

Feature Extraction and Pattern Matching in Time Series Data

WAN Po Man Polly

Supervised by
Prof. WONG Man Hon

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

©The Chinese University of Hong Kong

August 2001

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.



To my parents for their love and care.

Feature Extraction and Pattern Matching in Time Series Data

Submitted by

WAN Po Man Polly

for the degree of Master of Philosophy
at The Chinese University of Hong Kong in June 2001

Abstract

The efficiency of searching scaling-invariant and shifting-invariant shapes in a set of massive time series data can be improved if searching is performed on an approximated sequence which involves less data but contains all the significant features. However, commonly used smoothing techniques, such as moving averages and best-fitting polylines, usually miss important peaks and troughs and deform the time series. In addition, these techniques are not robust, as they often require users to supply a set of smoothing parameters which have direct effect on the resultant approximation pattern. To address these problems, an algorithm to construct a lattice structure as an underlying framework for pattern matching is proposed in this dissertation. As inputs, the algorithm takes a time series and the required level of details. The algorithm then identifies all the important peaks and troughs (known as control points) in the time series and classifies the points into appropriate layers of the lattice structure. The control points in each layer of the structure form an approximate pattern and yet preserve the overall shape of the original series with an approximation error lies within a certain bound. The lower the layer, the more precise the approximate pattern is. Putting in

another way, the algorithm takes different levels of data smoothing into account. Also, the lattice structure can be indexed to further improve the performance of pattern matching.

The technique of finding patterns in time series using control points can also be applied to the findings of meaningful patterns in bar charts of stock data. Instead of containing one data point per time unit, two data points per time unit are used to represent a bar in bar charts. These two data points may represent the highest and the lowest stock price on a particular day or within a week. Identifying chart patterns is a form of technical analysis in the stock market. The formation of chart pattern is highly flexible as long as the overall shape is preserved. In this dissertation, a novel approach that automatically identifies a set of typical chart patterns using control points as the basis for constructing the upper and lower trendlines is proposed. Having trendlines partitioned into smaller trendline pairs, a progressive pattern classification is then carried out.

在時間序列數據中選取特徵及配對形狀

作者 溫寶敏

香港中文大學 二零零一年六月

摘要

在一個涉及龐大時間量的序列數據中，對其縮放變及移位變不變的形狀進行搜尋，搜尋的效率是可以改進的。如果搜尋是在一個牽涉較少數據但保留重要特徵的約略序列上進行的話，效率將能提升。日常用的平滑技術，例如移動平均線及最合適多線段，一般都忽略了重要的凹凸點而破壞時間序列之原有形狀。與此同時，平滑技術需要用戶提供平滑參數，而這些參數對產生出來的約略形狀有直接影響。所以，這些技術並不穩健。本論文針對這些問題提出一種算法，為形狀配對建立一個組合格子結構，作為形狀配對的底層框架。用戶需要輸入一個時間序列與及所需要的細節程度，此算法根據這些輸入辨認出這個時間序列所有重要的凹凸點（支配點）並把這些點歸類於組合格子結構的合適層。在這結構中，被歸類於同一層的支配點形成一個保留了原來序列的整體形狀，而其誤差少於某一個範圍的約略形狀。運用越低層的支配點，所形成的約略形狀就越準確。換句話說，此算法能將不同程度的數據平滑都處理妥當。還有，這個組合格子結構可進行索引以提高形狀配對的性能。

運用支配點去找尋時間序列中的形狀這種技術也能應用在股票棒形圖中，可以把其有意義的形狀找尋出來。在棒形圖中，每一個時間單位包含兩個數據點，與在時間序列中只包含一個數據點不同。此兩個數據點可以代表每日或每週的最高及最低股票價格。股票價格的變化通常形成有意義的形狀，從棒形圖中找尋這些有意義的形狀是股票市場的一種專業分析。同一個圖表形狀的形成是高度靈活的，只要它的大致形狀得以保持便行。這論文提出一個嶄新的方

法，嘗試運用支配點去建立較高與較低走勢線的基礎，從而自動分辨圖表的形狀。將走勢線分拆成較小的走勢線對，如此一個漸進式的形狀歸類法便能夠進行。

Acknowledgments

This dissertation could not be done without the help and support of many people. I am happy to take this opportunity to thank them.

First, I would like to thank my final year project and Master degree supervisor, Prof. Wong Man Hon, for his guidance and encouragement throughout my study. I treasure his insights and criticism which pointed me to the right direction of research. This dissertation could not have been done without his guidance.

I would also like to thank Professor Ada Fu and Professor John Lui, who are members of my dissertation committee, for their invaluable comments to my work.

I would like to express my gratitude to my colleagues working in the same office. They are Mr. Cheung Chak Chung Ray, Mr. Leong Monk Ping Norris, Mr. Leung Ka Ho Ivan, Mr. Sze Chin Ngai Cliff, Mr. Wong Hiu Yung, and Miss Yuen Wing Seung. I thank them all for their support and giving me happiness in these two years. In particular, I would like to thank Norris for his love and care.

Finally, I would like to give special thanks to my family. I thank my mother Susanna for her warmth and understanding, my father Raymond for his support and my sister Jenny for taking care of our parents while I lived in school.

Contents

Abstract	i
Acknowledgements	v
Contents	vi
List of Figures	x
List of Tables	xiv
1 Introduction	1
1.1 Motivation and Aims	1
1.2 Organization of Thesis	5
2 Literature Review	6
2.1 Dimensionality Reduction	6
2.1.1 Fourier Transformation	6
2.1.2 Wavelet Transformation	8
2.1.3 Singular Value Decomposition	10

2.2	Searching Sequence Similarity with Transformation	11
2.2.1	Time Warping	11
2.2.2	Amplitude Scaling and Shifting	14
2.3	Data Smoothing and Noise Removal	18
2.3.1	Piecewise Linear Segmentations	18
2.3.2	Approximation Function	21
2.3.3	Best-fitting Line	23
2.3.4	Turning Points	24
3	Time-Series Searching with Scaling and Shifting in Amplitude and Time Domains	25
3.1	Representation	25
3.1.1	Control Points	26
3.1.2	Lattice Structure	28
3.1.3	Algorithm on Lattice Construction	31
3.2	Pattern Matching	32
3.2.1	Formulating the Problem of Similarity	35
3.2.2	Error Measurement	38
3.3	Indexing Scheme	39
3.3.1	Indexing with scaling and shifting proposed by Chu and Wong	40
3.3.2	Integrating with lattice structure	41
3.4	Results	43

4	Chart Patterns Searching for Chart Analysis	47
4.1	Chart Patterns Overview	47
4.1.1	Reversal Patterns	49
4.1.2	Continuation Patterns	52
4.2	Representation	53
4.2.1	Trendline Preparation	54
4.2.2	Trendline Pair	59
4.3	Three-Phase Pattern Classification	66
4.3.1	Phase One: Trendline Pair Classification	66
4.3.2	Phase Two: Patterns Merging and Rejection	74
4.3.3	Phase Three: Patterns Merging of Unclassified and Un-merged Trendline Pairs	89
4.4	Results	90
5	Conclusion	100
A	Supplementary Results	103
A.1	Ascending Triangle	103
A.2	Descending Triangle	104
A.3	Falling Wedge	106
A.4	Head and Shoulders	107
A.5	Price Channel	109
A.6	Rectangle	110

A.7 Rising Wedge	112
A.8 Symmetric Triangle	113
A.9 Double Bottom	113
A.10 Double Top	116
A.11 Triple Bottom	118
A.12 Triple Top	120
Bibliography	122
Publications	128

List of Figures

1.1	Sequence B is similar to a subsequence of A	2
1.2	Shape formed by different number of turning points.	3
2.1	An example of the time warping path.	12
2.2	An example of projecting subsequences to a 2-D plane.	17
2.3	Projected lines between segmented query sequence Q and part of the reference sequence R [Keo97].	20
2.4	Creating bit-string using a sliding window with equi-space grid as proposed in [KP99a].	21
2.5	Curve Fitting Algorithm by [SZ96]	22
3.1	Similar subsequences of different lengths.	26
3.2	Three local maxima: points A , B and C	28
3.3	Example of a lattice structure of a series with twelve control points.	29
3.4	Control points used to form features from the lattice structure of Figure 3.3	29
3.5	Lattice Structure Construction Algorithm.	33

3.6	A peak-to-peak problem after a point is chosen for breaking up a segment.	34
3.7	Examples of scaling and shifting transformation.	35
3.8	Average standard deviations of subsequences' time-domain and amplitude-domain values against the size of subsequences (size of sliding window).	42
3.9	The "W" shape found in layer 2 of American Online's Lattice structure(as shown in chart (a)) was used as a query pattern. Query results are marked by crosses and are highlighted and are shown in charts (b) to (h).	45
3.10	User time versus Relative Minimum Error of sequential search and our indexing scheme	46
4.1	An example of Bar Chart using the stock price of Sun Microsystems.	48
4.2	Reversal Patterns.	50
4.3	Continuation Patterns.	51
4.4	Succession of downtrend and uptrend lines	56
4.5	Rating Relative Point Algorithm.	58
4.6	Trendline Establishing Algorithm.	60
4.7	Successions of upper and lower trendlines drawn using <i>min_dist</i> =15 days and <i>max_dist</i> = 35 days.	61
4.8	Notations for relative highs and relative lows which form a price envelope.	62

4.9	Examples of trendline pairs projected from a trendline formed by (RH_1, RH_2)	63
4.10	Pseudocode of Generating Trendline Pairs	64
4.11	Continual conditions in which the $(j + 1)$ -th lower trendline can be retrieved and potentially form a trendline pair with the i -th upper trendline	65
4.12	An example on how <i>first</i> is updated	65
4.13	Valid Head and Shoulders pattern adapted from [sto].	70
4.14	Pseudocode of classifying trendline pairs into chart patterns	75
4.15	Different merging of trendline pairs who have been classified to the same pattern.	76
4.16	Translation of two consecutive trendlines such that the first point touches the origin.	82
4.17	Examples of Symmetrical Triangle found in SP500. (Upper: Halliburton Co, Lower: Visteon Corp)	94
4.18	Examples of Descending Triangle found in SP500. (Upper: US West, Lower: Lucent Technology)	94
4.19	Examples of Ascending Triangle found in SP500. (Upper: Illinois Tool, Lower: Kerr Mcgee)	95
4.20	Examples of Falling Wedge found in SP500. (Upper: Jefferson Pilot, Lower: Wells Fargo)	95
4.21	Examples of Rising Wedge found in SP500. (Upper: Eaton; Lower: Praxair, Inc.)	96

4.22	Examples of Price Channel found in SP500. (Upper: Marsh & McLennan; Lower: Unocal Corp.)	96
4.23	Examples of Rectangle found in SP500. (Upper: Schlumberger Ltd.; Lower: PACCAR Inc.)	97
4.24	Examples of Double Top found in SP500. (Upper: Occidental Petroleum; Lower: GTE)	97
4.25	Examples of Double Bottom found in SP500. (Upper: Abbott Labs; Lower: Sempra Energy)	98
4.26	Examples of Triple Top found in SP500. (Upper: St Paul.; Lower: Masco Corp.)	98
4.27	Examples of Triple Bottom found in SP500. (Upper: Ashland; Lower: St Jude Medical)	99
4.28	Examples of Head and Shoulders found in SP500. (Upper: Apply Computer; Lower: Bestfoods)	99

List of Tables

4.1	Studying chart pattern features by investigating the slopes.	67
4.2	Patterns overlap with patterns that bear the shape of Head-and-Shoulders Bottom.	72
4.3	Patterns overlap with patterns that bear the shape of Head-and-Shoulders Top	73
4.4	Greedy Approach for Merging Upper Trendlines.	78
4.5	Greedy Approach for Merging Lower Trendlines.	79
4.6	Merging unclassified and unmerged trendline pairs which can form Rectangle.	91
4.7	Merging unclassified and unmerged trendline pairs which can form Price Channel (Slopes Up).	92
4.8	Merging unclassified and unmerged trendline pairs which can form Price Channel (Slopes Down).	93

Chapter 1

Introduction

1.1 Motivation and Aims

Time series account for a significant portion of data stored in business, medical, engineering, and social science databases. A time series Q is a sequence of real numbers $\{q_1, q_2, \dots, q_n\}$ collected regularly in time, each of which represents a value at a particular instance. For example, stock prices and weather quantities such as temperature and humidity can be in the form of sequences, so they are time series data.

Studying patterns of time series has been an active research topic [AFS93, FRM94, RM97, KJF97, APWZ95, LW98, CW99, CLW98, CF99, YJC98b, YJC98a, KP99b, PWZP00, PCYH00, SZ96]. The most remarkable application of identifying patterns in time series is stock analysis. It is believed that the uptrend and downtrend lines of a stock may indicate its future trend [Sch98]. As a time series database usually involves a large amount of data (such as historical stock data), brute force searching for certain shapes among those datasets is unrealistic. To speedup the searching process, a subsequence extracted by a sliding window can

be indexed using a multi-dimensional indexing structure such as R -tree [Gut84]. However, R -tree or other multi-dimensional indexing structures cannot efficiently handle high dimensional data (known as the dimensional curse), that is, when the subsequence is too lengthy [WSB98]. Some researchers proposed to reduce the dimension of a subsequence before indexing. Each subsequence is first transformed into a feature point, typically by the Fourier transform [AFS93, FRM94], the Wavelet transform [CF99], or the Singular Value Decomposition [KJF97]. The feature points are then indexed by a high dimensional indexing structure such as R^* -tree. However, the transformed sequence may not be applicable for shape matching, especially when scaling and shifting are considered. As an example, Figure 1.1 shows two sequences: $A=\{10, 12, 14, 10, 6, 7, 8, 7\}$ and $B=\{5, 7, 3, 4\}$ with equal interval between any two consecutive data values. Scaling up sequence B by a factor of 2 in both amplitude and time dimensions yields a sequence $\{10, 12, 14, 10, 6, 7, 8, 7\}$ which is identical to A . However, one cannot tell this particular shape similarity simply by inspecting the transformed coefficients.

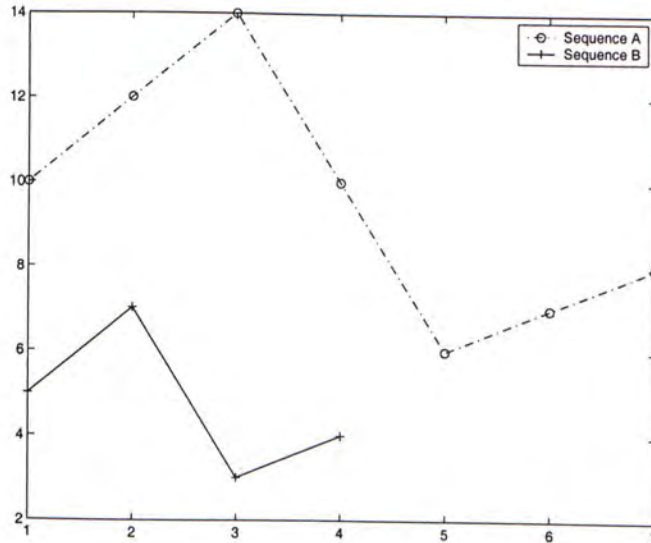


Figure 1.1: Sequence B is similar to a subsequence of A .

Without data pre-processing, the efficiency of searching scaling-invariant and shifting-invariant shapes in a set of massive time series can be very low. Performance can be improved if searching is performed on an approximated sequence which involves less data but contains all the significant features of the original sequence. However, the shape formed by a noisy sequence is subjective and depends on the number of data points used for approximation. Using American Online stock close price from 6/16/1999 to 12/14/1999, Figure 1.1 shows two different shapes of a time series formed by different number of turning points (the two sets of turning points are selected according to the algorithm proposed in this dissertation). Using fewer data points (as in Figure 1.1(a)) the shape formed by joining these points is coarser and the approximation error is greater. However it may be already informative enough for some applications.

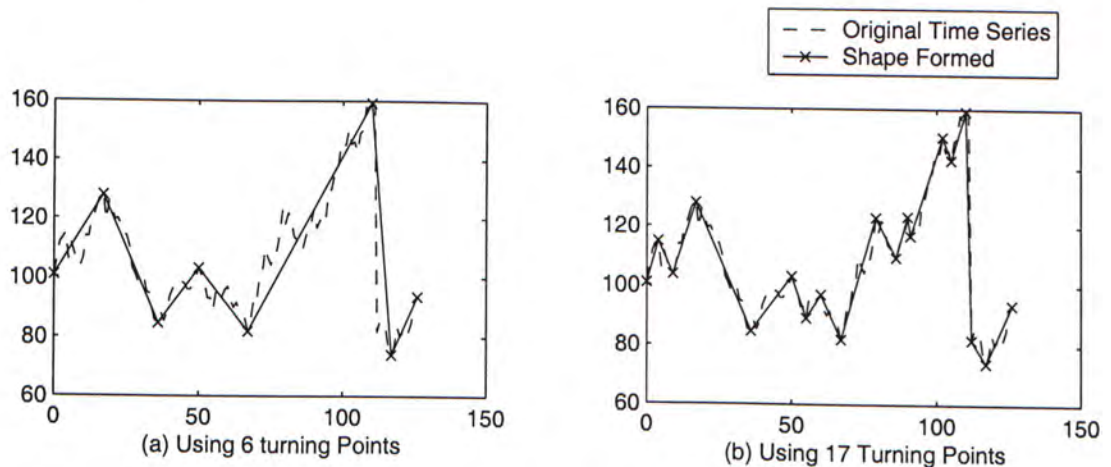


Figure 1.2: Shape formed by different number of turning points.

In this dissertation, a method for progressive sequence approximation through constructing a lattice structure is proposed. The lattice structure is built starting from the topmost layer which consists of data points forming the coarsest shape. Each layer is associated with a maximum approximation error which can also be

interpreted as the level of details that the chosen set of data points can represent the original time series. The lower the layer, the smaller the error as the generated shape will be continuously refined during the course of proceeding towards the lowest layer. Approximated sequences with a large approximation error may be informative enough while those with small approximation error may be too detailed and contain noises. Therefore, depending on the user's requirement, finding patterns of interest can be confined to a set of layers which are within an approximation error bound. To further enhance the searching performance, an integration of the lattice structure with the indexing structure reported by [CW99] is proposed.

Much research has been focused on finding similar patterns among a uni-value time series data, that is, each time unit only contains one value. In an attempt to find meaningful chart patterns in stock data, the proposed patterns searching technique was applied to multi-value time series. Stock data, as a form of time series data, can be presented in different types of charts. Two commonly used charts are close-only charts and bar charts. They differ from the type and number of price values used in each time unit. Close-only charts are based on closing values and ignore high and low price information; bar charts use both high and low prices (or even closing price) at every time unit. The price series used in a close-only chart, such as the one previously shown in Figure 1.1, can be regarded as the simplest form of time series whilst bar chart can be considered as multi-value time series. Being considered as a special case of the basic time series, the general technique of finding patterns of interests in a basic time series can also be applied to pattern searching in a bar-chart.

Thirteen prototype patterns commonly used in technical analysis can be found using the method proposed in this dissertation. Those patterns are Falling Wedge, Rising Wedge, Head and Shoulders Top, Head and Shoulders Bottom, Double

Top, Double Bottom, Triple Top, Triple Bottom, Ascending Triangle, Descending Triangle, Symmetrical Triangle, Pricing Channel, and Rectangle. The identification process first spots out a set of significant turning points upon which trendlines can be drawn. Then a progressive pattern classification based on the established trendlines is carried out.

1.2 Organization of Thesis

The dissertation is organized as follows.

In Chapter 2, we will give a literature review on different approaches for time series searching and indexing methodologies. These approaches are classified into three categories namely dimensionality reduction, sequence similarity with transformation, and similarity based on sequence approximation.

In Chapter 3, we will describe our approach to find similar sequences by means of turning points. First of all, a lattice structure, which is the data representation, is defined. Under this lattice structure, we then define subsequence scaling, translation and best matches, and an indexing scheme is proposed. Finally, experimental results using real-life stock data are given.

