the results greatly increased after the first two iterations. Obviously, the refined query can match users' perceptions of similarity.

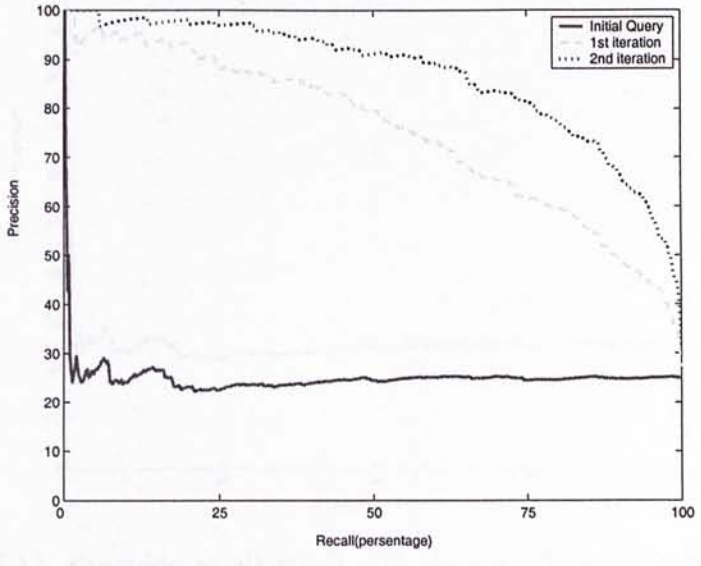

Figure 5.10: Precision recall graph with the top 10 results as feedback.

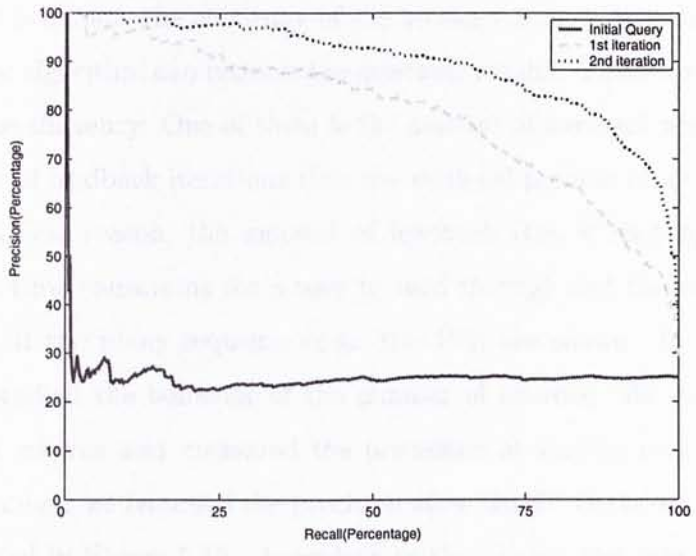

Figure 5.11: Precision recall graph with the top 20 results as feedback.

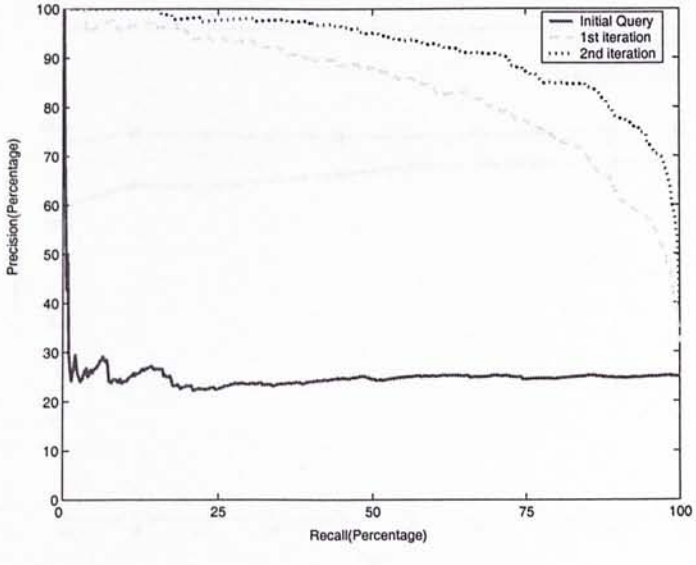

Figure 5.12: Precision recall graph with the top 30 results as feedback.

5.3.2 Efficiency of the strategy

Besides the accuracy, the efficiency of the strategy is also important. That is how fast the algorithm can retrieve the desirable results. There are two factors affecting the efficiency. One of them is the amount of feedback and another is the number of feedback iterations that the retrieval process needs.

For practical reason, the amount of feedback (top n results) should be small. It is time consuming for a user to read through and choose among all the results, if too many sequences(e.g. top 100) are shown. In this experiment, we studied the behavior of the number of returns. We gave different numbers of returns and measured the precisions at various recall levels. In each experiment, we recorded the precision after the 3^{rd} iteration. The results are delineated in Figure 5.13. According to the results, the greatest increase of precision occurs at top 15 returns and the increasing rate approaches zero at about top 40 returns. It is a desirable outcome, as it is only 2% of the whole data set. Therefore, the user can use less effort to accomplish his or her mission.

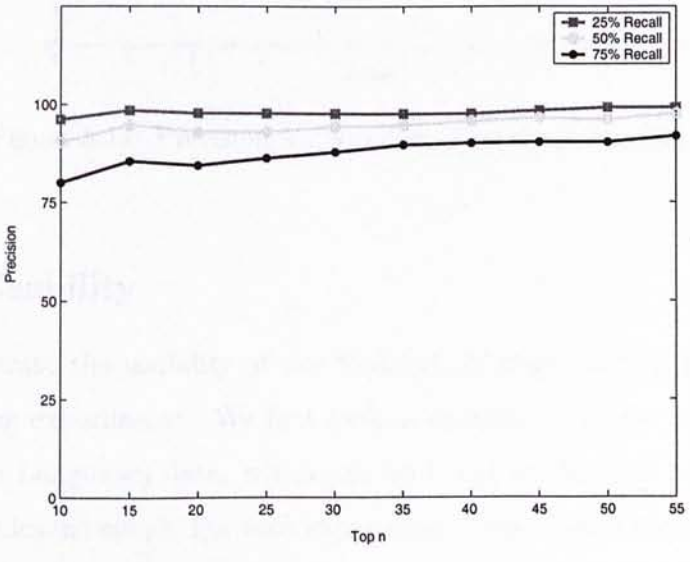

Figure 5.13: Precision Vs Number of results for feedback

On the other hand, the user cannot give feedback forever. Hence, the number of iterations is also very critical. We performed experiments to study the effect of the number of iterations. In the experiment, we measured the precisions of each iteration at different recall levels. Figure 5.14 illustrates the results. According to the experiments, the greatest increase of precision appears at the first iteration and the later iterations cause minor increase of precision. This property is very preferable, as reasonable results can be achieved within a short time.

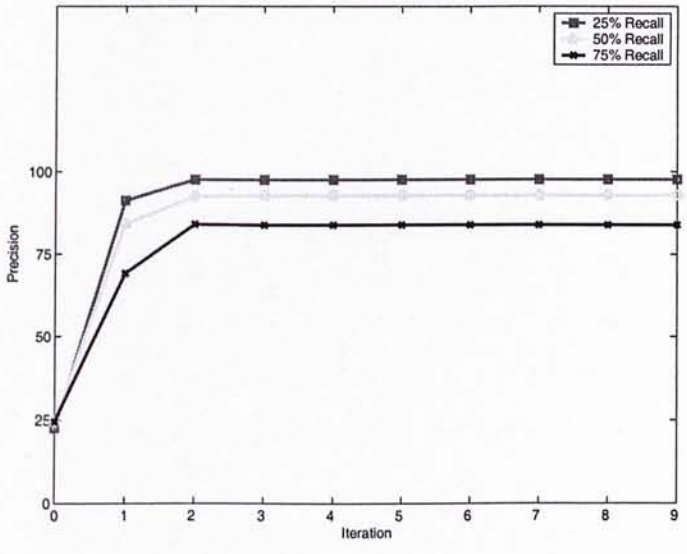

Figure 5.14: Precision Vs Number of feedback iterations

5.3.3 Usability

To demonstrate the usability of our feedback strategy on real data, we ran the following experiments. We first built a database from the Auslan (Australian Sign Language) data, which can be found at the UCI KDD Archive (http://kdd.ics.uci.edu/). For each experiment, a time series was randomly selected from the database as the initial query. The top 20 results were returned and judged by a user. The user marked the relevant results and repeated the

search. The results are summarized in Table 5.2 - 5.6. In each table, the user feedback, new answers based on the preference of the user, and the unqualified answers after feedback are presented respectively.

The tables are divided into three parts. The sequences in each part are presented in pair, where the red one is query and the black one is result from database, for visual comparison. Part one are the sequences marked by the user as relevant. Part two are the new retrieved sequences according to the user's preference. Part three are those removed sequences after user feedback.