In Chapter 4, we show how to utilize turning points to find typical chart patterns used in technical analysis. We first give an overview to common chart patterns found in a bar chart. Then, a strategy in building trendlines using turning points are given. A three-phase chart pattern classification is then presented. Finally, examples of different chart patterns found in real-life chart pattern are given.

A conclusion will be given in Chapter 5.

Chapter 2

Literature Review

In this chapter, three main research threads of sequence matching and the corresponding indexing methodologies are discussed. Firstly, several feature extraction methods on reducing the sequence's dimension for the ease of indexing are discussed. Secondly, different approaches on finding similar sequences invariant to some transformations are introduced. Finally, sequence matching approaches based on sequence approximations are discussed.

2.1 Dimensionality Reduction

2.1.1 Fourier Transformation

Various approaches have been proposed for time series searching. An earlier approach uses the Discrete Fourier Transform (DFT) to map time sequences to the frequency domain [AFS93]. Given a time series $x = (x_0, x_1, \dots, x_{n-1})$, its

Fourier coefficients are given by:

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

where j is the imaginary unit ($j = \sqrt{-1}$). The original time series can be recovered by the inverse transform:

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

X_f are complex numbers except $f=0$. Using Parseval's theorem, they observed that the Euclidean distance in the time domain is the same as that in frequency domain. Therefore, coupled with the conjecture that first few Fourier coefficients contain most of the energy, an effective index (so called F -index) with a low dimensionality can be built using R^* -trees. This method guarantees no false dismissals but it may cause false alarms. Therefore, a post-processing is needed to filter unqualified data sequences.

F -index was further developed and an efficient indexing method to locate one dimensional subsequences was proposed in [FRM94]. They used a sliding window of size w over the data sequence and extracted its features by using the amplitudes of the first few coefficients of the w -point DFT. This results in a trail in the feature space. Instead of storing individual points of the trail in an R^* -tree, they proposed to divide the trail of a given data sequence into sub-trails and represent each of them with its *minimum bounding (hyper)-rectangle (MBR)*. Such a sub-trail indexing structure is referred to as ' ST -index'. Therefore, only a few MBRs are needed instead of storing thousands of points of a given trail. As in [AFS93], false alarms are still possible.

An enhancement of feature extraction and matching method was proposed in [LYC96]. They use the correlation coefficient as an alternative similarity measure between the target sequence and the stored sequence. An adaptive scan

on the extracted features of the stored sequences is performed based on the target sequence. The idea is to select first the subset of features with the largest filtering effect to perform the matching using correlation. Then the next most discriminating set of features is used for matching. This process is iterated until all of the features are exhausted. Although false hit is impossible, misses are possible when the correlation coefficient between low resolution versions of two sequences below the query threshold even if the correlation coefficient between the corresponding full sequences is above. Also, it is expensive to search through all features hierarchically especially when the sequences are long.

Using the DFT, a first indexing method that can handle moving average and time warping was proposed in [RM97]. For each query, a transformation is given. A new index is built in real time by applying this transformation to all feature points extracted by the DFT. However, it is costly to build the index for each query and thus affect the query response time.

2.1.2 Wavelet Transformation

Besides the DFT, the Discrete Wavelet Transform (DWT) has also been proposed to reduce the dimensions of feature vectors in time series. The pioneered work by [CF99] used the Haar Wavelet Transform in which the first few coefficients are kept for similarity searching. Given a sequence $x = (x_0, x_1, \dots, x_{n-1})$, the process of the one-dimensional Haar Wavelet Transform begins with averaging the sequence data together pairwise, in order to get a new sequence which is half of the size of the original sequence x . To recover the original sequence, some detail coefficients, which are the differences between the average values and the original values, are computed. The process continues until a resolution of one is reached. Given a sequence $x = (9, 7, 3, 5)$, the process of the Haar Wavelet

Transform is as follows:

Resolution	Averages	Detail Coefficients
4	(9 7 3 5)	
2	(8 4)	(1 -1)
1	(6)	(2)

They proposed to use the DWT because it has better pruning power and lower complexity compared with the DFT. Also, it can deal with vertical shifts of sequences into the similarity model easily.

Making use of wavelet decomposition, a wavelet-based tree structure, termed *TSA*-tree, for multilevel trend and surprise (sudden changes) queries on time sequence data was proposed in [STZ00]. A single-level one-dimensional wavelet decomposition is performed on a sequence to get the trend sequence and surprise sequence which are respectively the approximation coefficients vector and the detail coefficients vector of the wavelet decomposition. Thus, a *TSA* tree can be constructed by applying the decomposition repeatedly on the trend sequence obtained at each level. As the *TSA*-tree is large, they proposed two methods to optimize the tree structures by dropping nodes or coefficients. The drawbacks of these optimization is that surprises may lose and errors are induced during time-series reconstruction.

In the view that the Haar transformation has a better performance, using this transformation to compact time sequences was also proposed in [KS01]. Their emphasis is on creating a grid of trees using wavelet coefficients to handle variable length queries. In the grid, each row corresponds to the set of trees built using a window size ranging from 2^a to 2^b where $a < b$ and 2^a is the minimum possible length of a query. Each query having length $k2^a$ is partitioned into separate pieces with sizes starting from $2^a, 2^{a+1}, \dots$ such that the summation of all partitions equals to the length of the query. Although they can handle variable length

efficiently, the query size is restricted at the powers of two and a large storage space is needed for keeping a grid of trees.

2.1.3 Singular Value Decomposition

Besides the DFT and the DWT, another dimension reduction method is by the Singular Value Decomposition or *Karhunen Loeve* Transform. The use of the Singular Value Decomposition for querying time sequences was pioneered by [KJF97]. The idea is to reduce the dimensionality of the data set using projection. The data is projected into a sub-space and the problem is to find the best sub-space for projection. The data in a few condensed dimensions are then indexed to support fast retrieval for a given query. Suppose there are M sets of time series x_1, x_2, \dots, x_M , each of which is of length n or in other words, $x_1, x_2, \dots, x_M \in \mathbb{R}^n$. The problem is to find m out of n orthonormal vectors $u_1, u_2, \dots, u_n \in \mathbb{R}^n$ such that the following expected mean square error Υ over M time series is the minimum

$$\begin{aligned} \Upsilon &= \sum_{i=m+1}^n u_i^T E [(x - d)(x - d)^T] u_i \\ &= \sum_{i=m+1}^n u_i^T A u_i \end{aligned}$$

where d is a displacement vector in \mathbb{R}^n . The minimization of Υ yields $d = E[x]$ and $Au_i = \lambda_i u_i$ where u_i and λ_i are respectively the eigenvector of A and its corresponding eigenvalues. To handle dynamic database using the SVD, efficient methods of updating of the SVD-based index were discussed in [KAAS99].

Combining clustering with the SVD was proposed in [TCL98] to improve the efficiency in processing approximate nearest-neighbor queries. Their method was motivated by the insight that subdividing heterogeneously distributed vectors into groups gives more efficient representation. This method first partitions the data set using a clustering technique and then SVD is applied to vectors in each

cluster to produce a vector space of transformed features with reduced dimensional. Although it requires less dimensions than the SVD, there is an overhead in updating the clusters as well as the SVD dimensions for dynamic database.

2.2 Searching Sequence Similarity with Transformation

2.2.1 Time Warping

The technique of dynamic time warping is first used in the field of speech recognition. Using dynamic programming as the basis for both isolated and connected word recognition was first presented in [RL90]. Later, [BC96] proposed a pattern detection algorithm in time series based on the dynamic time warping technique used in the voice recognition. The pattern recognition process involves searching a time series, S , for a template, T , where $S = (s_1, s_2, \dots, s_n)$ and $T = (t_1, t_2, \dots, t_m)$. The sequences S and T can be arranged to form an n -by- m grid where each grid point (i, j) corresponds to an alignment between elements s_i and t_j . A time warping path, $W = (w_1, w_2, \dots, w_p)$, aligns the elements of S and T , such that the distance between them is minimal. In other words, W is a sequence of 2- D grid points as shown in Figure 2.1.

Before defining the problem of dynamic time warping, a distance measure between two elements is needed. Two commonly used distance functions, δ , are the absolute difference and the squares of the difference:

$$\begin{aligned}\delta(i, j) &= |s_i - t_j| \\ \delta(i, j) &= (s_i - t_j)^2\end{aligned}$$

Then, the dynamic time warping problem as a minimization over potential time

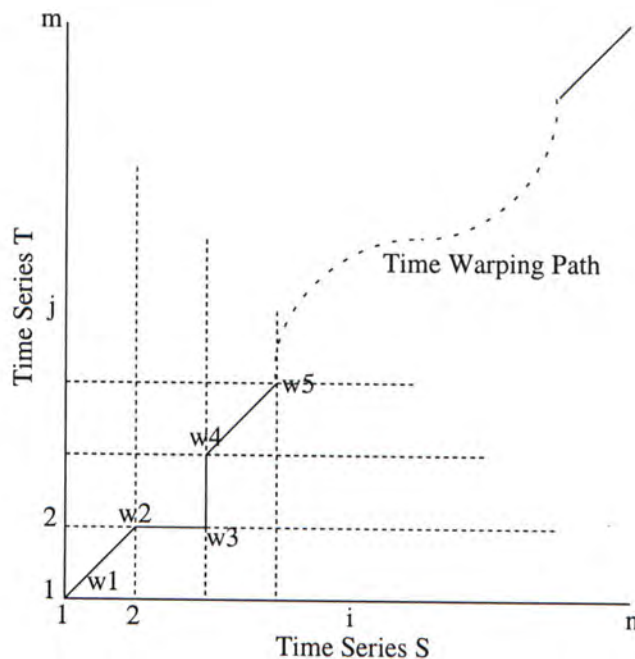


Figure 2.1: An example of the time warping path.

warping paths based on the cumulative distance for each path can be defined

$$DTW(S, T) = \min_W \left[\sum_{k=1}^p \delta(w_k) \right]$$

Out of concern for efficiency, it is important to reduce the space of possible warping paths, therefore the dynamic programming formulation is based on the following recurrence relation, $\Upsilon(i, j)$, which defines the cumulative distance between each pair of points as follows

$$\Upsilon(i, j) = \delta(i, j) + \min[\Upsilon(i-1, j), \Upsilon(i-1, j-1), \Upsilon(i, j-1)].$$

Computing the cumulative time warping distance in a sequential manner is an expensive task, two methods to speed-up sequential scanning on time series is proposed in [YJC98b]. Based on FastMap, they make use of the given distance measures to map sequences into points in a k - d space followed by building an

indexing structure. Another technique is to determine a cheaply computed lower bound on the original distance function, which can be used as a filter to discard non-qualifying sequences quickly. As addressed in the paper, false dismissal is possible using the FastMap method. In addition, the choice of dimension k is critical for the best searching performance.

To address the problem of false dismissal in time warping, a new indexing technique using a suffix tree and employing lower-bound distance functions to filter dissimilar subsequences were proposed in [PCYH00]. As branch pruning in the index structure cannot help if there is no common subsequences, they introduced the concept of categorization to increase the number of common subsequences. The first method is an equal-length categorization which is fast but information about value and frequency distributions of the sequences may be lost. The second method is a maximum-entropy categorization. This can minimize the information loss but it is not easy to determine the suitable number of categories.

A new distance function D_{tw-lb} that consistently underestimates the time warping distance between two sequence was proposed in [KPC01]. Their time warping distance, however, does not accumulate the time warped distance between two elements but just only the maximum time warping distance. Given two sequences S and Q , their time warping distance, $D_{tw}(S, Q)$, is defined recursively as follows:

$$\begin{aligned}
 D_{tw}(\langle \rangle, \langle \rangle) &= 0 \\
 D_{tw}(S, \langle \rangle) &= D_{tw}(\langle \rangle, Q) = \infty \\
 D_{tw}(S, Q) &= \max \left\{ \begin{array}{l} |First(S) - First(Q)| \\ \min \left\{ \begin{array}{l} D_{tw}(S, Rest(Q)) \\ D_{tw}(Rest(S), Q) \\ D_{tw}(Rest(S), Rest(Q)) \end{array} \right. \end{array} \right.
 \end{aligned}$$

where $First(S)$ and $Rest(S)$ denote the first element in S and the subsequence of S excluding $First(S)$. For each sequence S , a four-tuple feature vector is extracted and indexed in an R^* -tree. The feature vector contains the first element, the last element, and the elements with the largest and smallest value in S . With the lower bound distance function

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

and the the following inequality

$$D_{tw}(S, Q) \geq D_{tw-lb}(S, Q),$$

a candidate set can be returned from the tree by enlarging the four-tuple feature vector of the query Q . As there are false alarms, post-processing of the candidates are needed. Although an index-based time warping without false dismissal is made possible, their time warping distance function is weaker than the traditional one as they only consider the maximum distance among all the time warped points and neglect the other time-warped distances which may be essential in determining similarity.

2.2.2 Amplitude Scaling and Shifting

Besides time warping, another definition of similarity which is based on scaling and shifting transformation along the amplitude axis is also considered in some applications. Proper amplitude scaling and offset translation are necessary before determining if the two sequences are similar. An intuitive way is to apply a global normalization to the time sequence. However, such a global scaling will make the

transformation easily affected by the chosen scale points which may be outliers or noises.

To minimize the bias of global normalization, a similarity search with scaling and translation based on local subsequence normalization was discussed in [ALSS95]. In general, two sequences are considered similar if they have enough non-overlapping time-ordered pairs of subsequences that are similar. Two normalized subsequences are called ϵ -similar if the absolute differences of two corresponding elements are bounded by a tolerance ϵ . The matching process is divided into three stages, namely “atomic” subsequence matching, long subsequence matching, and sequence matching. In the first stage, every locally normalized subsequence of size ω are indexed into an R^* -tree. Therefore, all atomic subsequence pairs which are ϵ -similar can be easily retrieved by spatial join. The second stage employs a fast algorithm for stitching atomic matches to form long subsequence matches, gap is allowed to exist between the atomic matches. In the third stage, the maximal length match in sequences is determined.

A definition of sequence similarity based on the slope of sequence is discussed in [CLW98]. By using a sliding window, each extracted subsequence is translated into a binary string according to the sign of the slope of two consecutive data within the subsequence. All binary strings are then indexed using a hashing algorithm. In a similar fashion, a query will also be transformed into a bit-stream for searching potential candidates in the hash table. The algorithm can also be extended to handle linearly scaled similar subsequences. However, the range of scaling factors has to be pre-specified before the hash table is built, thus decreasing the flexibility if a different set of scaling factors are needed at different time.

A projection algorithm for similarity matching which considered vertical shifting and linear scaling was proposed in [LW98]. Besides the Euclidean distance,

their criteria for similarity sequences also takes “individual” distance into account as they argued that the Euclidean distance alone cannot ensure similarity in shape. The distance between each corresponding data pair from two sequences must not exceed a user defined threshold σ which is the individual distance. Given a subsequence of $A' = (a_1, a_2, \dots, a_N)$ and a query sequence $Q = (q_1, q_2, \dots, q_N)$, they proved that the following inequality must hold:

$$-2\sigma \leq (a_i - a_1) - (q_i - q_1) \leq 2\sigma \quad \forall i, 2 \leq i \leq N$$

Using this inequality, N -dimensional subsequences can be projected to an $(N-1)$ -dimensional space. All subsequences that are similar to a basic query subsequence must have their projection points inside a bounding box centered at the query data projection point. The overall strategy is to first partition a database sequence $A = (a_1, a_2, \dots, a_n)$ into t segments where each segment contains n/t data elements. Then a sliding window of size ω is slid over each segment and also the query sequence to extract subsequences. All subsequences extracted from each segment is projected to a plane forming a search space. Therefore, t search spaces are created. Using $\omega = 3$, the process of projecting subsequences of the first segment S_1 of sequence onto a search space is depicted in figure 2.2. Although the author suggested to use a small t , a query sequence cannot be answered if it span over two segments even there exists a similar subsequence spanning over two segments. Finally, they proposed to use turning points for handling linear scaling problem but, as mentioned in the paper, the algorithm is vulnerable to data which is noisy at high frequencies.

An indexing scheme which can efficiently search for subsequences that are similar to a query after scale-shift transformation was proposed in [CW99]. A database sequence $A = (a_1, a_2, \dots, a_n)$ is said to be similar to a query sequence $Q = (q_1, q_2, \dots, q_n)$ if there exists a scaling factor s and a shifting offset t such

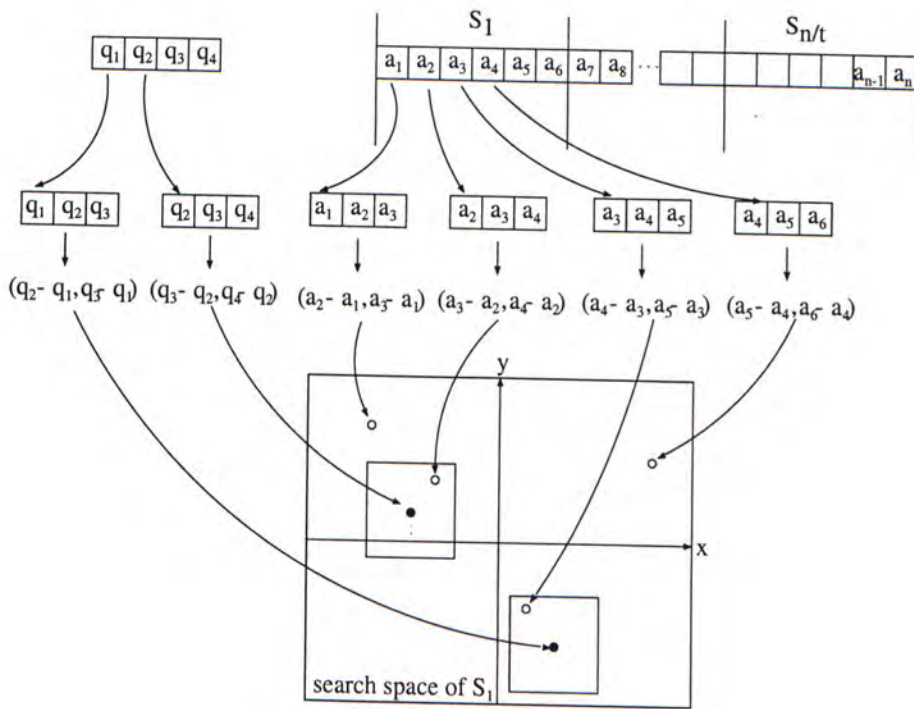


Figure 2.2: An example of projecting subsequences to a 2-D plane.

that $\sqrt{\sum_{i=1}^n (q_i s + t - a_i)^2} \leq \epsilon$ where ϵ is a user-specified error bound. The whole searching algorithm is composed of three main steps. The first step is to extract a set of subsequences using a window of length n which slides over each data sequence. Then these subsequences will be transformed by the *Shift Eliminated Transformation* and the resulting transformed subsequence is indexed into an R -tree. In this transformation $T_{se}(\vec{p})$, a point \vec{p} (or a sequence with length n) is projected to the plane which passes through the origin $\vec{0}$ and has its normal vector in the direction of $\vec{N} = (1, 1, \dots, 1)$ in \mathbb{R}^n . In the second step, a query Q will also be transformed and let the transformed query be $T_{se}(Q)$. Searching begins from the root node of the R -tree built by transformed subsequences. At each level, every node's MBR is enlarged by ϵ and if the line $tT_{se}(Q)$ penetrates this enlarged MBR, then the node is traversed. When a leaf node is reached, the original sequence is retrieved if the minimum point line distance between the corresponding transformed sequence and the line $tT_{se}(Q)$ is within ϵ . The last step is to post-process all subsequences retrieved in the second step. Those subsequences whose scaling factors and shifting offsets are within user specified bounds are reported. Although amplitude domain scaling and shifting are the sole consideration in this indexing scheme, our proposed method which considers time domain scaling and shifting also adopts their indexing scheme but with some modifications.

2.3 Data Smoothing and Noise Removal

2.3.1 Piecewise Linear Segmentations

Piecewise approximation is described as a way of feature extraction, data compaction, and noise filtering. A fast algorithm which allows a variable number of

segments was proposed by [PH74] in 1974. In general, after an arbitrary initial choice on number of segments, segments are split or merged in order to drive the error norm under a pre-specified bound. There has been considerable work using piecewise linear segmentations as the underlying representation in time series pattern matching.

A probabilistic approach for pattern matching using segmentation was proposed in [KS97]. Local features, such as peaks, troughs, and plateaus which are captured from the segmented sequence, are defined using a prior distribution of the expected deformations form a basic template. Global shape information is represented using another prior distribution of the relative locations of the individual features. A probabilistic model then integrates the local and the global information and directly leads to an overall distance measure between sequence patterns based on the prior knowledge. A search algorithm using this distance measure was proposed.

An algorithm deciding the optimal number of linear segments for pattern matching was proposed in [Keo97]. The number of linear segments k is chosen after an intensive experiment on different numbers of segments. For each number k , the *balance of error*, which is the variance in segment error, is calculated. The k with the minimum value of *balance of error* is selected for that sequence. After applying segmentation to all sequences and also the query sequence, the subsequence matching is accomplished by sliding the segmented query Q sequence along the segmented database sequence R . The left edge of Q is anchored with the left side of every segment in R . The distance is based on the variance of the length of the projected lines in between Q and R as shown in 2.3.1. Using this distance measurement, they proposed a method to handle longitudinal scaling. However, prior knowledges on the maximum stretch and shrink are needed. To begin with, two small sub-queries Q_l and Q_r , which are the leftmost and rightmost segments,

are extracted from Q . Two searches are done for each of the sub-queries, one where the sub-query is being stretched to the maximum amount and one where being compressed to the minimum amount. The goodness of fit is computed whenever the stressed or compressed sub-query is aligned with any segment of the reference sequence. The whole query matching is started with the segment having the best goodness of fit with Q_l and Q_r . This similarity model is further extended to handle classification and clustering as proposed in [KP98].

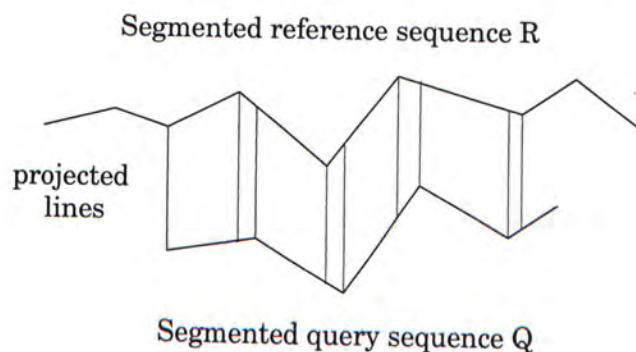


Figure 2.3: Projected lines between segmented query sequence Q and part of the reference sequence R [Keo97].

To model subjectivity in information retrieval, an approach using relevance feedback from the user to adjust the similarity metric was proposed in [KP99b]. Again, segmentation is carried out to every sequence as in [Keo97]. They argued that the Euclidean distance metric does not capture many notions of similarity between time series, in particular, it is sensitive to various distortions like offset translation and amplitude scaling. Depending on the domain and the user, one may wish a query to be sensitive or insensitive to distortions of varying degrees. They addressed this problem by introducing a profile that encodes the user's subjective notion of similarity in a domain.

Previous work using segmentation did not incorporate an indexing scheme.

An indexing algorithm using bins that contains time series subsequences of approximately the same shape was proposed in [KP99a]. After applying segmentation to every sequence, a sliding window with equi-spaced grid is used. Each grid produces a bit decided by the majority sign of the slopes within the grid as shown in Figure 2.3.1. The resulting bit-strings from the window are then indexed using a set of bins. When the distribution of all subsequences into bins is finished, the processing of individual bins takes place. This step simply consists of comparing every pair of items in the same bin and precomputing a distance matrix for the ease of subsequent pruning. Having the same grid placed over the query, it is compared with all bins which are then ordered to improve the efficiency of searching.

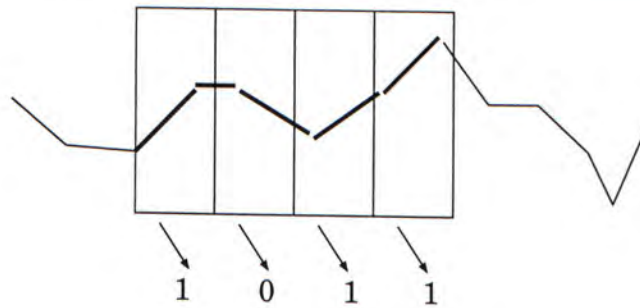


Figure 2.4: Creating bit-string using a sliding window with equi-space grid as proposed in [KP99a].

2.3.2 Approximation Function

Using families of real-valued functions as an approximate representation was presented in [SZ96]. They proposed a curve fitting algorithm which recursively subdivides the sequence into subsequences, each of which is fitted by a curve (or any real-valued functions). The curve-fitting process stops when all points in a subsequence is within an error bound ϵ . They demonstrated the curve fitting algorithm

using linear interpolation which simply takes as its curve the line interpolating the endpoints of a sequence, as shown in Figure 2.5. Using linear interpolation as the curve fitting function, they applied it on actual digitized segments of electrocardiograms (ECG) and suggested an indexing scheme for answering “ $R - R$ interval” query which looks for two prominent peaks with intervals bounded by δ . In summary, they only emphasized on breaking the time series into disjoint regions, each of which is approximated by a function chosen from a predefined family. However, they did not study the problem of general query processing on approximate representations of time series.

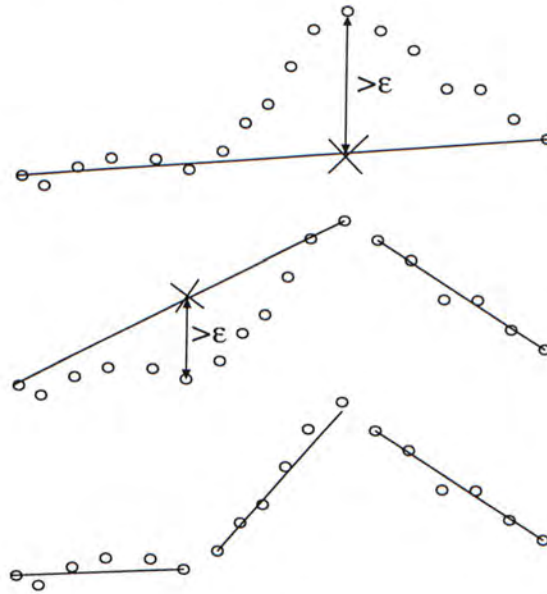


Figure 2.5: Curve Fitting Algorithm by [SZ96]

Two specific approximation methods which are wavelet based and line-fitting based were proposed in [WW00]. For wavelet based approximation, they first use the standard method to obtain an approximation, and then compare each subsequence of the approximation with the corresponding subsequence from the original one, and calculate distance between them. If the distance is greater than

a threshold, more wavelets in that region are included into the approximation. For the line-fitting approach, a linear interpolation approach similar to [SZ96] was proposed. Starting from the first time point A of the series, the farthest point B is found such that distance between each point in-between A and B and the line joining A and B is within a certain boundary. The process continues at B until the end of the sequence is reached. They give some tradeoffs between two approximation methods but again, efficient query processing technique incorporating into sequence approximation was not discussed.

2.3.3 Best-fitting Line

Using best-fitting line to handle movement pattern query was proposed in [YJC98a]. A time series matches a pattern if the series can be converted into a word that match the pattern. The series is first partitioned into consecutive subsequence of some given unit length. For each subsequence, if its distance from its best fitting line is within some given tolerance (a monotonic function of the length of the subsequence), then a letter to which its best fitting line belong is identified. The concatenation of all the letters for all subsequences become the resulting word. The classification of a line into a letter is based on its slope which falls into one of the three pre-specified disjoint ranges that are associated with three letters U (upward), F (flat) and D (downward). Taking the advantage of a dynamic programming technique as well as properties of best fitting lines and tolerance functions, they proposed an efficient algorithm to pre-compute the best fitting lines of all the subsequence and to find the slopes of them if the distances are within the allowed tolerances. They also showed that points belonging to a particular gradient letter are usually clustered roughly into right-angled, isosceles triangular shapes. Thus, instead of storing these points individually, triangles are used to index them.

2.3.4 Turning Points

Using turning points to approximate database sequence was proposed in [PWZP00]. The authors proposed to use first-order turning points (named as Landmark) instead of raw data processing. They introduced a smoothing process called Minimal Distance/Percentage Principle (MDPP) to filter out undesirable landmarks. Given a sequence of landmarks $(x_1, y_1), \dots, (x_n, y_n)$, a minimal distance D and a minimal percentage P , MDPP process removes the landmarks (x_i, y_i) and (x_{i+1}, y_{i+1}) if $x_{i+1} - x_i < D$ and $\frac{|y_{i+1} - y_i|}{(|y_i| + |y_{i+1}|)/2} < P$. Instead of using the Euclidean distance, they proposed a two-dimensional dissimilarity measurement function that considers time drift and amplitude difference separately. Under the same parameters of MDPP smoothing, two sequences of landmarks $L = (L_1, \dots, L_n)$ and $L' = (L'_1, \dots, L'_n)$ where $L_i = (x_i, y_i)$ and $L'_i = (x'_i, y'_i)$, the distance between the k -th landmarks is defined by $\Delta(L, L') = (\delta_k^{time}(L, L'), \delta_k^{amp}(L, L'))$ where $\delta_k^{time}(L, L') = \frac{(x_k - x_{k-1}) - (x'_k - x'_{k-1})}{|x_k - x_{k-1}| + |x'_k - x'_{k-1}|}$ and $\delta_k^{amp}(L, L') = \frac{|y_k - y'_k|}{(|y_k| + |y'_k|)/2}$. The distance between two sequences is $\Delta(L, L') = (\|\delta_k^{time}(L, L')\|, \|\delta_k^{amp}(L, L')\|)$. After the landmarks smoothing process, a set of seven features is extracted from every landmark point. Different transformation will be invariant under different subsets of those features. Depends on users' preference on the set of transformations, only a subset of those features is needed for indexing using S^2 -tree. They also presented the upper bounds and lower bounds for the seven features for the ease of querying landmark sequences. Although strings of features can be indexed, a new indexing structure has to be built if a different set of transformations is considered, The similarity of two sequences highly depends on the MDPP parameters. If two sequences, one of which may be relatively noisier, are considered to be dissimilar under certain MDPP parameters. However, they may be consider similar under another set of MDPP parameters.

Chapter 3

Time-Series Searching with Scaling and Shifting in Amplitude and Time Domains

In this chapter, an approach to find similar sequences which are invariant to scaling and shifting in both amplitude and time domains is presented. Using this approach, similar subsequences are not necessary to have the same length. The idea is to capture critical points and organize them into a lattice structure which encapsulates different levels of details about the original time series. Besides precomputing a lattice structure, an indexing scheme is also employed to enhance the efficiency of evaluating pattern queries.

3.1 Representation

A good representation or approximation of time series data is important for time and memory efficiency of the searching algorithm. Specifically, the complexity

of searching for subsequences which are similar to the query sequence can be very high as subsequences with different lengths may also form such a shape. Therefore, we propose a method to find different sets of control points, which reside in a lattice structure, to represent the original time series data with different error bounds or depth of details. As an example, Figure 3.1 shows two similar sequences (A and B) of different lengths and the corresponding control points.

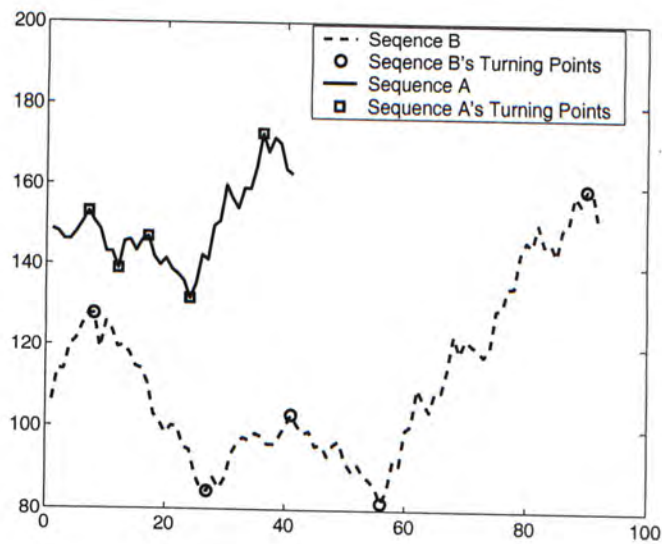


Figure 3.1: Similar subsequences of different lengths.

3.1.1 Control Points

As suggested in [Sch98], trendlines can be drawn using successions of relative highs (lows), which are respectively data points higher (lower) than those over N prior and N succeeding days. Usually, this kind of data points are referred as turning points. N is a user-defined parameter which has direct effect on the resulting trendline. Intuitively, a larger N results in a set of trendlines showing a more general view of the time series but those relative highs and lows used can represent more significant changes comparing with those derived from a smaller

N . The notion of control point is similar to relative highs and relative lows except that we do not classify highs and lows (or peaks and troughs) and there is no fixed N when extracting control points from the series. In general, a control point is a data point that indicates remarkable changes over a certain period of time. We only consider control points which are either local maximum or local minimum in a time series $A = \{a_1, a_2, a_3, \dots, a_n\}$.

Definition 1 (N -day-coverage control point) *A local maximum or local minimum, a_j , is an N -day-coverage control point if N is a maximum integer such that either $\{\forall m \in [-N, N], a_j \geq a_{j+m}\}$ or $\{\forall m \in [-N, N], a_j \leq a_{j+m}\}$ holds.*

As mentioned in Chapter 1, the coarseness of a shape extracted from a time series is inversely proportional to the number of points used in the representation. A coarser shape is formed by joining a smaller set of points which can highlight the most important changes. By progressively adding relatively less important points into the set, the shape for approximation can be refined. Therefore, all control points of a time series have to be first sorted according to their importance before organizing them in a lattice structure (to be described in Section 3.1.2). The goal is to rank the control points such that those which can represent more significant changes in the series are ranked higher.

The value of N for each control point depends on how many neighboring data points that are all larger or smaller than it without considering how much they differ from it. Using N as the quantitative measure of the importance of a control point is not informative enough, because we are interested in the pattern formed by a set of control points and the rate of change of data values is a crucial factor that affects the pattern formed. Consider the three control points (A, B and C) in the time series shown in Figure 3.2. Both points A and B are 10-day-coverage control points and C is a 5-day-coverage control point. However, point

B should be of the least importance among the three control points from the shape formation perspective. We therefore propose a measure of the significance of a point a_j , known as $Importance(a_j)$.

Definition 2 (Importance) Given an N -day-coverage control point, a_j , its importance among the series is defined by:

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

According to definition 2, points A, B and C have values of N equal to 10, 10 and 5 respectively but their $Importance$ are 100, 10 and 25.

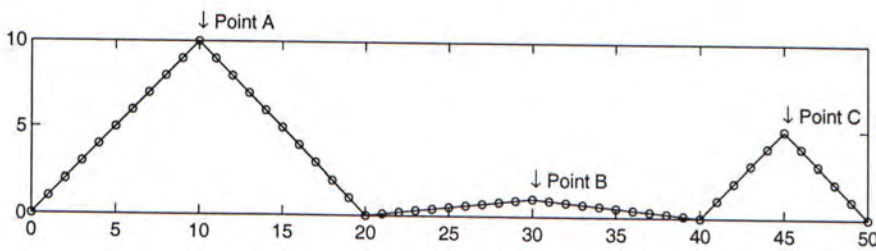


Figure 3.2: Three local maxima: points A, B and C.

3.1.2 Lattice Structure

After sorting the time series data according to their relative importance, the lattice structure can be constructed. The lattice structure consists of layers of control points. These layers are disjoint to each others, that is, a control point only appears in one of the layers of the lattice structure. As an example, Figure 3.3 shows a lattice structure formed by twelve control points, each labelled by its index value (time value) in a circle. Each layer of control points together with control points from all upper layers form a series that approximates the original

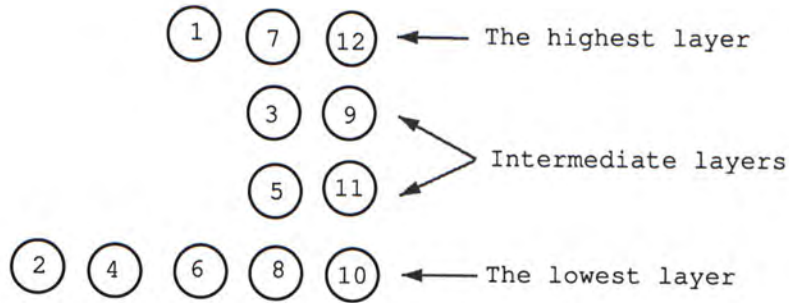


Figure 3.3: Example of a lattice structure of a series with twelve control points.

series. Figure 3.4 shows the actual set of control points used to form features from the lattice structure of Figure 3.3.

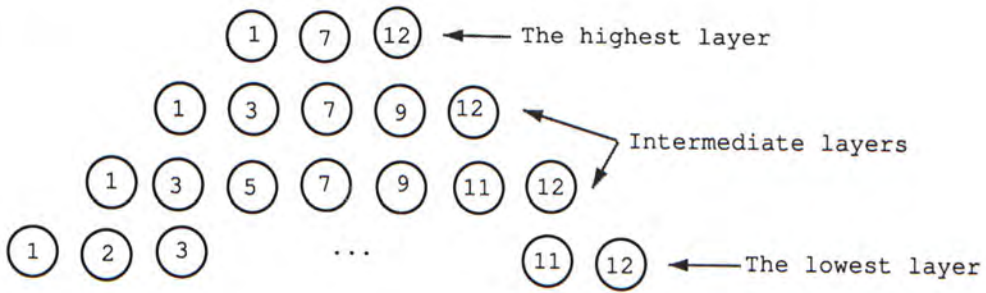


Figure 3.4: Control points used to form features from the lattice structure of Figure 3.3

Before presenting the algorithm on establishing the lattice structure of a given time series $A = \{(ax_1, ay_1), \dots, (ax_n, ay_n)\}^\dagger$, the formal definitions of the lattice structure, the segments generated and their properties are first given.

Definition 3 (Lattice Structure) *A lattice structure is defined as follows: It consists of N layers of control points. Each layer, layer $_i$ where $i \in \{1, \dots, N\}$,*

[†]This is a two-dimensional notation of a series. (ax_i, ay_i) represents the time-domain value (ax_i) and the amplitude value (ay_i) of a point a_i .

has a set of control points CP_i where $CP_i \subset A$. The highest and lowest layers are respectively $layer_1$ and $layer_N$.

The set of control points used for forming an approximation pattern at $layer_i$ is $CP_1 \cup \dots \cup CP_{i-1} \cup CP_i$. Having them sorted according to their time-domain values (ax_i), we use this sorted sequence of control points, denote it as $C_i = \{(cx_1, cy_1), \dots, (cx_m, cy_m), (cx_{m+1}, cy_{m+1})\}$, to form m consecutive segments which together form an approximation pattern.

Definition 4 (Segment) *The j -th segment of $layer_i$ is defined as the following line segment*

$$Seg_j(x) = x \left(\frac{cy_{j+1} - cy_j}{cx_{j+1} - cx_j} \right) + cy_j - cx_j \left(\frac{cy_{j+1} - cy_j}{cx_{j+1} - cx_j} \right) \quad x \in [cx_j, cx_{j+1}].$$

As the segment may not pass through all the data points, we have to compute the error distance between the segment and all data points aligned along the same time interval. We first define the error induced in each segment in Definition 5. Then, based on this segment error, the error of each layer is defined in Definition 6.