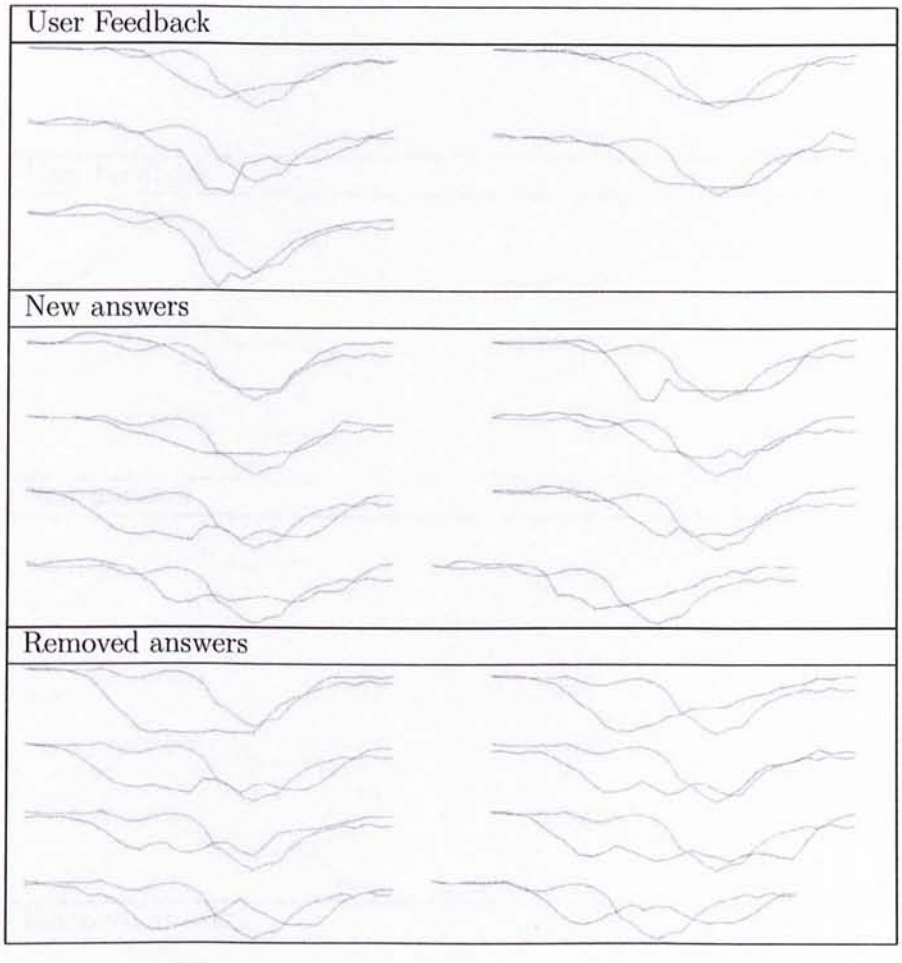

Table 5.2: Example I of relevance feedback on time series.

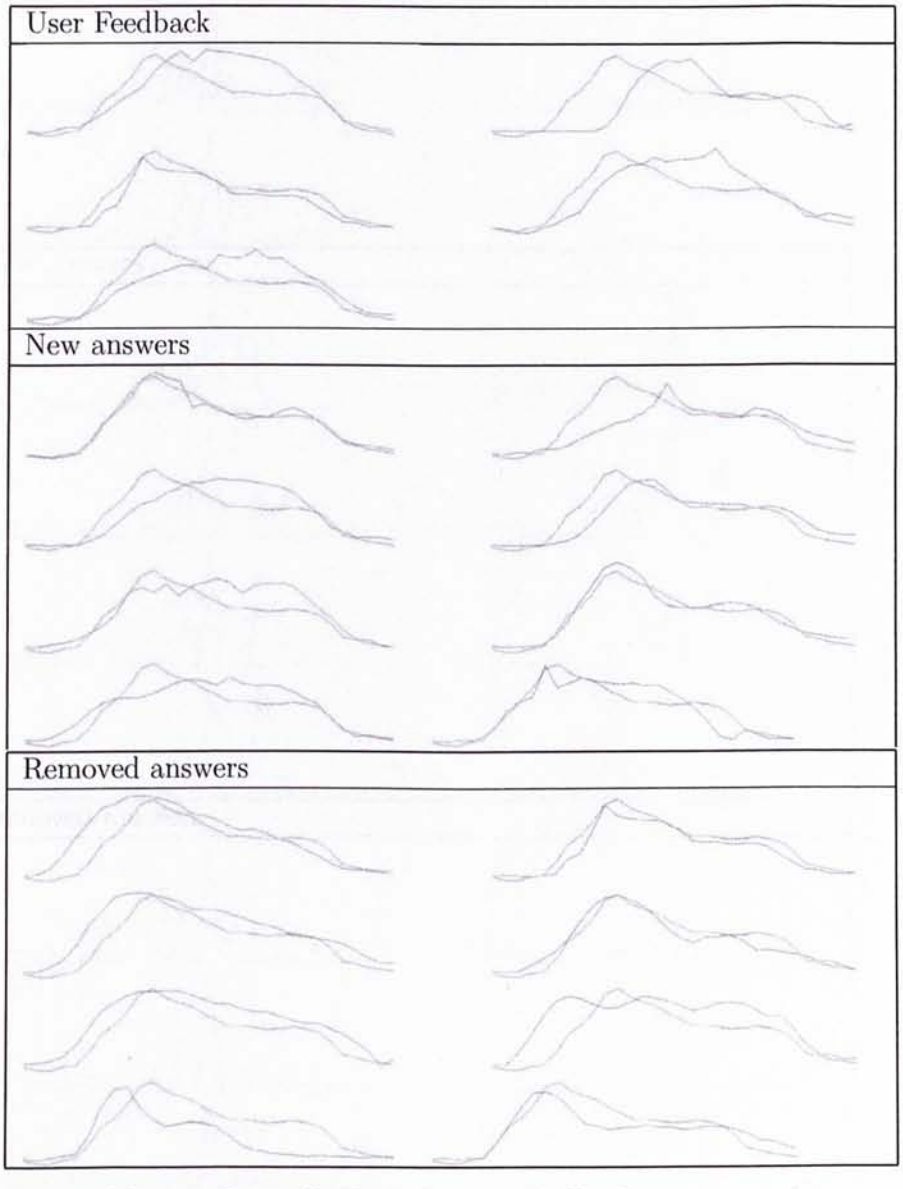

Table 5.3: Example II of relevance feedback on time series.