Definition 5 (Segment Error) *The segment error of any segment, Seg_j , is defined as*

$$SegErr_j = \sum |Seg_j(ax) - ay|$$

for all (ax, ay) satisfying $cx_j < ax < cx_{j+1}$ and $(ax, ay) \in A$.

Definition 6 (Layer Approximation Error) *The layer approximation error of any layer, $layer_i$, is defined as*

$$LayerErr_i = \max_{j=1}^m (SegErr_j)$$

A lattice structure has the following three properties:

Property 1 (Compactness) *The sets of control points at different layers are disjoint to each other, that is, for any $i \neq j$, $CP_i \cap CP_j = \emptyset$.*

Property 2 (Significant-Feature-First) *For any control point (cx_j, cy_j) in C_i , it must be the most important point among all control points located between its two adjacent control points which are (cx_{j-1}, cy_{j-1}) and (cx_{j+1}, cy_{j+1}) . Formally, the following should hold*

$$Importance((cx_j, cy_j)) > Importance((cx_l, cy_l)),$$

where $cx_{j-1} < cx_l < cx_{j+1}$.

Property 3 (Feature Refinement) *For any two different layers, $layer_i$ and $layer_j$ where $i, j \in \{1, \dots, N\}$, $LayerErr_i < LayerErr_j \Leftrightarrow i > j$.*

The approximation error is monotonically increasing from that of the highest layer ($LayerErr_1$) to that of the lowest layer ($LayerErr_N$). As mentioned in Chapter 1, sequence approximation can be subjective, users may control the coarseness of the patterns extracted by restricting the approximation error bound with an upper limit and lower limit, say $[\varepsilon_{up}, \varepsilon_{low}]$. In other words, the constructed lattice structure should have $LayerErr_1 \leq \varepsilon_{up}$ and $LayerErr_N \geq \varepsilon_{low}$.

3.1.3 Algorithm on Lattice Construction

The algorithm applies a top-down approach to construct the lattice structure from $layer_1$ to $layer_N$. Figure 3.5 shows the pseudo-code of the algorithm. For each iteration of loop L1, a set of control points will be chosen for the newest layer formed (new_layer) and inserted into the lattice structure ($Lattice[new_layer]$).

Each control point selected (*selected_CP*) has the largest value of *Importance* either among all control points (if no control point has been selected yet) or within a segment with the largest segment error among all the segments formed by the set of previously selected control points (*new_CP_set*). The largest segment error before (*pre_SegmentErr*) and after (*post_SegmentErr*) inserting the selected control point into the set (*new_CP_set*) will be compared. If the largest segment error is increased after the insertion, the selected control point failed to minimize the maximum segment error. This phenomenon is due to the trough-to-trough or peak-to-peak problem as shown in Figure 3.6. Further control points have to be selected in loop L2 for such problem and also if the maximum segment error exceeds the user specified bound (ϵ_{up}). Finally, the set of selected control points will only be included into the new layer if the layer error (or the maximum segment error) is not below the user specified lower bound (ϵ_{low}).

3.2 Pattern Matching

With a lattice structure holding different layers of patterns of a time series subjected to an approximation bound $[\epsilon_{up}, \epsilon_{low}]$, a query on existence of certain pattern can be answered without scanning all the raw data in the time series. The definition of similarity is based on linear scaling and shifting in both the time and the amplitude dimensions. In the following sections, a definition of similarity between a query pattern $Q = \{(qx_1, qy_1), \dots, (qx_{n-1}, qy_{n-1}), (qx_n, qy_n)\}$ and a sequence of control points $C = \{(cx_1, cy_1), \dots, (cx_{n-1}, cy_{n-1}), (cx_n, cy_n)\}$ and the error measurement of matching will be given. With control points holding different layers of features subjected to different error bounds, a query on existence of certain shapes under an error bound ϵ can be answered without checking every data in the series. We claim that the subsequence, bounded by the control points,

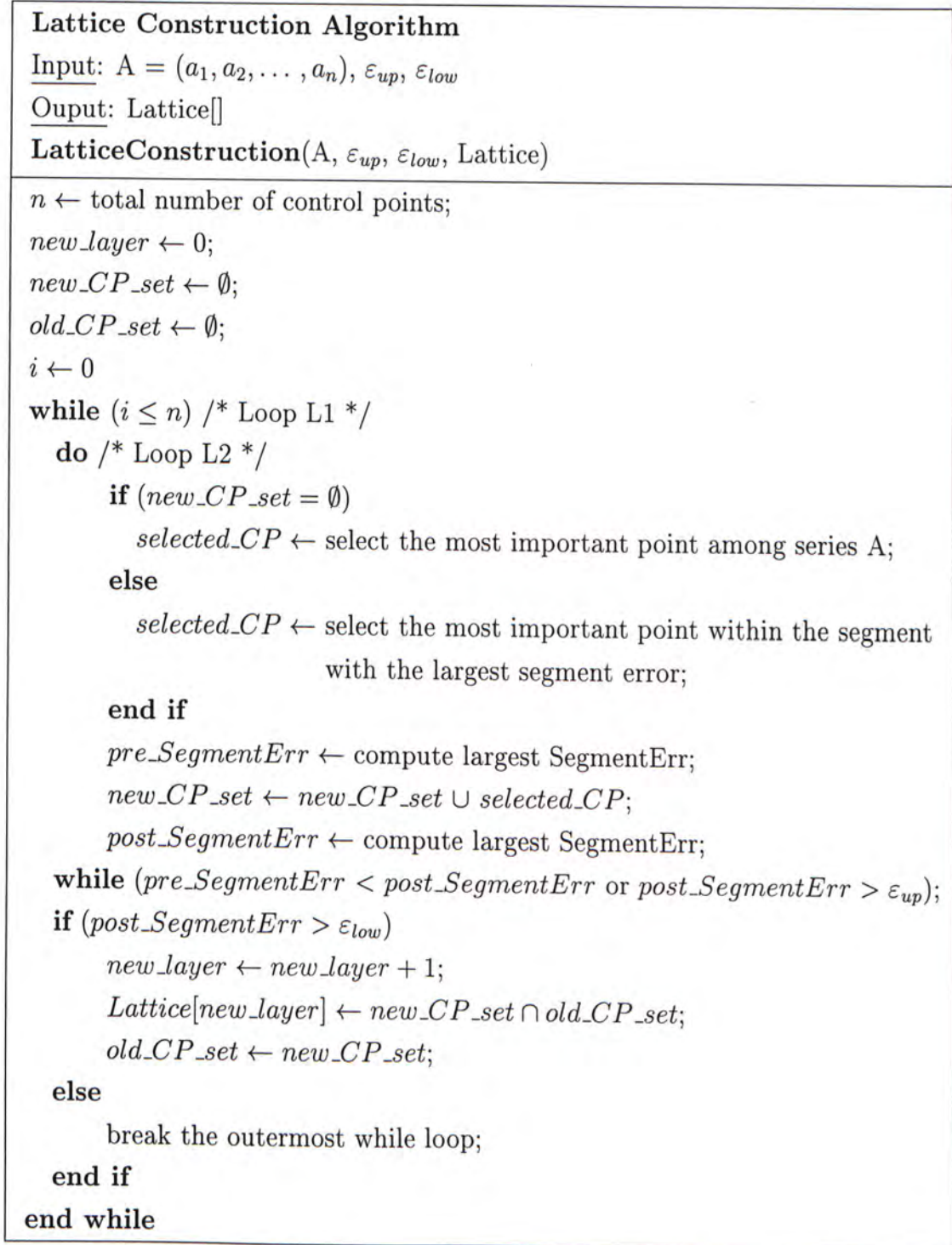


Figure 3.5: Lattice Structure Construction Algorithm.

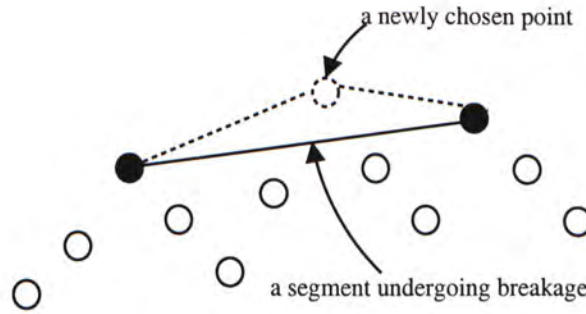


Figure 3.6: A peak-to-peak problem after a point is chosen for breaking up a segment.

forms a similar shape with the query if the shape of the query and those control points are similar with respect to a threshold.

The definition of similarity is based on linear scaling and shifting transformations in both the time and the amplitude dimensions. Figure 3.7 is used as an example to demonstrate the effects of scaling and shifting on sequence similarity. In this figure, four sequences are claimed to be similar. They are $A = \{(1, 5), (2, 10), (3, 6), (4, 12), (5, 4)\}$, $B = \{(1, 10), (2, 20), (3, 12), (4, 24), (5, 8)\}$, $C = \{(1, 25), (2, 30), (3, 26), (4, 32), (5, 24)\}$, $D = \{(1, 2.5), (1.5, 5), (2, 3), (2.5, 6), (3, 2)\}$. Although they are different time series, they are closely related. B can be obtained after scaling A along the amplitude axis by a factor of 2; C can be obtained from A by shifting along the amplitude axis up by 20 units. If B is scaled along the amplitude axis by a factor of 0.5 and then shifted up by 20 units, it becomes C . Moreover, if the interval between data points of A is scaled by a factor of 0.5, it becomes D . A , B , C and D are essentially the same when suitable scaling and shifting transformations are applied.

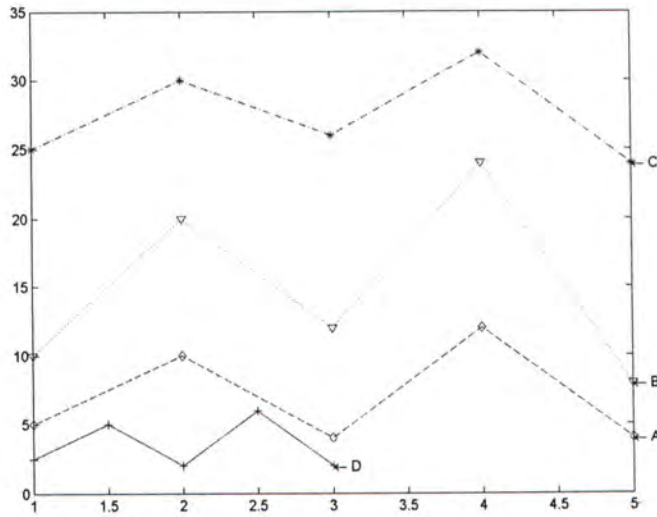


Figure 3.7: Examples of scaling and shifting transformation.

3.2.1 Formulating the Problem of Similarity

Consider each (qx_i, qy_i) and (cx_i, cy_i) where $i \in [1, n]$ as a 2D coordinates such that the query series Q and series of control points C can be visualized as polylines in a 2D space. Our problem is to *find an optimum way to scale and translate series Q such that the resultant shape best matches C or vice versa.*

Before solving the problem, clear definitions of *scaling*, *translation* and *best matches* are given. We use homogeneous coordinates (affine space) so that we can represent both 2D scaling and translation with a single 3×3 matrix. Therefore, each of the 2D coordinates, (qx_i, qy_i) , is now transformed into $(qx_i, qy_i, 1)$ (affine transformations). Next, we define s_x and s_y as the scaling on X and Y -dimensions respectively and also, t_x and t_y as the translation on X and Y -dimensions respectively. In this way, we can formulate the following matrix to describe a unique

2D scaling and translation,

$$\begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix}.$$

Hence, we are ready to apply this transformation on each point (qx_i, qy_i) in Q and obtain their transformed positions by

$$\begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} qx_i \\ qy_i \\ 1 \end{pmatrix}.$$

Furthermore, we can define the error concerning this scaling and translation on Q . Let E be the error such that

$$E = \sum_{i=1}^n \left\| \begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} qx_i \\ qy_i \\ 1 \end{pmatrix} - \begin{pmatrix} cx_i \\ cy_i \\ 1 \end{pmatrix} \right\|^2. \quad (3.1)$$

Consequently, our matching problem becomes a minimization problem such that we have to find s_x, s_y, t_x and t_y that minimize E . Furthermore, we can transform E into a simplified form by making use of the independence in dimensions,

$$\begin{aligned} E &= \sum_{i=1}^n \left\| \begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} qx_i \\ qy_i \\ 1 \end{pmatrix} - \begin{pmatrix} cx_i \\ cy_i \\ 1 \end{pmatrix} \right\|^2 \\ &= \sum_{i=1}^n \left[\left\| (qx_i s_x + t_x) - cx_i \right\|^2 + \left\| (qy_i s_y + t_y) - cy_i \right\|^2 \right] \\ &= \sum_{i=1}^n \left[\left\| (qx_i s_x + t_x) - cx_i \right\|^2 \right] + \sum_{i=1}^n \left[\left\| (qy_i s_y + t_y) - cy_i \right\|^2 \right] \\ &= E_x + E_y. \end{aligned}$$

With this derivation, our matching problem can further be simplified such that we can minimize E_x with respect to s_x and t_x and E_y with respect to s_y and

t_y independently. Therefore, to minimize E , we can minimize E_x and E_y . Since E_x and E_y are structurally the same, the following derivation will be based on E_x . Recall that

$$E_x = \sum_{i=1}^n \left[\left\| (qx_i s_x + t_x) - cx_i \right\|^2 \right].$$

To minimize E_x , we can first take partial derivative on E_x with respect to s_x and t_x ,

$$\begin{aligned} \frac{\partial E_x}{\partial s_x} &= 2 \sum_{i=1}^n [(qx_i s_x + t_x) - cx_i] qx_i \\ \frac{\partial E_x}{\partial t_x} &= 2 \sum_{i=1}^n [(qx_i s_x + t_x) - cx_i]. \end{aligned}$$

Putting $\frac{\partial E_x}{\partial s_x} = \frac{\partial E_x}{\partial t_x} = 0$ into these two equations yields

$$\begin{aligned} \sum_{i=1}^n [(qx_i s_x + t_x) - cx_i] qx_i &= 0 \\ \sum_{i=1}^n [(qx_i s_x + t_x) - cx_i] &= 0 \end{aligned}$$

and re-arranging s_x and t_x as the subjects, we have

$$\begin{aligned} \left(\sum_{i=1}^n qx_i^2 \right) s_x + \left(\sum_{i=1}^n qx_i \right) t_x &= \sum_{i=1}^n (cx_i qx_i) \\ \left(\sum_{i=1}^n qx_i \right) s_x + nt_x &= \sum_{i=1}^n (cx_i). \end{aligned}$$

Now, we define

$$\begin{cases} P_2 = \sum_{i=1}^n qx_i^2 \\ P_1 = \sum_{i=1}^n qx_i \\ Q_2 = \sum_{i=1}^n (cx_i qx_i) \\ Q_1 = \sum_{i=1}^n (cx_i) \end{cases}$$

and obtain

$$\begin{pmatrix} P_2 & P_1 \\ P_1 & n \end{pmatrix} \begin{pmatrix} s_x \\ t_x \end{pmatrix} = \begin{pmatrix} Q_2 \\ Q_1 \end{pmatrix}.$$

Solving the equation gives

$$s_x = \frac{P_1 Q_1 - n Q_2}{P_1^2 - n P_2}$$

and

$$t_x = \frac{P_1 Q_2 - P_2 Q_1}{P_1^2 - n P_2}.$$

Similarly, we can apply the same technique to find s_y and t_y to minimize E_y . More precisely, we denote this set of values as s_{x0} , t_{x0} , s_{y0} and t_{y0} so as not to confuse with s_x , t_x , s_y and t_y . Since they minimize E_x and E_y , E can also be minimized as E_x and E_y are independent. As a conclusion, s_{x0} , t_{x0} , s_{y0} and t_{y0} will guarantee to minimize the square fitting error so that the query Q can best match with control points C .

3.2.2 Error Measurement

Up to now, we can find s_{x0} , t_{x0} , s_{y0} and t_{y0} for the best matching. However, we still need to measure how good Q matches C under this set of values. Regarding this, we substitute s_{x0} , t_{x0} , s_{y0} and t_{y0} back to Equation 3.1 so that we can find the minimum error, say E_{min} .

Intuitively, from the definition of E , E is the sum of Euclidean distance between each corresponding points in C and the transformed Q . Here, we have to note that if the sequence data in Q or C are large numerically, the resultant E_{min} would be large even if the two series just deviate from each other slightly. To isolate the sizing effect of Q and C , we first look at the formulation of E_{min} ,

$$E_{min} = E_{minx} + E_{miny}.$$

Note that E_{min} has two components E_{minx} and E_{miny} that corresponding to the minimum error terms in X and Y -dimensions. By taking into account the range of data in both time and amplitude domains, we propose another error definition defined in Definition 7.

Definition 7 (Relative Minimum Error) *The relative minimum error between the query sequence and a control sequence is defined as*

$$E_{min\phi} = \frac{E_{minx}}{r_{Qx}r_{Cx}} + \frac{E_{miny}}{r_{Qy}r_{Cy}}. \quad (3.2)$$

where r_{Qx} is the range of Q in X -dimension, that is $\max(qx_i) - \min(qx_i)$. Similarly, r_{Qy} , r_{Cx} and r_{Cy} are the range of Q in Y , range of C in X and range of C in Y respectively.

Therefore, to decide the matching error, we consult $E_{min\phi}$ instead of E_{min} .

3.3 Indexing Scheme

With the lattice structure, different shapes are captured at different layers. Given a query on a certain shape, sequential search for similar shapes is possible by traversing every control point at each layer. However, if a large set of time series (for example, different kinds of stocks) is needed to be processed and in turn forming large lattice structures, it would be wise if an indexing scheme can be applied to improve the searching performance. To achieve this, the indexing scheme proposed by Chu and Wong [CW99] can be modified and integrated with the lattice structure. The integrated indexing scheme significantly reduces the number of retrieved control points. However, there can be false alarms. Hence, post-processing on every set of control points to reject unqualified sets is needed.

3.3.1 Indexing with scaling and shifting proposed by Chu and Wong

In the work of [CW99], an indexing scheme which can efficiently search for any subsequences that are similar to a query after scale-shift transformation was proposed. The transformation is one dimensional along the amplitude domain.

To better illustrate the integration of this indexing scheme and the lattice structure, we rewrite the transformation with our notation: Denote a subsequence $C = \{(cx_1, cy_1), (cx_2, cy_2), \dots, (cx_n, cy_n)\}$ and a query sequence $Q = \{(qx_1, qy_1), (qx_2, qy_2), \dots, (qx_n, qy_n)\}$. The Euclidean distance between C and Q after scaling and shifting Q by a factor of s_y and an offset of t_y respectively is defined as

$$D(Q, C, s_y, t_y) = \sqrt{\sum_{i=1}^n ((qy_i s_y + t_y) - cy_i)^2}. \quad (3.3)$$

C is similar to Q if there exists a scaling factor s_y and a shifting offset s_t such that $D(Q, C, s_y, t_y) \leq \epsilon$.

The mechanism: Each subsequence is transformed by the SE-Transformation[†] and the resulting subsequence is inserted into an R -tree, in which each of its non-leaf node is bounded by a minimum bounding hyper-rectangle (MBR) defined by two end points, $L = (l_1, l_2, \dots, l_n)$ and $H = (h_1, h_2, \dots, h_n)$, $l_i < h_i$ for $1 \leq i \leq n$. A query Q is also transformed by the SE-Transformation and let the transformed query be $T_{se}(Q)$. Searching begins from the root node of the R -tree built by transformed subsequences. At each level, every node's MBR is enlarged by ϵ and if the line $tT_{se}(Q)$ penetrates this enlarged MBR ($L' = (l_1 - \epsilon, l_2 - \epsilon, \dots, l_n - \epsilon)$

[†]In \mathfrak{R}^n , a point \vec{p} (or a sequence with length n) is projected by the Shift-Eliminated (SE) Transformation $T_{se}(\vec{p})$ to the plane which passes through the origin $\vec{0}$ and has its normal vector in the direction of $\vec{N} = (1, 1, \dots, 1)$ in \mathfrak{R}^n .

and $H' = (h_1 + \epsilon, h_2 + \epsilon, \dots, h_n + \epsilon)$, then the node is traversed. When a leaf node is reached, the original sequence is retrieved if the minimum point line distance between the corresponding transformed sequence and the line $tT_{se}(Q)$ is within ϵ .

3.3.2 Integrating with lattice structure

The above indexing scheme cannot be directly applied to our lattice structure as the error definition (Equation 3.3) is different. The error definition in the indexing scheme, $D(Q, C, s_y, t_y)$, assumes the query sequence Q and any subsequence C have the same data interval. In contrast, the control points at each layer in the lattice structure are not necessarily equally spaced along the time axis. Our error definition, $E_{min\phi}$ (Equation 3.2), has thus taken time interval into account and allows scaling and shifting in both time and amplitude domains.

To integrate our definition of $E_{min\phi}$ with the R -tree indexing scheme, we can first temporarily set aside values from one dimension, assuming the query and all subsequences have equal values in the disregarded dimension and index the other dimension. Intuitively, amplitude domain should first be indexed while time domain should be set aside since time intervals within a subsequence usually have small deviations, but amplitude domain contributes to different features that people concerns about (for example, chart patterns).

Using standard deviation as a measure, we have carried out an experiment to compare the dispersion of subsequence's data intervals and data values. 500 lattice structures built from S&P500 historical stock data with approximation error bound $[\epsilon_{up} = 250, \epsilon_{low} = 50]$ have been analyzed. For each sliding window with size ranging from 3 to 12, we extracted subsequences from each layer of every lattice structure. The standard deviations of in data intervals and data

values of each subsequence are computed. Both the average standard deviations on amplitude and time domains of all the extracted subsequences against different sizes of sliding windows are plotted in Figure 3.3.2. This experiment showed that data values are more disperse than data intervals regardless of the size of the sliding window. As the size of sliding window increases, data values are shown to be much more disperse than data intervals. In searching for similar patterns of interest, considering time domain in advance may therefore result in higher rate of false alarms. This may deteriorate the searching performance as variations in data intervals are in general smaller compared with data values.

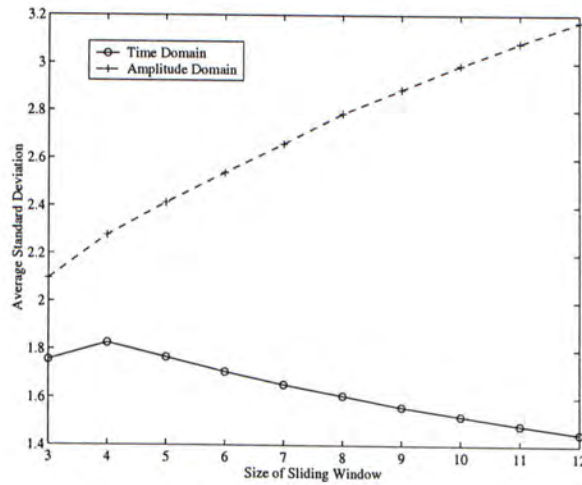


Figure 3.8: Average standard deviations of subsequences' time-domain and amplitude-domain values against the size of subsequences (size of sliding window).

Now, at the preliminary searching stage, we assume subsequences and the query sequence have the same data interval between corresponding pair of data. Referring to our error definition in the time domain, we have E_{minx} equals to zero. By rearranging equation 3.2, we have

$$E_{miny} = (E_{min\phi} - \frac{E_{minx}}{r_{Qx}r_{Cx}})r_{Qy}r_{Cy}.$$

Given the searching criteria $E_{min\phi}$ of a query $Q(qx_i, qy_i)$ where $i \in [1, n]$, any qualifying patterns $C(cx_i, cy_i)$ must satisfy

$$\begin{aligned} E_{miny} &\leq \left(E_{min\phi} - \frac{E_{minx}}{r_{Qx}r_{Cx}} \right) r_{Qy}r_{Cy} \\ &\leq E_{min\phi} r_{Qy}r_{Cy}. \end{aligned}$$

Thus, if E_{minx} is neglected, the error bound ($E_{min\phi} r_{Qy}r_{Cy}$) for searching similar patterns in amplitude domain will be larger, ensuring that no qualified control point sequence is missed. Note that E_{miny} has one variant (r_{Qy}) which depends on the range of data values. Therefore, searching the SE transformed subsequence in an R -tree would not have a fixed ϵ for MBR enlargement. Instead, at each tree level, every node's bounding box is enlarged from (L, H) to (L'', H'')

$$\begin{aligned} L'' &= (l_1 - \sqrt{E_{min\phi} r_{Qy} r_{Cy}}, l_2 - \sqrt{E_{min\phi} r_{Qy} r_{Cy}}, \dots, l_n - \sqrt{E_{min\phi} r_{Qy} r_{Cy}}) \\ H'' &= (h_1 + \sqrt{E_{min\phi} r_{Qy} r_{Cy}}, h_2 + \sqrt{E_{min\phi} r_{Qy} r_{Cy}}, \dots, h_n + \sqrt{E_{min\phi} r_{Qy} r_{Cy}}) \end{aligned}$$

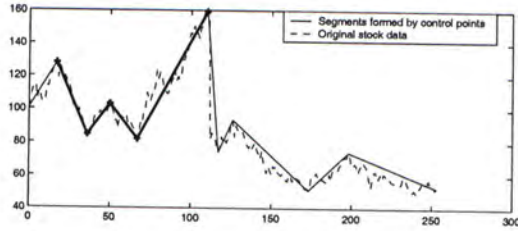
where $r_{Cy} = \max(h_i) - \min(l_i)$ and $r_{Qy} = \max(qy_i) - \min(qy_i)$. When a leaf node is reached, the original control points of those transformed subsequences should be further checked against the query. Those control points with relative minimum error $E_{min\phi}$ larger than the user specified error bound should be rejected. More criteria can also be imposed on scaling factors and shifting offsets in both time and amplitude domains.

3.4 Results

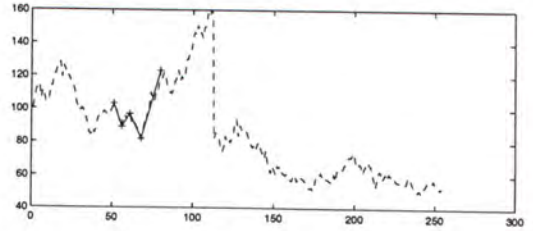
Experiments were carried out using historical data (from 6/16/1999 to 6/15/2000) of S&P 500 stocks. Every stock (open price only) is first pre-processed into a

lattice structure with approximation error bound $[\varepsilon_{up} = 250, \varepsilon_{low} = 50]$. Figure 3.9(a) shows the second layer of lattice structure of American Online and a “W” shape (highlighted by a thicker line) was formed by five consecutive control points starting from the second leftmost one. For instance, a query on this shape is carried out to all other stocks using sequential search. The query results are restricted by the two scaling factors ($s_x > 0.3, s_y > 0.3$) and the relative minimum error ($E_{min\phi} < 0.037$). Seven sequences of control points were reported to have satisfy the query constraints and they are shown in Figure 3.9 (b) to (h).

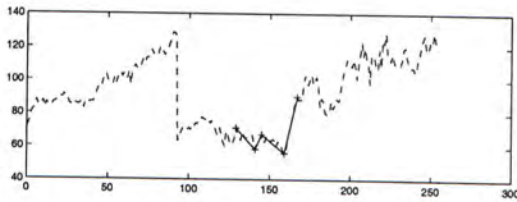
Using the same set of lattice structures, an experiment on searching the same query pattern was performed using our proposed indexing scheme and a comparison on searching performance was made. The dimension of the R -tree is equal to the five which is the same as of query sequence. A window of length five is placed and slid over each layer of every lattice structure. As the control points of any upper layer in lattice structure is the subset of its lower layer, there may be duplicates of the five extracted control points while the window slid over layer by layer. To ignore any duplicated set of control points, another R -tree was built to index time domain values (cx_1, cx_2, \dots, cx_5) of the set of control points whose data values (cy_1, cy_2, \dots, cy_5) had already been SE-transformed and indexed into the R -tree. Thus, each set of control points are first checked against the time domain indexing structure to avoid unnecessary duplications. With this indexing scheme, the query pattern in Figure (3.9a) requires 2.1 seconds user time (not including preprocessing time) whilst sequential search requires 42 seconds searching time. To further compare the performance between the sequential search and the indexing scheme, 10 queries were performed for different error bounds and the average user time used were collected. Their performances are plotted in Figure 3.10.



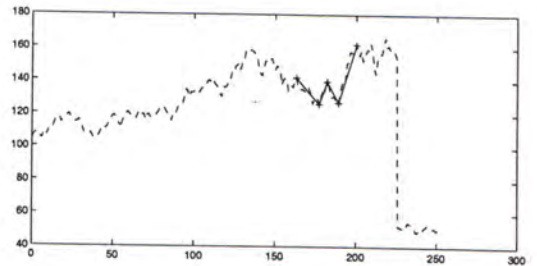
(a) The query pattern.



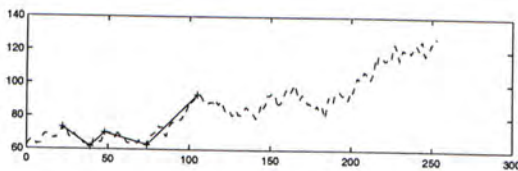
(b) American Online. $E_{min\phi} = 0.032$



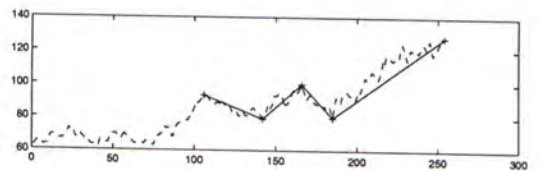
(c) Adobe Systems. $E_{min\phi} = 0.031$.



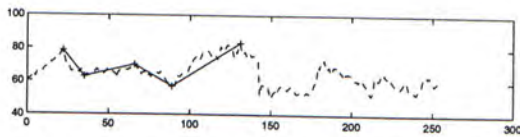
(d) General Electric. $E_{min\phi} = 0.036$.



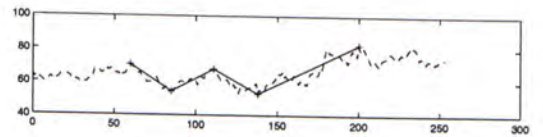
(e) Warner-Lambert. $E_{min\phi} = 0.025$.



(f) Warner-Lambert. $E_{min\phi} = 0.036$



(g) Lucent Technologies. $E_{min\phi} = 0.031$.



(h) Schlumberger Ltd. $E_{min\phi} = 0.021$.

Figure 3.9: The “W” shape found in layer 2 of American Online’s Lattice structure(as shown in chart (a)) was used as a query pattern. Query results are marked by crosses and are highlighted and are shown in charts (b) to (h).

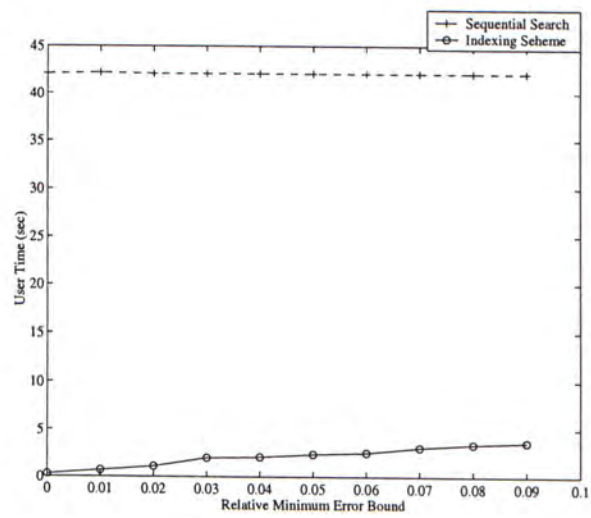


Figure 3.10: User time versus Relative Minimum Error of sequential search and our indexing scheme

Chapter 4

Chart Patterns Searching for Chart Analysis

In this chapter, a novel approach of finding meaningful chart patterns using critical points is proposed. Identifying prototype patterns has been a major issue in chart analysis. The identified chart patterns are useful in forecasting future trend. While one-day patterns (like gaps and spikes) can be easily identified, reversal patterns (like wedges) and continuation patterns (like triangles) can not be trivially retrieved .

4.1 Chart Patterns Overview

The most popular charting method is the bar chart (or the candlestick chart). A bar chart is plotted using the highest, the lowest and closing stock price. As an example, Figure 4.1 shows a bar chart of the stock price of Sun Microsystems adapted from [sto]. The top and bottom ends of a vertical line represents the daily high and daily low respectively whilst the daily closing price is represented

by a horizontal line. In this example, the chart is plotted from daily data but it can also be plotted from weekly data.



Figure 4.1: An example of Bar Chart using the stock price of Sun Microsystems.

Chart analysis is the process of recognizing and interpreting of individual patterns from a bar chart. Technically, these patterns are categorized into three main types, namely one-day patterns, continuation patterns and reversal patterns [EM97]. One-day patterns are found easily by examining each vertical line with its adjacent vertical lines. Gaps, spikes and reversal days are examples of one-day patterns. A gap day is one with its low being above the previous day's high or its high being below the previous day's low; a spike is a day whose high is sharply above the high of the preceding and succeeding days; a reversal day is a day that witnesses a new high in an upmove followed by reversing to a close below the preceding day's close. In contrast to one-day patterns, continuation patterns and reversal patterns are formed within long-term trends. Both patterns need careful examinations of trends and sharp changes over a period of time.

We propose a novel approach that automatically identifies reversal and continuation patterns using critical turning points. These two categories have several major types of patterns that are commonly used by stock analyst and the expla-

nation of these chart patterns in Sections 4.1.1 and 4.1.2 are adapted from [sto]. Figures 4.2 and 4.3 show different bar chart patterns which are of reversal type and continuous type respectively. The close price indicator is omitted as it does not contribute to the chart pattern formation. Again, the top and bottom of the vertical lines represent day highs and day lows. A detailed description of the formation of these patterns is described in the following subsections.

4.1.1 Reversal Patterns

In general, a reversal pattern implies that the previous trend will be reversed when the pattern is complete. Here is a list of such patterns that can be identified by our method.

Falling Wedge (Figure 4.2 (a)): It is a pattern that begins wide at the top and contracts as the price drops. The overall price pattern forms a cone that slopes down as the highs and lows converge. There should be at least two highs to form the upper trendline and two lows to form the lower trendline. Each high (or low) should be lower than the previous high (or low).

Rising Wedge (Figure 4.2 (b)): It is a pattern that begins wide at the bottom and contracts as the price rises and the trading range narrows. There should be at least two highs to form the upper trendline and two lows to form the lower trendline. Each high (or low) should be higher than the previous high (or low).

Head and Shoulders Top (Figure 4.2 (c)): It contains three consecutive peaks with the middle peak (head) being the highest and two roughly equal and lower adjacent peaks (shoulders). There are two lows which respectively mark the end of the left shoulder and beginning of the right shoulder. The line joining these two lows is called a neckline.

Head and Shoulders Bottom (Figure 4.2 (d)): It can be referred to as the inverse of head and shoulders top. It forms after a downtrend, and its completion

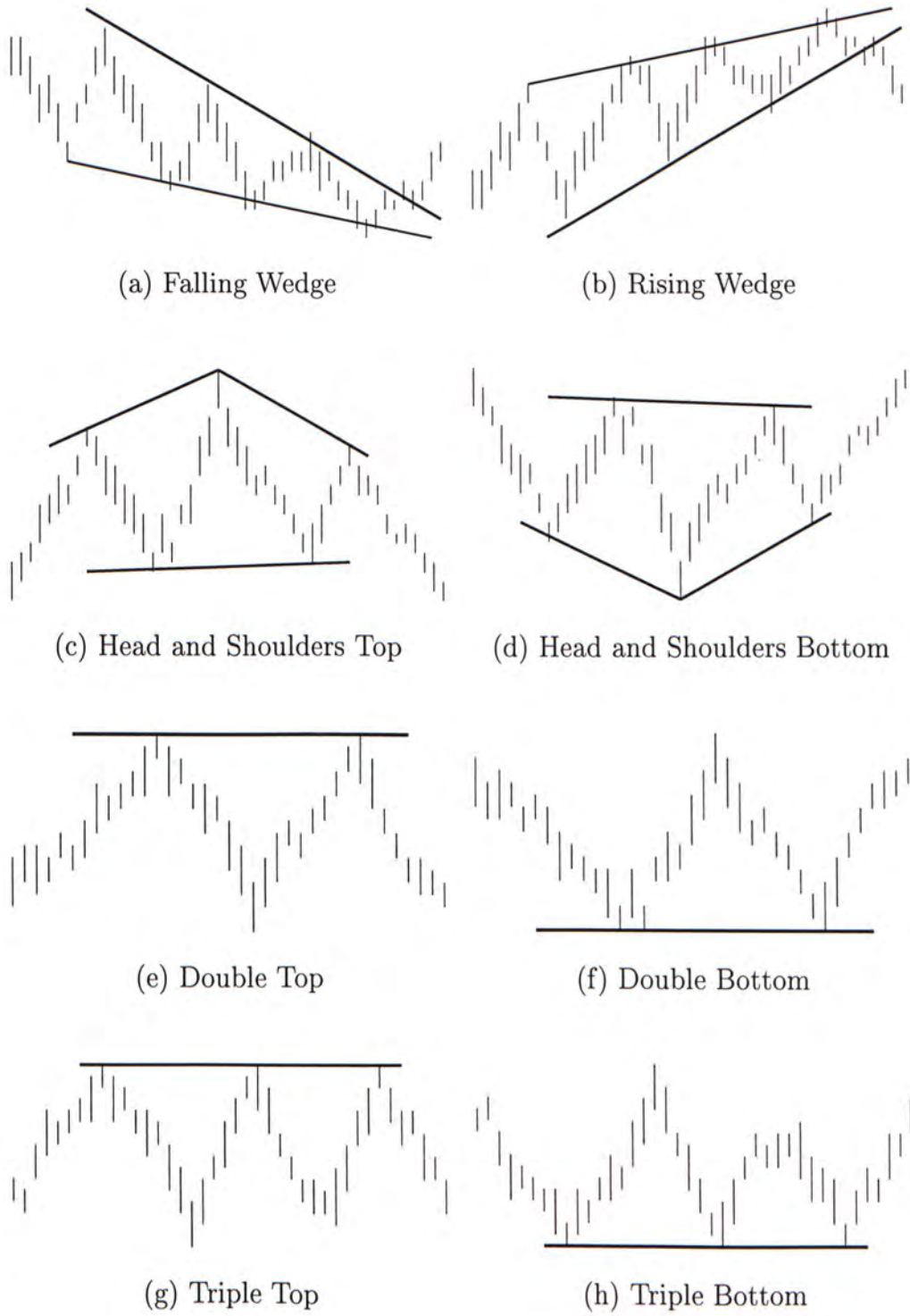


Figure 4.2: Reversal Patterns.