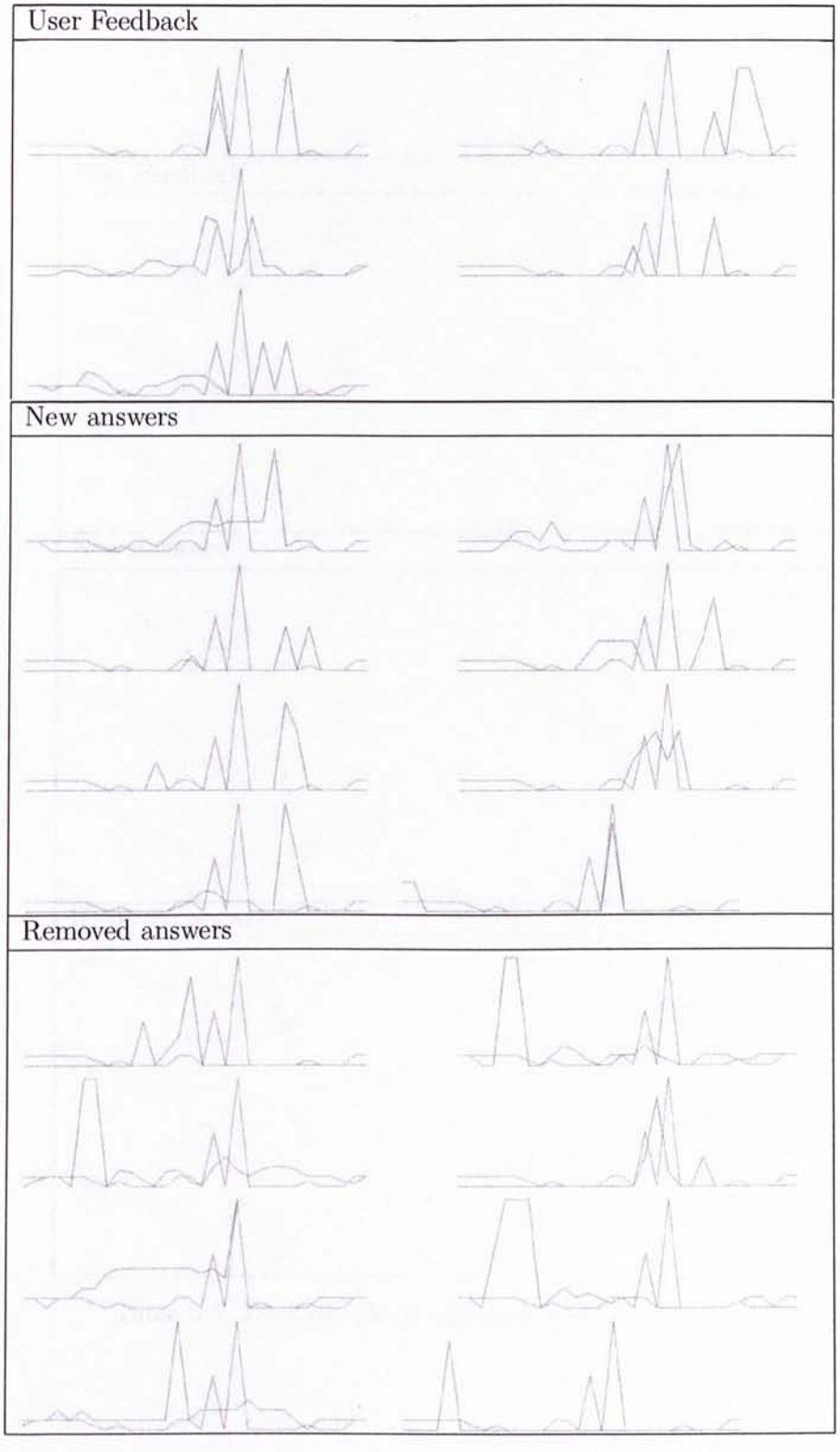

Table 5.4: Example III of relevance feedback on time series.