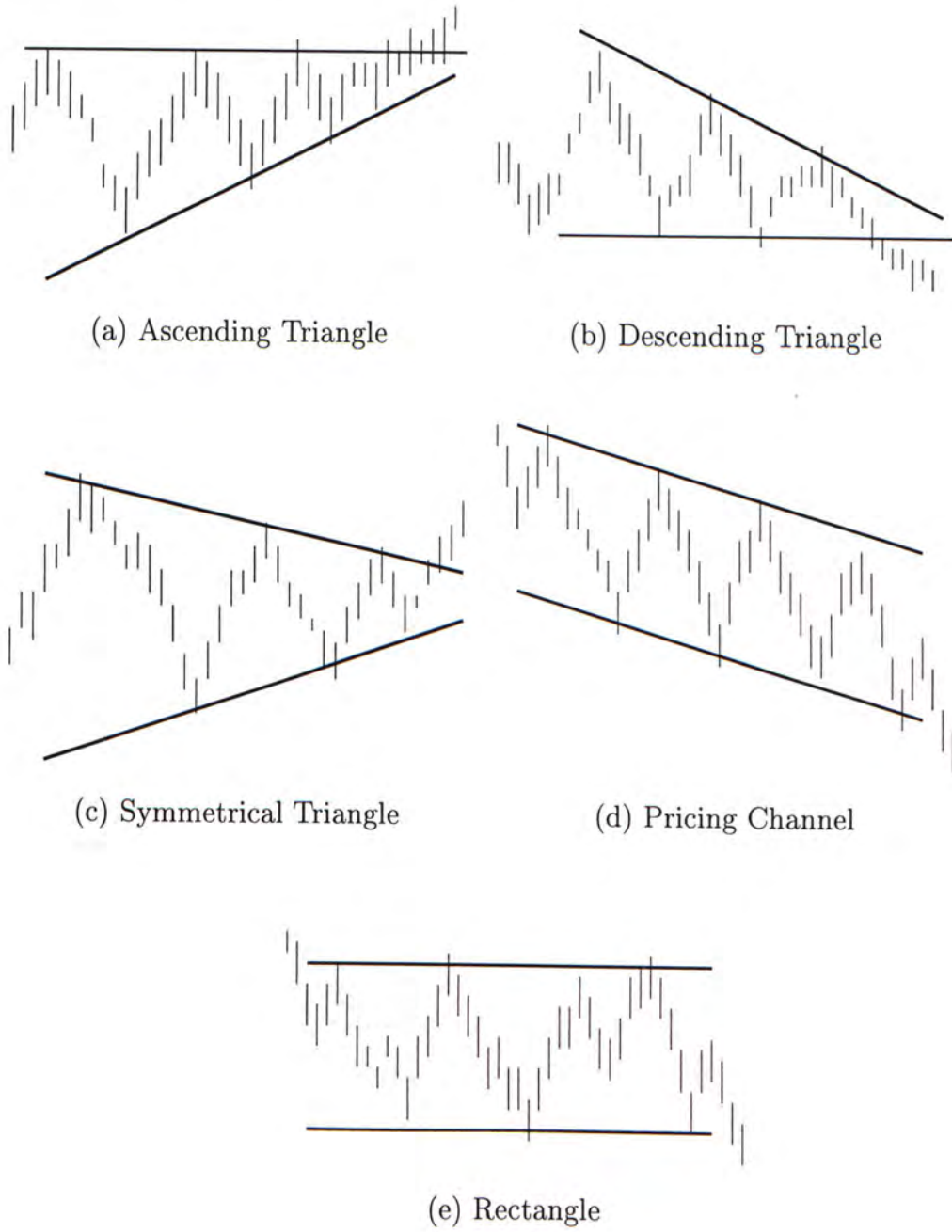


Figure 4.3: Continuation Patterns.

marks reverse change in trend. It encompasses three consecutive troughs (lows) with the middle trough (head) being the deepest and the two adjacent troughs (shoulder) being shallower. There are two highs which respectively mark the end of the left shoulder and beginning of the right shoulder. The line joining these two highs is called the neckline. The neckline should be roughly horizontal.

Double Top (Figure 4.2 (e)): It forms after an extended downtrend. It is made up of two successive peaks (highs) which should be roughly equal and with a moderate trough (low) in between.

Double Bottom (Figure 4.2 (f)): It forms after an extended uptrend. It is made up of two successive troughs (lows) which should be roughly equal and with a moderate peak (high) in between.

Triple Top (Figure 4.2 (g)): It is a pattern formed by three highs which should be reasonably equal, well spaced and mark significant turning points. They do not have to be exactly equal but at least equivalent to each other.

Triple Bottom (Figure 4.2 (h)): It is a pattern formed by three lows which should be reasonably equal, well spaced and mark significant turning points. They do not have to be exactly equal but at least equivalent to each other.

4.1.2 Continuation Patterns

A continuation pattern implies that the previous trend will resume when the pattern is complete. Here is a list of such patterns that can be identified by our method.

Ascending Triangle (Figure 4.3 (a)): It can be also referred to as a right-angle triangle. Two or more highs, with reasonable proximity of each other, form a horizontal trendline at the top. Two or more rising lows form an ascending trendline that converges on the horizontal trendline. Each of these rising lows should be in between two highs of the top horizontal trendline.

Descending Triangle (Figure 4.3 (b)): It can also be referred to as a right-angle triangle. Two or more lows, with reasonable proximity of each other, form a horizontal line at the bottom. Two or more declining peaks form a descending trendline above that converges with the horizontal line as it descends. Each of these declining highs should be in between two lows of the top horizontal trendline.

Symmetrical Triangle (Figure 4.3 (c)): It can also be referred to as a coil which is wide at the beginning and narrowing over time. At least two highs and two lows are required such that two trendlines form a symmetrical triangle. The second high should be lower than the first and the upper line should therefore be sloping down. The second low should be higher than the first and the lower line should therefore be sloping up.

Price Channel (Figure 4.3 (d)): It is a pattern that may be sloping up or down and is bounded by an upper and lower trendlines which are parallel to each other.

Rectangle (Figure 4.3 (e)): This pattern is easily identifiable by two comparable highs and two comparable lows. The highs and lows can be connected to form two parallel lines that make up the top and bottom of a rectangle.

4.2 Representation

As in finding patterns in a time series, a good data representation is also essential in finding chart patterns. All patterns described in Section 4.1 are formed by an upper trendline and (or) a lower trendline which are roughly constructed by joining a set of highs and lows. Therefore, trendlines are crucial representations for mining chart patterns, particularly when processing a massive set of highs and lows. Finding the appropriate set of highs (lows) for the upper (lower) trendlines is one of the major challenges in finding patterns in charts. As will be shown in

this section, traditional method of trendline formation is not suitable and flexible enough for identifying chart patterns. Therefore, we propose a new method in constructing the upper and lower trendlines. After forming the upper and lower trendlines individually, we have to view the chart as a whole and prepare for searching. This is done by forming a set of trendline pairs, each of which composed of a portion of upper trendline and lower trendline. These trendline pairs can be considered as basic units for later chart pattern classifications.

4.2.1 Trendline Preparation

One standard definition of an uptrend(downtrend) is a succession of higher (lower) highs and higher (lower) lows [Sch98]. Thomas DeMark claimed that the drawing of trendlines is a highly arbitrary process [Dem94]. Presented with the same chart, different people will intuitively draw different trendlines, and even the same person might draw different trendline at different times. DeMark's methodology for defining trendlines is explained by the following definitions:

Definition 8 (Relative High) *A relative high refers to a day with its daily high higher than that of the N prior and N succeeding days, where N is a user-defined parameter.*

Definition 9 (Relative Low) *A relative low refers to a day with its daily low lower than that of the N prior and N succeeding days, where N is a user-defined parameter.*

Definition 10 (Downtrend Line) *A downtrend line is defined as the line connecting the most recent relative high and the most recent preceding relative high that is also higher than the most recent relative high.*

Definition 11 (Uptrend Line) *The uptrend line is defined as the line connecting the most recent relative low and the most recent preceding relative low that is also lower than the most recent relative low.*

Successive downtrend lines and uptrend lines can be drawn based on Definitions 8 to 11. Depends on user's preference, the drawing of downtrend (uptrend) line is not restricted by using relative highs (lows) only but relative lows can also be used. Figure 4.4 shows successive downtrend lines and uptrend lines drawn using $N = 5$, $N = 9$ and $N = 13$ respectively. As can be seen from the figures, different values for N yield very different trend lines. The lower the value of N , the more frequently the downtrend lines and uptrend lines are redefined. Schwager stated that the choice of N is strictly a matter of subjective preference [Sch98]. Generally speaking, short-term traders gravitate to low values of N and long term traders prefer to use high values.

Although relative highs and lows constructed by the parameter N can mark long term or short term changes, their spacing or distribution over a period of time are not considered. To become a qualified chart patterns, however, those relative highs and relative lows need to be properly spaced and able to mark significant turning points. In order to meet such requirements, we propose some modifications to the selection of relative highs, relative lows, and the subsequent trendline drawing process.

To mark out significant turning points, a quantitative measurement that facilitates the comparison of importance of a set of highs or lows over a period of time is essential. Solely using the parameter N cannot give comparison among all relative highs or relative lows which are selected by this single parameter. Moreover, a relative high or a relative low can be chosen more than once with different values of N . Although it is possible to mark out significant changes by

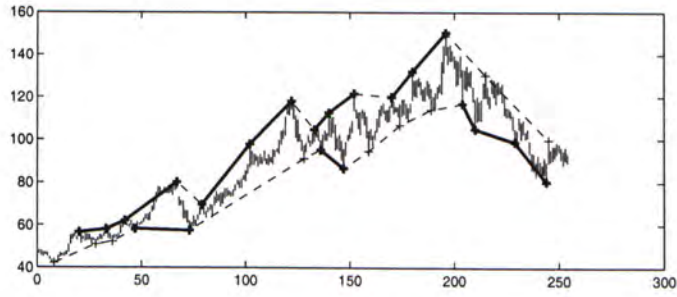
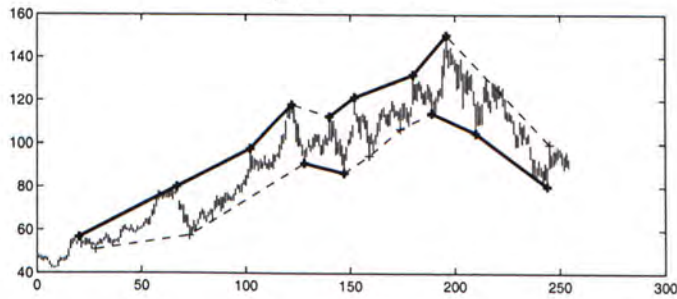
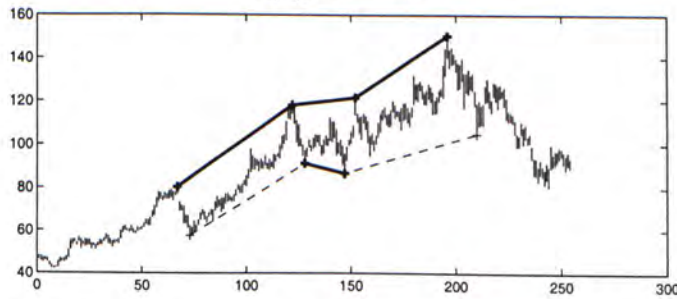
(a) $N = 5$.(b) $N = 9$.(c) $N = 13$

Figure 4.4: Succession of downtrend and uptrend lines

using a larger value of parameter N , the spacing is unpredictable. Instead of fixing this parameter, we inspect the extent that each relative high or relative low can cover. To make ranking of importance feasible, we thus define the term $\text{max-}N$ for relative high and relative low as follows:

Definition 12 (max- N Relative High) *A daily high is higher than the high on at most N prior and N succeeding days.*

Definition 13 (max- N Relative Low) *A daily low is lower than the low on at most N prior and N succeeding days.*

Therefore, different relative highs (lows) will have different $\text{max-}N$ values and each of them should have a unique $\text{max-}N$ value. For example, if a relative high which is at most 5 days higher than its preceding and succeeding high, then it is a $\text{max-}5$ relative high. Therefore, $\text{max-}N$ can be used for comparing the significance among relative highs or relative lows over a period of time. Figure 4.5 shows the pseudocode for computing the $\text{max-}N$ of each high and low in a given series.

Another issue in qualifying a chart pattern is the way that highs and lows used in making trendlines are spaced. Again, it is subjective but one can specify the ideal spacing using a range which simply denotes the minimum distance (we denote it as *min_dist*) and maximum distance (we denote it as *max_dist*) between adjacent highs or lows. Our definition of trendlines ensures good spacing of highs or lows, and at the same time marks significant turning points. The semantics of downtrend and uptrend lines which are previously stated in Definitions 10 and 11 are preserved. However, we need to enforce the drawing of trendlines by using $\text{Max-}N$ relative high and $\text{Max-}N$ relative low instead of a fixed parameter N . Given the two user-defined parameters (*min_dist* and *max_dist*), the following definitions show how upper trendlines and lower trendlines are drawn using $\text{Max-}N$ relative high and $\text{Max-}N$ relative low respectively.

Rating Relative PointInput: price[1 → num_days],typeOutput: max_N[1 → num_days]

```
for index = 1 to num_days {
  days = 0;
  if index ≤ num_days - index
    coverage = index - 1;
  else
    coverage = num_days - index;
  switch{type} {
    case 'high' :
      for j = 1 to coverage {
        if price[ index ] ≤ price[ index - j ] or price[ index ] ≤ price[ index + j ]
          break for loop;
        days = j
      }
    case 'low' :
      for j = 1 to coverage {
        if price[ index ] ≥ price[ index - j ] or price[ index ] ≥ price[ index + j ]
          break for loop;
        days = j;
      }
  }
  max_N[index] = days;
}
```

Figure 4.5: Rating Relative Point Algorithm.

Definition 14 (Upper Trendline) *The upper trendline is defined as the line joining the formerly chosen relative high and a succeeding Max-N relative high which is at least min_dist and at most max_dist away from the former relative high, and there is no $max-N'$ relative high such that $N' > N$.*

Definition 15 (Lower Trendline) *The lower trendline is defined as the line joining the formerly chosen relative low and a succeeding Max-N relative low which is at least min_dist and at most max_dist apart from the former relative low, and there is no $max-N'$ relative low such that $N' > N$.*

Based on Definitions 14 and 15 for finding succeeding relative highs and relative lows, trendlines will be continually generated as new relative highs and relative lows are chosen while traversing along the time line. Figure 4.6 shows the pseudocode for generating successions of trendlines using either highs or lows. As an example, with the stock prices in Figure 4.4, the upper trendlines and lower trendlines which are defined using Definitions 14 and 15 are shown in Figure 4.7.

4.2.2 Trendline Pair

The upper and lower trendlines together enclose all highs and lows forming a form a price envelop. It is possible that some portions of this price envelop can form valid chart patterns as described in Section 4.1. However, it is not practical to check every portion of the price envelop against all kinds of chart patterns and find their maximum likelihoods. In addition, the number of relative highs and relative lows which are involved in forming a chart pattern is not fixed.

Instead of searching a complete chart pattern in an atomic manner, we propose to decompose the price envelop into small pieces called trendline pairs which can be considered as basic units for later pattern recognition. A trendline pair

Trendline Establishing Algorithm

Input: min_dist,max_dist,max_N[1→num_days]

Output: trend[1→num_trendlines]

```
index = 0;
while index < num_days - max_dist {
    index = index + 1;
    best_so_far = -1;
    idx_best_so_far = -1;
    for i = (index + min_dist ) to (index + max_dist) {
        if max_N[i] > best_so_far {
            idx_best_so_far = i;
            best_so_far = max_N[i];
        }
    }
    index = idx_best_so_far;
    trend[num_trendlines] = index;
    num_trendlines = num_trendlines + 1;
}
```

Figure 4.6: Trendline Establishing Algorithm.

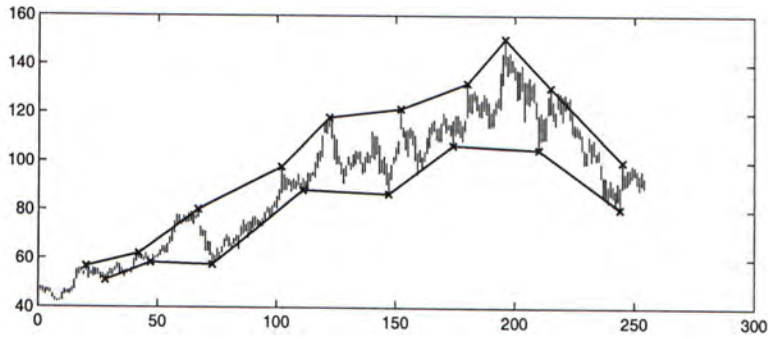


Figure 4.7: Successions of upper and lower trendlines drawn using $min_dist = 15$ days and $max_dist = 35$ days.

is composed of an upper trendline and a lower trendline which are respectively formed by two relative highs and two relative lows. The time periods of the upper trendline and lower trendline must overlap in order to form a trendline pair. Let $U = (RH_1, \dots, RH_n)$ be the series of n relative highs for establishing a succession of upper trendlines and $L = (RL_1, \dots, RL_m)$ be the series of m relative lows for establishing a succession of lower trendlines where $RH_i = (RHT_i, RHP_i)$ and $RL_j = (RLT_j, RLP_j)$. RHT_i and RLT_j denote the time value of the i -th relative high in U and the j -th relative low in L respectively, and RHP_i and RLP_j denote the corresponding prices. Therefore, the i -th upper trendline is represented by the line joining (RHT_i, RHP_i) and (RHT_{i+1}, RHP_{i+1}) , and the j -th lower trendline is represented by the line joining (RLT_j, RLP_j) and (RLT_{j+1}, RLP_{j+1}) . These notations are illustrated in Figure 4.8.

The following is a formal definition of a trendline pair.

Definition 16 (Trendline Pair) Given $U = (RH_1, RH_2, \dots, RH_n)$ and $L = (RL_1, RL_2, \dots, RL_m)$, any consecutive pairs of relative highs (RH_i, RH_{i+1}) and relative lows (RL_j, RL_{j+1}) , where $1 \leq i < m$ and $1 \leq j < n$, form a trendline pair if it satisfies the overlapping condition: $\{RLT_j < RHT_{i+1} \wedge RHT_i < RLT_{j+1}\}$.

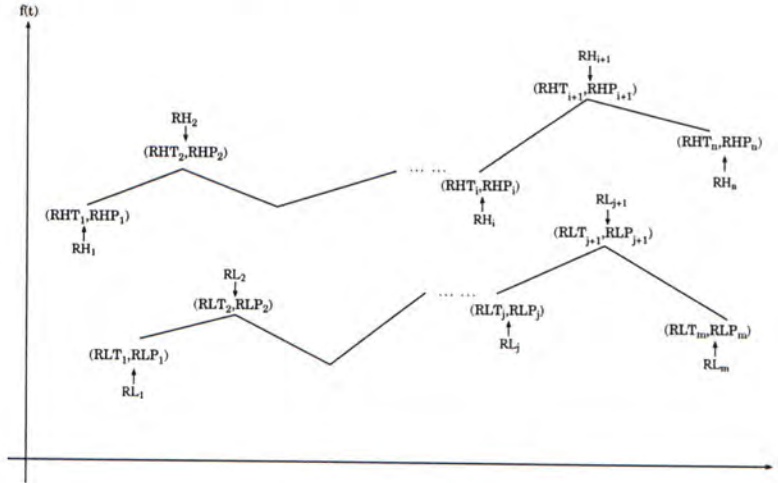
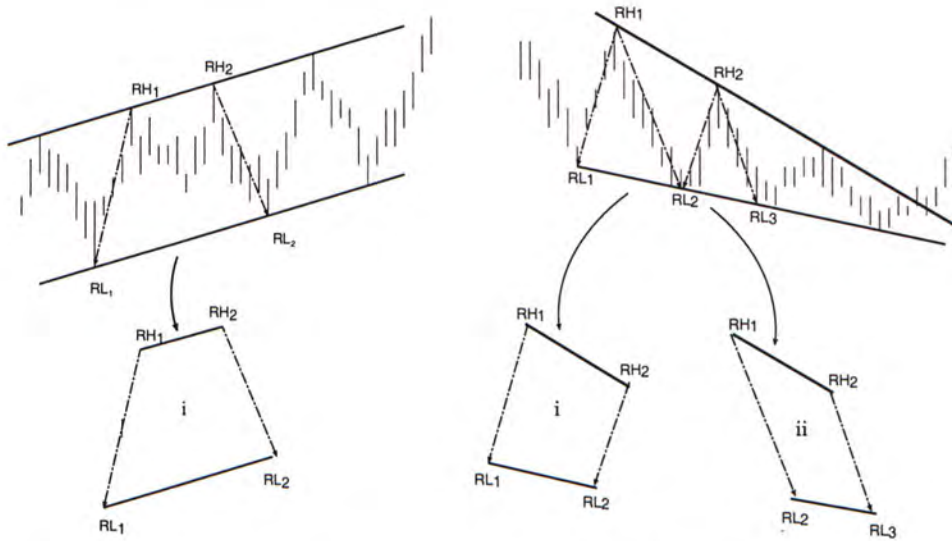


Figure 4.8: Notations for relative highs and relative lows which form a price envelope.