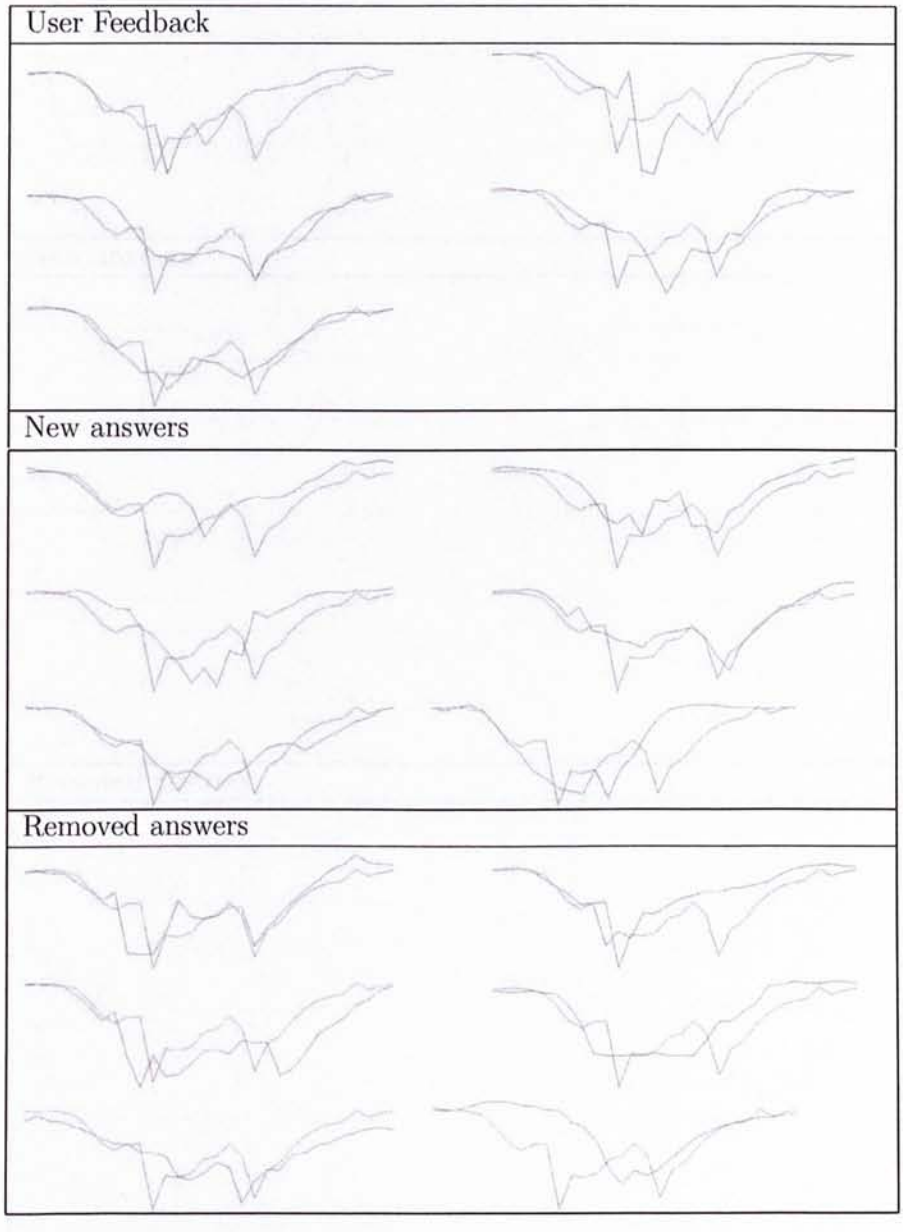

Table 5.5: Example IV of relevance feedback on time series.

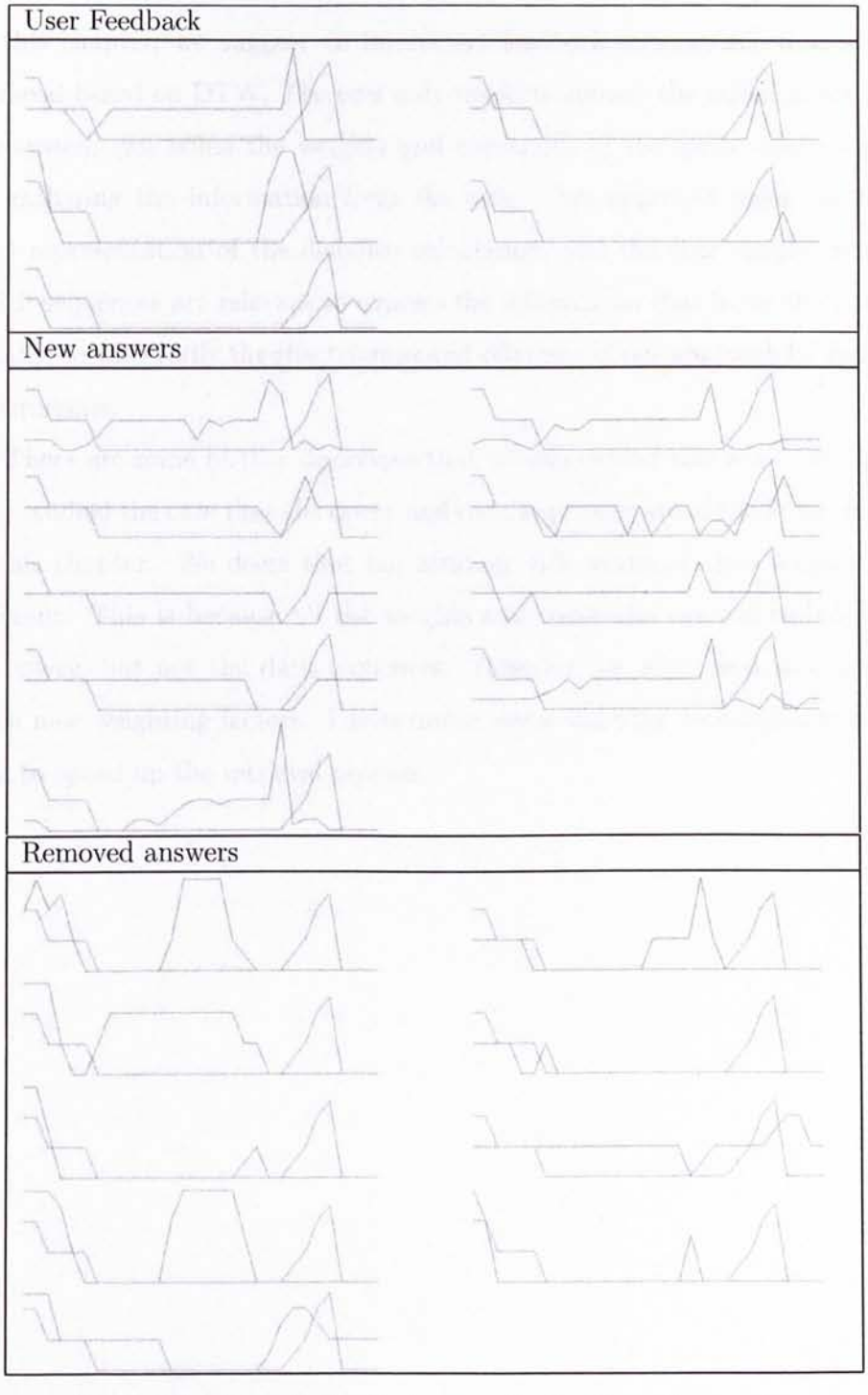

Table 5.6: Example V of relevance feedback on time series.

5.4 Summary

In this chapter, we suggest an interactive feedback strategy for time series retrieval based on DTW. The user only needs to submit the initial query and the system can refine the weights and constraint of the query continuously by gathering the information from the user. Our approach hides the low-level representation of the distance calculation, and the user simply decides which sequences are relevant to express the information that he or she wants. Finally, we also verify the effectiveness and efficiency of our approach by various experiments.

There are some further directions that we can extend this work. We have only studied the case that the query and data sequences are of the same length in this chapter. We deem that our strategy still works, if their lengths are different. This is because all the weights and constraint are associated with the query, but not the data sequences. However, we may need to consider some new weighting factors. Furthermore, some indexing techniques may be able to speed up the retrieval process.

Chapter 6

Conclusion

In this thesis, we study the problem of subsequences matching under Dynamic Time Warping (DTW). DTW has been widely studied recently in the time series domain, as it is more robust than Euclidean distance. Different lower bound techniques are suggested for similarity searching under DTW. However, most of them focus on whole sequence matching, or suggest a very loose lower bound. Two approaches are proposed in this dissertation to handle subsequences matching. We first generalize a better lower bound technique to handle subsequences matching of arbitrary lengths under DTW. We suggest an efficient approach, which guarantees no false dismissal and uses small space overhead. Our approach is based on the sliding window and the warping window constraint. Based on the sliding window approach, each prefix of all possible subsequences is indexed. For the query process, a query bounding rectangle (QBR) is formed to perform range query. Each answer in the candidate set is examined by DTW to get the final answer.

On the other hand, we notice that the QBR is very similar to the MBR. Thus we propose another indexing strategy, which is to form the MBRs based on the warping window constraint for each possible subsequences. Then, the leaf nodes of the R-tree will contain data bounding rectangles(DBR) instead of points. Hence, the query sequence can be used directly as a multi-dimensional

point instead of forming a bounding rectangle. Experiments are done on historical stock data. From the experiments, we show that our proposed methods outperform the traditional sequential search and the rectangle-based index has better performance than the point-based one when the width of the warping window is large.

While Dynamic Time Warping is more robust than Euclidean distance, it still cannot capture users' perceptions of similarity and preferences. Relevance feedback is a technique used to "learn" user's preference and perform retrieval. Most of the research effort about relevance feedback focuses on the image and text domain, but less attempt has been done on the the time series domain. Therefore, we suggest to utilize relevance feedback with DTW to enhance the precision of the retrieved answers.

During the query process, the user can simply choose the relevant sequences as feedback. The warping window constraint and the weighting factors embedded in the DTW are refined accordingly to accomplish the user's goal. We have run experiments on both synthesis and real data and the proposed method shows improvement on the precision of the retrieved answers.

Appendix A

Deduction of Data Bounding Hyper-rectangle

Here, we show how we can deduce the Data Bounding Hyper-rectangle. Consider the scenario stated in Section 4.3.1. $Q[3:6]$ is aligned with $S[4:8]$. The time warping distance between them is:

$$D_{tw}(S[4:8], Q[3:6]) = d(4,3) + d(5,3) + d(6,4) + d(7,5) + d(8,6)$$

If we remove the duplicated elements (i.e. (i,x),(j,x),(k,x) ...) in every columns, e.g. $(5,3)$, we have:

$$D_{tw}(S[4:8], Q[3:6]) \geq d(4,3) + d(6,4) + d(7,5) + d(8,6)$$

We define this operation as F_{rd}, i.e.

$$D_{tw}(S[x:y], Q[i:j]) \geq F_{rd}(D_{tw}(S[x:y], Q[i:j])) \tag{A.1}$$

According to Eq. 2.2, each q_v must align with one s_u, where $s_u \in \{s_{max(v-r,x)} \ldots s_{min(v+r,y)}\}$. For example, each element of $Q[3:6]$ must align with at least one element of $S[4:8]$ within the warping window, i.e. q_4 must align with s_4, s_5, or s_6.

Therefore, we have the following inequality.

$$
\begin{aligned}
d(u,v) &\geq \min\{d(max(v-r,x),v)\ldots d(min(v+r,y),v)\} \\
&\geq \min\{|s_{max(v-r,x)}-q_v|\ldots|s_{min(u+r,y)}-q_v|\} \\
&\geq \begin{cases}
|\max\{s_{max(v-r,x)}\ldots s_{min(v+r,y)}\}-q_v| \\
if q_v > \max\{s_{max(v-r,x)}\ldots s_{min(u+r,y)}\} \\
|\min\{s_{max(v-r,x)}\ldots s_{min(v+r,y)}\}-q_v| \\
if q_v < \min\{s_{max(v-r,x)}\ldots s_{min(u+r,y)}\} \\
0 \quad otherwise
\end{cases} \\
&\equiv d_{lb}(u,v) \qquad \text{(A.2)}
\end{aligned}
$$

We define the lower bound of each element as $d_{lb}(u,v)$. From Eq. A.1 and Eq. A.2, we have the following inequality:

$$D_{tw}(S[i:j],Q[x:y]) \geq F_{rd}(\sum_{v=i}^{j} d_{lb}(u,v)) \qquad \text{(A.3)}$$

Appendix B

Proof of Theorem 2

Theorem 3 Given two subsequences $S[x:y]$ and $Q[i:j]$, where $l = Len(Q[i:j])$ and $i - r \leq x \leq i + r$ and $j - r \leq y \leq j + r$, if ϵ-Q is not overlapped with the DBR, then $D_{tw}(S[x:y], Q[i,j]) > \epsilon$.