A trendline can be involved in forming different trendline pairs depending on the values of *min_dist* and *max_dist* used in forming the upper and lower trendline. In general, a trendline can project up to $\lfloor \frac{max_dist}{min_dist} \rfloor$ trendline pairs. Figure 4.9 shows three scenarios in which one, two and three trendline pairs are projected from an upper trendline. Figure 4.10 shows the pseudocode for generating a full set of trendline pairs from a succession of upper trendlines and lower trendlines. Starting from the leftmost upper trendline, every upper trendline is compared with one or more lower trendlines. The variable *first* points to the leftmost relative low of the first lower trendline to be compared with an upper trendline. If either the overlapping condition is satisfied or the upper trendline lags behind, the next adjacent lower trendline will be retrieved and compared. Figure 4.11 illustrates this continual condition.

Whenever the overlapping condition is satisfied, the variable *first* is updated to the first relative low of the lower trendline pair which is included in the most recently formed trendline pair. Figure 4.12 shows the value of *first* after pro-

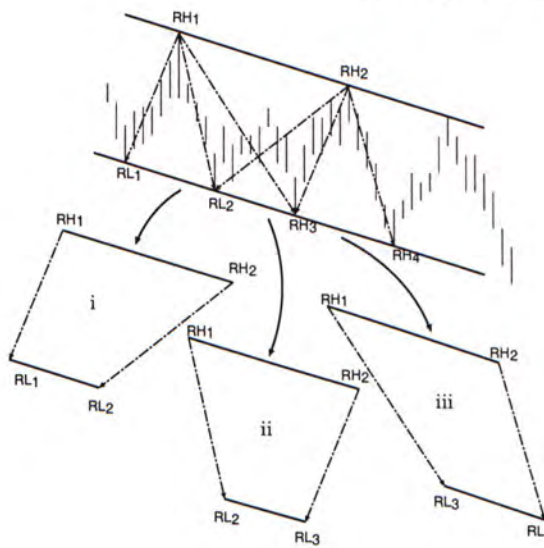


(a) One trendline pair:

- i. $[(RH_1, RH_2), (RL_1, RL_2)]$

(b) Two trendline pairs:

- i. $[(RH_1, RH_2), (RL_1, RL_2)]$
- ii. $[(RH_1, RH_2), (RL_2, RL_3)]$



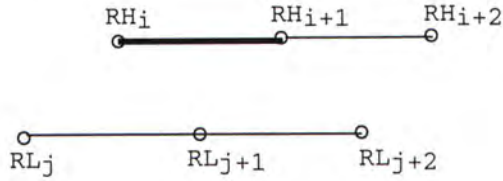
(c) Three trendline pairs:

- i. $[(RH_1, RH_2), (RL_1, RL_2)]$
- ii. $[(RH_1, RH_2), (RL_2, RL_3)]$
- iii. $[(RH_1, RH_2), (RL_3, RL_4)]$

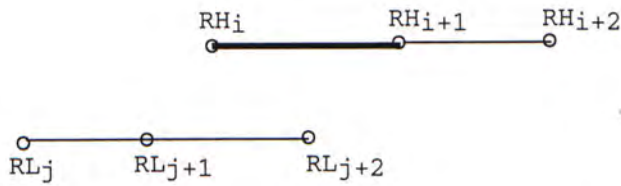
Figure 4.9: Examples of trendline pairs projected from a trendline formed by (RH_1, RH_2) .

Generating Trendline Pairs
<u>Input</u> : $U = (RH_1, RH_2, \dots, RH_n), L = (RL_1, RL_2, \dots, RL_m)$
<u>Output</u> : <i>trendline_pair</i> [1 \rightarrow <i>num_pairs</i>]
<pre> <i>first</i> = 1; <i>num_pairs</i> = 0; for <i>i</i> = 1 to <i>n</i> - 1 for <i>j</i> = <i>first</i> to <i>m</i> - 1 if ($RLT_j < RHT_{i+1}$ and $RHT_i < RLT_{j+1}$) <i>trendline_pair</i>[<i>num_pairs</i>] = $\{(RH_i, RH_{i+1}), (RL_j, RL_{j+1})\}$; <i>first</i> = <i>j</i>; <i>num_pairs</i>++; else if ($RLT_j > RHT_{i+1}$) break inner for and proceed to next relative high end if end for end for </pre>

Figure 4.10: Pseudocode of Generating Trendline Pairs



(a) The i -th upper trendline overlaps with the j -th lower trendline.



(b) The i -th upper trendline lags behind the j -th lower trendline.

Figure 4.11: Continual conditions in which the $(j + 1)$ -th lower trendline can be retrieved and potentially form a trendline pair with the i -th upper trendline

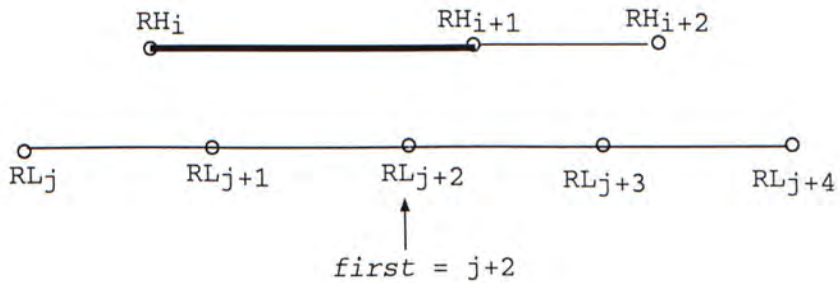


Figure 4.12: An example on how *first* is updated

jecting all possible trendline pairs from the upper trendline formed by RH_i and RH_{i+1} . Therefore, by keeping track of the most recently formed trendline pair, any further retrieval of lower trendlines is obviated. Once the overlapping condition is no longer satisfied when proceeding to the next lower trendline, the next adjacent upper trendline is retrieved. Again, the first lower trendline to be compared is formed by the *first*-th and the (*first* + 1)-th relative lows.

4.3 Three-Phase Pattern Classification




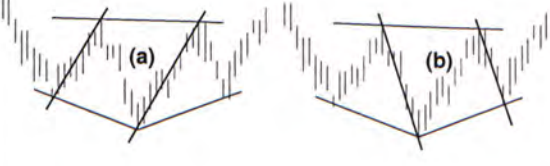
Having the price envelope sliced into trendline pairs (the basic units), a progressive pattern classification can be carried out. Basically, the pattern classification is divided into three phases. In the first phase, every trendline pair is inspected individually and a preliminary classification is made. Merging of similar adjacent trendline pairs will be done in the second phase and at the same time unmerged patterns may be rejected. The final phase attempt to merge trendline pairs which are either unclassified in the first phase or rejected in the second phase. Detailed descriptions of the three phases are given in the following subsections.



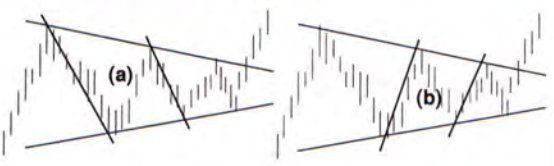


4.3.1 Phase One: Trendline Pair Classification

In this phase, each trendline pair is classified by inspecting the slopes of its upper and lower trendlines. The classification is based on the sign of the slope of the upper trendlines and lower trendlines as well as their slope differences. A careful study on how to classify those chart patterns using trendline pairs is needed. Except double (or triple) top (or bottom), all patterns shown in Figures 4.2 and 4.3 can be sliced into basic trendline pairs. It can be shown that trendline pairs extracted from different charts have different properties in terms of slope.

This analysis is summarized in Table 4.1. The first column shows different types of chart patterns and their corresponding sloping features are presented in the second column. In this table, M_{upper} and M_{lower} denote the slopes of an upper trendline and a lower trendline respectively. To facilitate the explanation, each type of patterns is presented with two identical examples which is formed by minimal numbers of relative highs and relative lows.

Table 4.1: Studying chart pattern features by investigating the slopes.

<i>Chart Patterns</i>	<i>Sloping Features</i>
Falling Wedge 	Trendline pairs a and b: <ul style="list-style-type: none"> · $M_{upper} < 0$ · $M_{lower} < 0$ · $M_{upper} > M_{lower}$
Rising Wedge 	Trendline pairs a and b: <ul style="list-style-type: none"> · $M_{upper} > 0$ · $M_{lower} > 0$ · $M_{upper} < M_{lower}$
Head and Shoulders Top 	Trendline pair a: <ul style="list-style-type: none"> · $M_{upper} < 0$ Trendline pair b: <ul style="list-style-type: none"> · $M_{upper} > 0$
Head and Shoulders Bottom 	Trendline pair a: <ul style="list-style-type: none"> · $M_{lower} < 0$ Trendline pair b: <ul style="list-style-type: none"> · $M_{lower} > 0$
<i>continued on next page</i>	

<i>continued from previous page</i>	
<i>Chart Patterns</i>	<i>Sloping Features</i>
<p>Ascending Triangle</p> 	<p>Trendline pairs a and b:</p> <ul style="list-style-type: none"> · $M_{upper} \approx 0$ · $M_{lower} > 0$
<p>Descending Triangle</p> 	<p>Trendline pairs a and b:</p> <ul style="list-style-type: none"> · $M_{upper} < 0$ · $M_{lower} \approx 0$
<p>Symmetrical Triangle</p> 	<p>Trendline pairs a and b:</p> <ul style="list-style-type: none"> · $M_{upper} < 0$ · $M_{lower} > 0$ · $M_{upper} \approx M_{lower}$
<p>Pricing Channel</p> 	<p>Trendline pairs a and b:</p> <ul style="list-style-type: none"> · $M_{upper} \approx M_{lower}$ · $M_{upper} \neq 0$ · $M_{lower} \neq 0$
<p>Rectangle</p> 	<p>Trendline pairs a and b :</p> <ul style="list-style-type: none"> · $M_{upper} \approx 0$ · $M_{lower} \approx 0$

Excluding Head and Shoulders Top (Bottom), it can be seen in Table 4.1 that the sloping features of other chart patterns are mutually exclusive. As there is no restriction on the slope of the necklines (M_{upper} for Head and Shoulders Top and M_{lower} for Head and Shoulders Bottom), hence it is impossible to identify

the uniqueness of trendline pairs exist in these types of pattern. As an example, if trendline pair (b) of Head and Shoulders Top have sloping features $\{M_{upper} < 0, M_{lower} \approx 0\}$, then this trendline pair would confuse with that of Descending Triangle during the course of classification. To avoid ambiguity, we propose to leave out potential trendline pairs for Head and Shoulders patterns in this phase. Such a strategy does not misclassify nor omit any valid chart patterns, the arguments are as follows:

- §1. *If a Head and Shoulder (either Top or Bottom) pattern exists in the chart but one of its trendline pair is being classified as other patterns, there should be no mis-classification nor omission of valid chart patterns.*

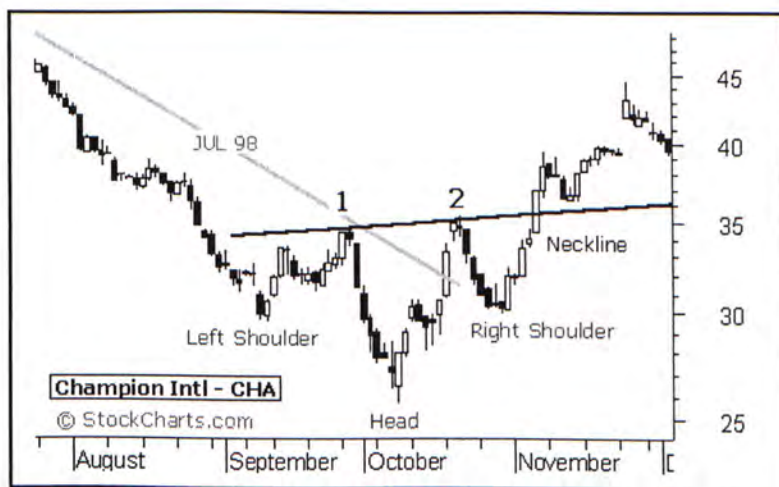
We show below that if a Head and Shoulders (Top or Bottom) pattern is valid, none of its trendline pairs can overlap with that of any other valid chart patterns. As overlapping of chart patterns is impossible, a wrongly classified trendline pair, which belongs to a valid Head and Shoulders Pattern, would be eventually rejected as it cannot be merged in Phase two.

As stated in [Sch98], the Head and Shoulders (Top or Bottom) is not considered complete until the neckline is penetrated and it should be formed after a major price movement has occurred. Therefore, patterns that bear the shape of a head-and-shoulders formation, but lack of these requirements, may be misleading and should be disqualified. To illustrate valid Head and Shoulders patterns graphically, we adapted two charts from [sto] as shown in Figure 4.13.

To study whether an overlapped head and shoulders pattern fulfils those requirements, we show all patterns possibly bearing the shape of a head-and-shoulders and at the same time overlap with a valid chart pattern in Tables 4.2 and 4.3. In these tables, the head-and-shoulders are marked by



(a) Head and Shoulders Top formed after a major up movement of price (as marked by the grey trendline) with the neckline being penetrated by the down movement of price.



(b) Head and Shoulders Bottom formed after a major down movement of price (as marked by the grey trendline) with the neckline being penetrated by the up movement of price.

Figure 4.13: Valid Head and Shoulders pattern adapted from [sto].

bold arcs while its neckline is marked by a bold line. It can be seen from the tables that even if a pattern exhibits a head-and-shoulders shape, it is unqualified if it coexists with another chart pattern whose upper trendline overlaps either (1) on the left, the previous trendline is the same as the neckline, meaning a premature anticipation of head-and-shoulders formation, or (2) on the right, the neckline remains the trendline of latter price movement, meaning a premature completion of head-and-shoulders formation.

§2. *If a Head and Shoulder (Top or Bottom) pattern exists in the chart but none of its trendline pairs is being classified, there should be no false dismissals.*

The unclassified trendline pairs which form a valid Head and Shoulders pattern will be merged together in the third phase of classifications.

In Phase one, therefore, every trendline pair is either unclassified or only classified into one of the following patterns:

- 1 Rectangle
- 2 Price Channel
- 3 Rising Wedge
- 4 Falling Wedge
- 5 Descending Triangle
- 6 Ascending Triangle
- 7 Symmetrical Triangle

The classification requires two user inputs, namely the *zero-sloping threshold* σ and *equality threshold* ψ ($\sigma, \psi > 0$). The *zero-sloping threshold*, σ , indicates

Table 4.2: Patterns overlap with patterns that bear the shape of Head-and-Shoulders Bottom.

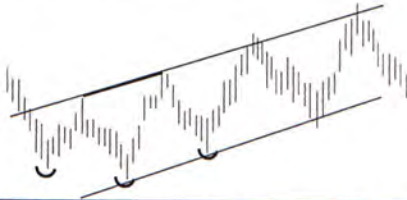
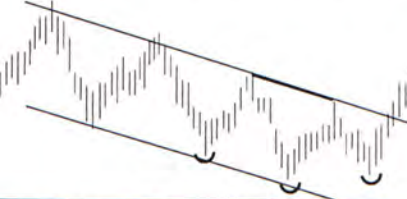

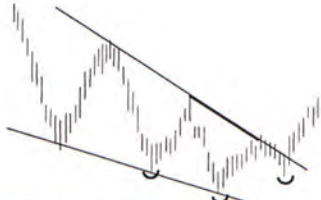
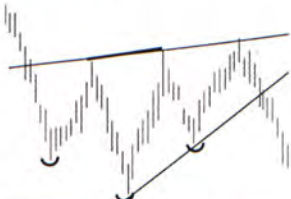
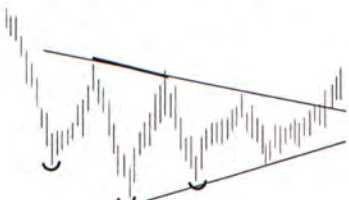
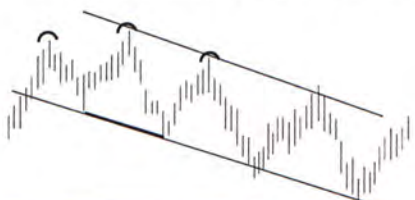
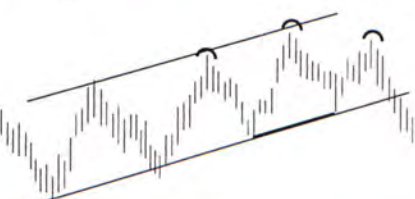


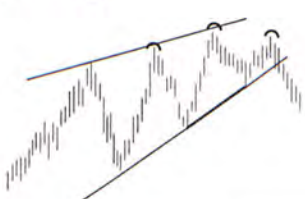

<i>Chart Patterns</i>	<i>Sloping Features</i>
Pricing Channel (Slopes Up) 	$M_{upper} > 0$ $M_{lower} > 0$ $M_{upper} \approx M_{lower}$
Pricing Channel (Slopes Down) 	$M_{upper} < 0$ $M_{lower} < 0$ $M_{upper} \approx M_{lower}$
Ascending Triangle 	$M_{upper} \approx 0$ $M_{lower} > 0$
Falling Wedge 	$M_{upper} < 0$ $M_{lower} < 0$ $ M_{upper} > M_{lower} $
Rising Wedge 	$M_{upper} > 0$ $M_{lower} > 0$ $ M_{upper} < M_{lower} $
Symmetrical Triangle 	$M_{upper} > 0$ $M_{lower} < 0$ $ M_{upper} \approx M_{lower} $

Table 4.3: Patterns overlap with patterns that bear the shape of Head-and-Shoulders Top

<i>Chart Patterns</i>	<i>Sloping Features</i>
Pricing Channel (Slopes Down) 	$M_{upper} < 0$ $M_{lower} < 0$ $M_{upper} \approx M_{lower}$
Pricing Channel (Slopes Up) 	$M_{upper} > 0$ $M_{lower} > 0$ $M_{upper} \approx M_{lower}$
Ascending Triange 	$M_{upper} < 0$ $M_{lower} \approx 0$
Falling Wedge 	$M_{upper} < 0$ $M_{lower} < 0$ $ M_{upper} > M_{lower} $
Rising Wedge 	$M_{upper} > 0$ $M_{lower} > 0$ $ M_{upper} < M_{lower} $
Symmetrical Triangle 	$M_{upper} > 0$ $M_{lower} < 0$ $ M_{upper} \approx M_{lower} $

user's perception on how a trendline with non-zero slope can be considered as a horizontal line; The *equality threshold*, ψ , indicates user's perception on how two trendlines with different slopes can be considered to have the same slope. Given these two thresholds, Figure 4.14 shows the pseudocode for classifying a trendline pair according to its sloping feature $\{M_{upper}, M_{lower}\}$. The slopes of the upper and lower trendlines are set to zero if they are bounded by the *zero-sloping threshold*. Also, if the absolute slope difference between the upper and lower trendlines is within the *equality threshold*, the upper and lower trendlines are considered parallel.