Proof: By contradiction,

Assume ϵ-Q is not overlapped with DBR. If $D_{tw}(S[x:y], Q[i,j]) \leq \epsilon$, then from equation Eq. A.3, we have,

$$F_{rd}(\sum_{v=i}^{j} d_{lb}(u,v)) \leq \epsilon \tag{B.1}$$

By assumption, there exists a q_v such that,

$$\begin{aligned}
&\Rightarrow \begin{cases} q_v + \epsilon < \min\{s_{max(v-r,x)} \cdots s_{min(v+r,y)}\} \\ q_v - \epsilon > \max\{s_{max(v-r,x)} \cdots s_{min(v+r,y)}\} \end{cases} \\
&\Rightarrow \begin{cases} |q_v - \min\{s_{max(v-r,x)} \cdots s_{min(v+r,y)}\}| > \epsilon \\ |q_v - \max\{s_{max(v-r,x)} \cdots s_{min(v+r,y)}\}| > \epsilon \end{cases} \\
&\Rightarrow \; d_{lb}(u,v) > \epsilon
\end{aligned}$$

which contradicts with inequality Eq. B.1, so that the proof is completed.

Bibliography

[1] Gaurav Aggarwal, Ashwin T.V., , and Sugata Ghosal. An image retrieval system with automatic query modification. *IEEE Transactions on Multimedia*, (2):201–214, June 2002.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *Proceedings of the Fourth Intl. Conf. On Foundations of Data Organization and Algorithm*, pages 69–84, 1993.

[3] R. Agrawal, K.-I. Lin, Harpreet S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling and translation in time-series databases. *Proceedings of the 21st VLDB Conference*, pages 490–501, 1995.

[4] R. Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying shapes of histories. *In proceedings of the 21st International Conference on Very Large Databases*, pages 502–514, 1995.

[5] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

[6] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *Proceedings of the ACM SIGMOD COnference on Management of Data*, pages 322–331, 1990.

[7] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The x-tree: An index structure for high-dimensional data. *Proceedings of the 22th VLDB conference*, pages 28–39, 1996.

[8] J. Berndt, D. & Clifford. Using dynamic time warping to find patterns in time series. *AAAI-94 Workshop on Knowledge Discovery in Database*, pages 229–248, 1994.

[9] K. P. Chan and A. Fu. Efficient time series matching by wavelets. *15th IEEE ICDE*, pages 126–133, 1999.

[10] Chia-Hui Chang and Ching-Chi Hsu. Enabling concept-based relevance feedback for information retrieval on the www. *IEEE Transactions on Konwledge and Data Engineering*, 11:595–609, July/Auguest 1999.

[11] K. W. Chu and M.H.Wong. Fast time-series searching with scaling and shifing. *PODS*, pages 237–248, 1999.

[12] Ingemar J. Cox, Matt L. Miller, Thomas P. Minka, Thomas V. Papathomas, and Peter N. Yianilos. The bayesian image retrieval system, pichunter, theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, January 2000.

[13] Goldin D. and Kanellakis P. On similarity queries for time-series data: constraint specification and implementation. *Proceedings of the 1st Int'l Conference on the Principles and Practice of Constraint Programming*, pages 137–153, 1995.

[14] Korn F., Jagadish H., and Faloutsos C. Efficiently supporting ad hoc queries in large datasets of time sequences. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pages 289–300, 1997.

[15] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series database. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 419–429, 1994.

[16] Guo-Dong Guo, Anil K. Jain, Wei-Ying Ma, , and Hong-Jiang Zhang. Learing similarity measure for natural image retrieval with relevance feedback. *IEEE Transactions on Neural Networks*, 13(4), July 2002.

[17] Antonm Guttman. R-tree: A dynamic index structure for spatial searching. *Proceedings ACM SIGMOD Conference on Management of Data*, pages 47–57, 1984.

[18] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. *PODS*, pages 36–45, 1995.

[19] Tamer Kahveci and Ambuj K. Singh. Variable length queries for time series data. *ICDE*, 2001.

[20] N. Katayama and S. Satoh. The sr-tree: An index struture for high-dimensional nearest neighbor queries. *Proceedings of tje ACM SIGMOD Conference on Management of Data*, pages 369–380, 1997.

[21] Eamonn Keogh, Kaushik Chakrabarti, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD*, pages 151–162, 2001.

[22] Eamonn Keogh and Padhraic Smyth. A probabilistic appraoch to fast pattern matching in time series databases. *Proceedings of the third conference on Knowledge Discovery in Databases and Data Mining*, pages 24–30, 1997.

[23] Eamonn J. Keogh. Exact indexing of dynamic time warping. *VLDB*, pages 406–417, 2002.

[24] Eamonn J. Keogh and Michael Pazzani. Relevance feedback retrieval of time series data. *SIGIR*, pages 183–190, 1999.

[25] Eamonn J. Keogh and Michael J. Pazzani. An indexing scheme for fast similarity search in large time series databases. *Proceedings Eleventh International Conference on Scientific and Statistical Database Management*, pages 239–241, 1999.

[26] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping to massive datasets. *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 1–11, 1999.

[27] S.-K. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. *IEEE ICDE*, pages 607–614, 2001.

[28] Flip Korn, H.V.Jagadish, and Christos Faloutsos. Efficient supporting adhoc queries in large datasets of time sequences. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 289–300, 1997.

[29] S. K. Lam and M. H. Wong. A fast project algorithm for sequence data seraching. *DKE 28(3)*, pages 321–339, 1998.

[30] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequenceses. *Proceedings of the 12th International Conference on Data Engineering*, pages 546–553, 1996.

[31] Joo-Hwee Lim, Jian Kang Wu, Sumeet Singh, , and Desai Narasimhalu. Learning similarity matching in multimedia content-based retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):846–850, 2001.

[32] Hong P, Tian Q, and Huang TS. Incorporate support vector machines to content-based image retrieval with relevance feedback. *IEEE International Conference on Image Processing*, pages 750–753, 2000.

[33] S. Park, W. W. Chu, J. Yoon, and C. Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. *ProceedingsIEEE ICDE*, pages 23–32, 2000.

[34] S. Park, S-W Kim, J-S Cho, and S. Padmanabhan. Prefix-querying: An approach for effective subsequence matching under time warping in sequence databases. *CIKM 2001*, pages 255–262, 2001.

[35] S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. *Proceedings IEEE KDEX Workshop*, 1999.

[36] T. Pavlidis and S.L. Horowits. Segmentation of plane curves. *IEEE Transactions on Computers*, 23:860–870, 1974.

[37] Chang Shing Perng, Haixum Wang, Sylvia R Zhang, and D. Stoot Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. *Proceedings of the 15th International Conference on Data Engineering*, pages 33–42, 2000.

[38] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Stott Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. *Proceedings of the 16th International Conference on Data Engineering*, 2000.

[39] Rosalind W. Picard. Computer learning of subjectivity. *ACM Computing Surveys*, 27(4):621–623, Dec. 1995.

[40] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. *ICDE*, 2002.

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

[42] D. Rafiei and A. Menedelzon. Similarity-based queries for time series data. *In Proceedings ACM SIGMOD*, pages 13–24, 1997.

[43] Rocchio, J. J., and Jr. *Relevance feedback in information retrieval: The Smart System - Experiment in Automatic Document Processing.* ed. Salton, G., Prentice-Hall Inc., 1971.

[44] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Video Technology*, 1998.

[45] Picard RW, Minka TP, and Szummer M. Modeling user subjectivity in image libraries. *International Conf. on Image Processing*, 1996.

[46] G. Saltion. *Automatic text processing.* Addison-Wesley, 1989.

[47] T. Sellis, N. Roussopoulos, and C. Faloutsos. R+tree: a dynamic index for multi-dimensional objects. *Proceedings of the 13th VLDB Conference*, pages 507–518, 1987.

[48] H. Shatkay and S. Zdonik. Approximate queries and representations for large data sequences. *Proceedings of 12th IEEE International Conference on Data Engineering*, pages 546–553, 1996.

[49] Kam Wing Shu, Sze Kin Lam, and Man Hon Wong. An efficient hash-based algorithm for sequence data searching. *The Computer Journal*, pages 402–415, 1998.

[50] Sergios Theodoridis and Konstantinos Koutroumbas. Pattern recognition. *San Diego : Academic Press*, 1999.

[51] Viola P Tieu K. Boosting image retrieval. *IEEE Conf. in Computer Vision and Pattern Recognition*, 2000.

[52] Ashwin T.V., Rahul Gupta, , and Sugata Ghosal. Adaptable similarity search using non-relevant information. *Proceeding of the 28th VLDB Conference*, August 2002.

[53] Lippman A Vasconcelos N. Bayesian relevance feedback for content-based image retrieval. *IEEE Workshop CBAIVL*, 2000.

[54] P. Wan and M. H. Wong. Efficient and robust feature extraction and pattern matching of time series by a lattice structure. *Proceedings of the 2001 ACM CIKM*, pages 271–278, 2001.

[55] Changzhou Wang and X. Sean Wang. Supporting content-based searches on time series via approximation. *Processings of the 12th International Conference on Statistical and Scientific Database Management*, 69-81, 2000.

[56] S. F. Wong and M. H. Wong. Efficient subsequence matching for sequences databases under time warping. *IDEAS*, 2003.

[57] B. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar sequences under time warping. *IEEE ICDE*, pages 201–208, 1998.

[58] Byoung Kee Yi, H. V. Jagadis, and C. Faloutsos. Supporting fast search in time series for movement patterns in multiple scale. *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*, pages 251–258, 1998.

[59] Xiang Sean Zhou and Thomas S. Huang. Exploring the nature and variants of relevance feedback. *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 2001.

Publication

- **Siu Fung Wong** and Man Hon Wong. Efficient Subsequence Matching for Sequences Databases under Time Warping. In *Proceedings of 7th International Database Engineering and Applications Symposium (IDEAS 2003)*, Hong Kong, China, Jul 2003.
- **Siu Fung Wong** and Man Hon Wong. Fast Subsequence Matching under Time Warping. *Technical Report, 2003.* (Submitted for Journal Publication)
- **Siu Fung Wong** and Man Hon Wong. Relevance Feedback for Time Series Retrieval under Time Warping. *Technical Report, 2003.* (Submitted for Journal Publication)