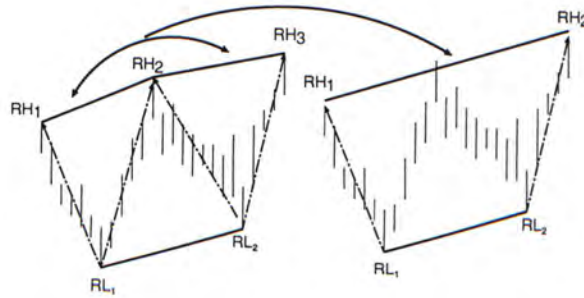
4.3.2 Phase Two: Patterns Merging and Rejection

In phase one, every trendline pair is processed and classified into one of the seven patterns, namely Rectangle, Price Channel, Rising Wedge, Falling Wedge, Descending Triangle, Ascending Triangle and Symmetrical Triangle. They are readily to be merged to form valid chart patterns. If two consecutive trendline pairs being previously classified as the same pattern, they can be potentially merged to form one single trendline pair. As an example, Figure 4.15 shows the merging of two adjacent trendline pairs which are preliminary classified to the same pattern either by (a) combining the upper trendlines or (b) combining the lower trendlines. The merging strategy must ensure the merged pattern can (1) sustain the properties of the original pattern as which the previous trendline pairs are classified, (2) bound all data points (highs and lows) properly.

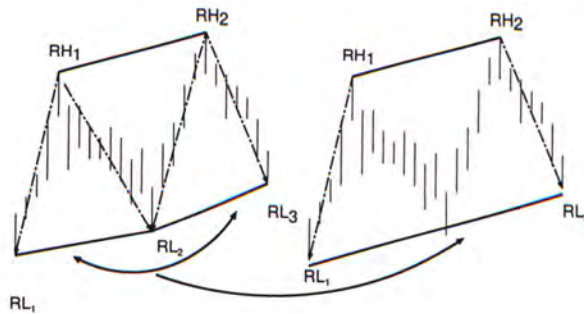
In merging two adjacent trendline pairs, we propose two strategies. The *Greedy Approach* is for merging trendline pair which are classified as Falling Wedge or Rising Wedge; the *Best-Fitting Line* approach is for merging trendline pairs which are classified as Rectangle, Price Channel, Symmetrical Triangle,

Classifying Trendline Pair
<u>Input:</u> $\sigma, \psi, M_{upper}, M_{lower}$
<u>Output:</u> <i>pattern_type</i>
<pre> If $M_{upper} \leq \sigma$ Then $M_{upper} = 0$; End if If $M_{lower} \leq \sigma$ Then $M_{lower} = 0$; End if If $M_{lower} == 0$ and $M_{lower} == 0$ Then $pattern_type = 'Rectangle'$; Else if $M_{upper} - M_{lower} \leq \psi$ Then $pattern_type = 'Pricing Channel'$; Else if $M_{upper} > 0$ and $M_{lower} > 0$ and $M_{lower} > M_{upper}$ Then $pattern_type = 'Rising Wedge'$; Else if $M_{upper} < 0$ and $M_{lower} < 0$ and $M_{lower} < M_{upper}$ Then $pattern_type = 'Falling Wedge'$; Else if $M_{lower} == 0$ and $M_{upper} < 0$ Then $pattern_type = 'Descending Triangle'$; Else if $M_{upper} == 0$ and $M_{lower} > 0$ Then $pattern_type = 'Ascending Triangle'$; Else if $M_{upper} < 0$ and $M_{lower} > 0$ and $M_{upper} + M_{lower} \leq \psi$ Then $pattern_type = 'Symmetrical Triangle'$; Else $pattern_type = 'Nil'$; End if </pre>

Figure 4.14: Pseudocode of classifying trendline pairs into chart patterns



(a) Merge two trendline pairs with their lower trendlines overlapped.



(b) Merge two trendline pairs with their upper trendlines overlapped.

Figure 4.15: Different merging of trendline pairs who have been classified to the same pattern.

Ascending Triangle and Descending Triangle.

Merging Strategy 1 (Greedy Approach for Merging Upper Trendlines)

Given two upper trendlines formed by any three Relative Highs: (RHP_1, RHT_1) , (RHP_2, RHT_2) and (RHP_3, RHT_3) , the formation of the merged trendline depends on the slopes of these two upper trendlines. If the slope of the first trendline is smaller than that of the second trendline $\left(\frac{RHP_1 - RHP_2}{RHT_1 - RHT_2} < \frac{RHP_2 - RHP_3}{RHT_2 - RHT_3}\right)$, then the merged trendline is equivalent to the line joining (RHP_1, RHT_1) and (RHP_3, RHT_3) ; If the slope of the second trendline is smaller than that of the first trendline $\left(\frac{RHP_1 - RHP_2}{RHT_1 - RHT_2} > \frac{RHP_2 - RHP_3}{RHT_2 - RHT_3}\right)$, then the merged trendline is equivalent to the second trendline extending from RHT_2 to RHT_1 .

Merging Strategy 2 (Greedy Approach for Merging Lower Trendlines)

Given two lower trendlines formed by any three Relative Lows: (RLP_1, RLT_1) , (RLP_2, RLT_2) and (RLP_3, RLT_3) , the formation of the merged trendline depends on the slopes of these two lower trendlines. If the slope of the first trendline is smaller than that of the second trendline $\left(\frac{RLP_1 - RLP_2}{RLT_1 - RLT_2} > \frac{RLP_2 - RLP_3}{RLT_2 - RLT_3}\right)$, then the merged trendline is equivalent to the line joining (RLP_1, RLT_1) and (RLP_3, RLT_3) ; If the slope of the second trendline is smaller than that of the first trendline $\left(\frac{RLP_1 - RLP_2}{RLT_1 - RLT_2} < \frac{RLP_2 - RLP_3}{RLT_2 - RLT_3}\right)$, then the merged trendline is equivalent to the second trendline extending from RLT_2 to RLT_1 .

Using the Greedy Approach, Tables 4.4 and 4.5 show respectively the merging of upper and lower trendlines that form different Falling Wedge or Rising Wedge patterns. In these tables, m' and m'' denote the slopes of the first and the second trendlines which are undergoing the merging process. The second and the third columns show two scenarios of merging, that is, when $m' < m''$ and $m' > m''$ and each merged trendline is represented by a dotted line. The Greedy Approach ensures the merged trendline pair retains the property of a Falling Wedge or a Rising Wedge.

Merging Strategy 3 (The Best-Fitting Line Approach) *This approach applies to the merging of the upper or lower trendlines. For simplicity, we use a common notation to represent relative high and relative low. A 2-D point (x_i, y_i) represents the time and price of either a relative high or relative low. Given two consecutive trendlines formed by three points: (x_0, y_0) , (x_1, y_1) and (x_2, y_2) , the least squares method [Edw76] is used to estimate the intercept β_0 and the slope β_1 of the best-fitting line so that the sum of the squares of the differences between the price y_i and the best fitting line, $y = \beta_0 + \beta_1 x$, is a minimum. Therefore, the*

Table 4.4: Greedy Approach for Merging Upper Trendlines.

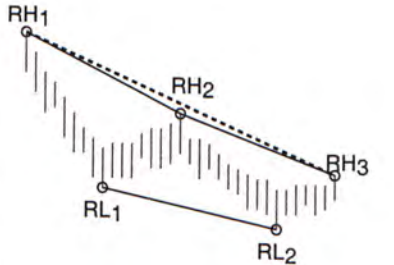
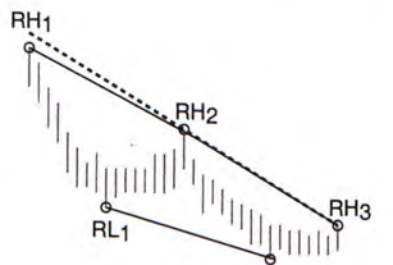
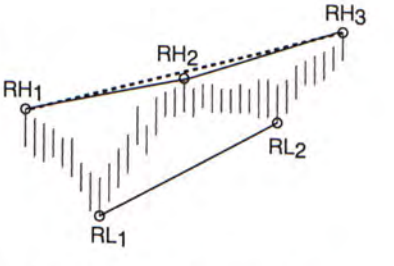
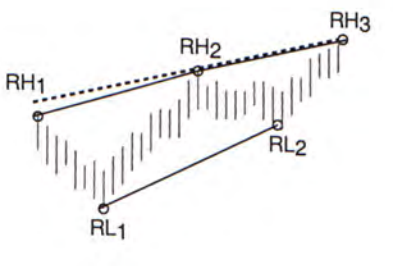
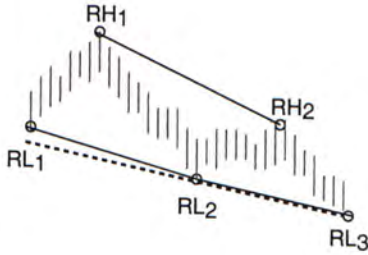
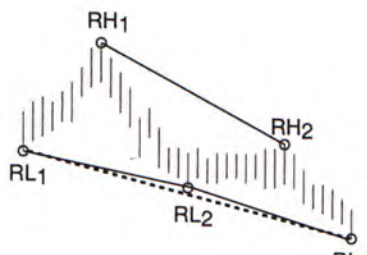
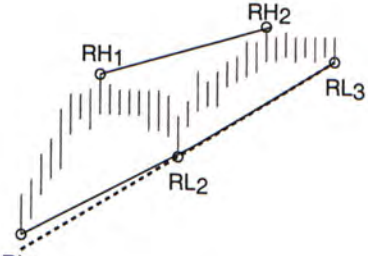
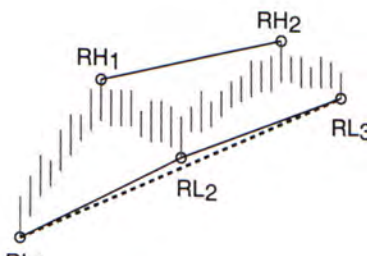
Chart Pattern	$m' < m''$	$m' > m''$
<p>Falling Wedge</p>	 <p> $m' \leq M_{lower} \wedge m'' \leq M_{lower}$ $\Rightarrow M'_{upper} \leq M_{lower}$ </p>	 <p> $m'' \leq M_{lower}$ $M'_{upper} = m''$ $\Rightarrow M'_{upper} \leq M_{lower}$ </p>
<p>Rising Wedge</p>	 <p> $m' \leq M_{lower} \wedge m'' \leq M_{lower}$ $\Rightarrow M'_{upper} \leq M_{lower}$ </p>	 <p> $m'' \leq M_{lower}$ $M'_{upper} = m''$ $\Rightarrow M'_{upper} \leq M_{lower}$ </p>

Table 4.5: Greedy Approach for Merging Lower Trendlines.

Chart Pattern	$m' < m''$	$m' > m''$
<p>Falling Wedge</p>	 <p>The diagram shows a falling wedge with two trendlines. The upper trendline is solid and connects points RH1 and RH2. The lower trendline is dashed and connects points RL1, RL2, and RL3. The slope of the lower trendline is shallower than that of the upper trendline.</p> $m'' \geq M_{upper}$ $M'_{lower} = m''$ $\Rightarrow M'_{lower} \geq M_{upper}$	 <p>The diagram shows a falling wedge with two trendlines. The upper trendline is solid and connects points RH1 and RH2. The lower trendline is dashed and connects points RL1, RL2, and RL3. The slope of the lower trendline is steeper than that of the upper trendline.</p> $m' \geq M_{upper} \wedge m'' \geq M_{upper}$ $\Rightarrow M'_{lower} \geq M_{upper}$
<p>Rising Wedge</p>	 <p>The diagram shows a rising wedge with two trendlines. The upper trendline is solid and connects points RH1 and RH2. The lower trendline is dashed and connects points RL1, RL2, and RL3. The slope of the lower trendline is shallower than that of the upper trendline.</p> $m'' \geq M_{upper}$ $M'_{lower} = m''$ $\Rightarrow M'_{lower} \geq M_{upper}$	 <p>The diagram shows a rising wedge with two trendlines. The upper trendline is solid and connects points RH1 and RH2. The lower trendline is dashed and connects points RL1, RL2, and RL3. The slope of the lower trendline is steeper than that of the upper trendline.</p> $m' \geq M_{upper} \wedge m'' \geq M_{upper}$ $\Rightarrow M'_{lower} \geq M_{upper}$

regression model is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 0, 1, 2$$

where ϵ_i is the error component with zero mean and unknown variance. The least square criterion is

$$S(\beta_0, \beta_1) = \sum_{i=0}^2 (y_i - \beta_0 - \beta_1 x_i)^2.$$

The least square estimators of β_0 and β_1 , denoted by $\hat{\beta}_0$ and $\hat{\beta}_1$ respectively, must satisfy

$$\left. \frac{\partial S}{\partial \beta_0} \right|_{\hat{\beta}_0, \hat{\beta}_1} = -2 \sum_{i=0}^2 (y_i - \beta_0 - \beta_1 x_i) = 0$$

and

$$\left. \frac{\partial S}{\partial \beta_1} \right|_{\hat{\beta}_0, \hat{\beta}_1} = -2 \sum_{i=0}^2 (y_i - \beta_0 - \beta_1 x_i) x_i = 0.$$

Simplifying these two equations yields

$$3\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=0}^2 x_i = \sum_{i=0}^2 y_i \quad (4.1)$$

$$\hat{\beta}_0 \sum_{i=0}^2 x_i + \hat{\beta}_1 \sum_{i=0}^2 (x_i)^2 = \sum_{i=0}^2 (x_i)(y_i). \quad (4.2)$$

Solving Equations 4.1 and 4.2 yields the solutions

$$\hat{\beta}_0 = \bar{y}_i - \hat{\beta}_1 \bar{x} \quad (4.3)$$

and

$$\hat{\beta}_1 = \frac{\sum_{i=0}^2 (y_i)(x_i) - \frac{\left(\sum_{i=0}^2 y_i\right) \left(\sum_{i=0}^2 x_i\right)}{3}}{\sum_{i=0}^2 (x_i)^2 - \frac{\left(\sum_{i=0}^2 x_i\right)^2}{3}} \quad (4.4)$$

where

$$\bar{y} = \frac{1}{3} \sum_{i=0}^2 y_i \quad \text{and} \quad \bar{x} = \frac{1}{3} \sum_{i=0}^2 x_i.$$

Therefore, $\hat{\beta}_0$ and $\hat{\beta}_1$ in Equations 4.3 and 4.4 are the least squares estimators of the intercept and slope. The best fitting line is then

$$y = \hat{\beta}_0 + \hat{\beta}_1 x.$$

For a trendline pair to be qualified as any one of the chart patterns, its lower and upper trendlines should possess the sloping feature of a particular pattern as described in Table 4.1. Therefore, the resultant trendline pair merged from two trendline pairs should also retain the sloping feature, otherwise another unexpected or invalid pattern would be formed.

Before showing that Merging Strategy 3 can preserve the sloping feature of the patterns: Rectangle, Price Channel and Symmetrical Triangle, we need to understand the relationship among the slopes of the two trendlines being merged and also that of the trendline resulted from merging. For the ease of presentation, the three points (either all Relative Highs or Relative Lows) forming two consecutive trendlines are all translated by $(-x_0, -y_0)$ such that the first point (x_0, y_0) becomes the origin. After such a translation, the three points are now re-denoted as $(0,0)$, (x'_1, y'_1) and (x'_2, y'_2) where $x'_1 = x_1 - x_0$, $y'_1 = y_1 - y_0$, $x'_2 = x_2 - x_0$ and $y'_2 = y_2 - y_0$. Suppose the first trendline formed by joining $(0,0)$ and (x'_1, y'_1) has a slope of m while the second trendline formed by joining (x'_1, y'_1) and (x'_2, y'_2) has a slope of $m + \Delta$, where Δ is a real number indicating the slope difference between the first and the second trendline pairs. Figure 4.3.2 illustrates the notations after this translation.

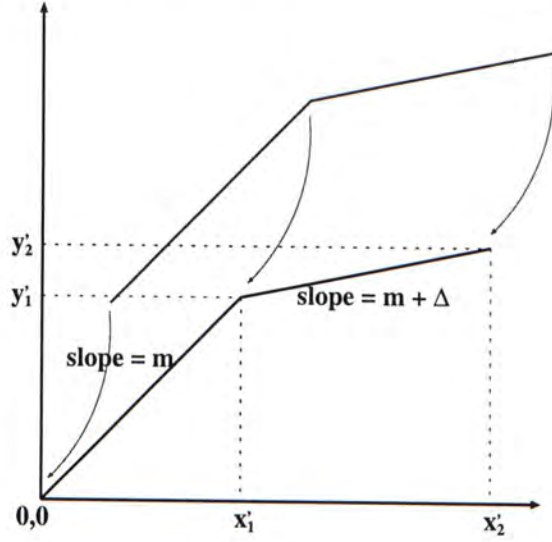


Figure 4.16: Translation of two consecutive trendlines such that the first point touches the origin.

To find out the relationship among m , $m + \Delta$ and $\hat{\beta}_1$ (slope of the best-fitting line), we first write y_1 and y_2 in terms of the slopes and the corresponding x -axis values

$$y'_1 = mx'_1, \quad (4.5)$$

$$y'_2 = mx'_1 + (m + \Delta)(x'_2 - x'_1). \quad (4.6)$$

Then, we expand Equation 4.4 using $(0, 0)$, (x'_1, y'_1) and (x'_2, y'_2) ,

$$\begin{aligned} \hat{\beta}_1 &= \frac{x'_1 y'_1 + x'_2 y'_2 - \frac{(y'_1 + y'_2)(x'_1 + x'_2)}{3}}{x_1'^2 + x_2'^2 - \frac{(x'_1 + x'_2)^2}{3}} \\ &= \frac{2x'_1 y'_1 + 2x'_2 y'_2 - y'_2 x'_1 - y'_1 x'_2}{2x_1'^2 + 2x_2'^2 - 2x'_1 x'_2} \\ &= \frac{y'_1(2x'_1 - x'_2) + y'_2(2x'_2 - x'_1)}{2x_1'^2 + 2x_2'^2 - 2x'_1 x'_2}. \end{aligned} \quad (4.7)$$

Substitute y'_1 and y'_2 from Equations 4.5 and 4.6 into Equation 4.7 gives

$$\begin{aligned}
 \hat{\beta}_1 &= \frac{mx'_1(2x'_1 - x'_2) + [mx'_1 + (m + \Delta)(x'_2 - x'_1)](2x'_2 - x'_1)}{2x_1'^2 + 2x_2'^2 - 2x_1'x_2'} \\
 &= \frac{mx'_1(x'_1 + x'_2) + (m + \Delta)(x'_2 - x'_1)(2x'_2 - x'_1)}{2x_1'^2 + 2x_2'^2 - 2x_1'x_2'} \\
 &= \frac{m[2x_1'^2 + 2x_2'^2 - 2x_1'x_2'] + \Delta(x'_2 - x'_1)(2x'_2 - x'_1)}{2x_1'^2 + 2x_2'^2 - 2x_1'x_2'} \\
 &= m + \frac{\Delta(2x_2'^2 + x_1'^2 - 3x_1'x_2')}{2x_1'^2 + 2x_2'^2 - 2x_1'x_2'}. \tag{4.8}
 \end{aligned}$$

We conclude from Equation 4.8 that the slope of the merged trendline satisfies the following set of inequalities

$$\begin{cases} m \leq \hat{\beta}_1 \leq m + \Delta & \text{if } \Delta \geq 0, \\ m + \Delta \leq \hat{\beta}_1 \leq m & \text{if } \Delta \leq 0. \end{cases} \tag{4.9}$$

Lemma 1 *If two trendline pairs individually form Price Channels and both of them have a common trendline with slope m' , the resultant trendline pair after applying Merging Strategy 3 also forms a Price Channel, that is, the following criterion always holds:*

$$|m' - \hat{\beta}_1| \leq \psi. \tag{4.10}$$

Proof: If the two trendline pairs are classified to the pattern Price Channel, the following set of inequalities must hold:

$$\begin{cases} |m' - m| \leq \psi \\ |m' - (m + \Delta)| \leq \psi \end{cases}$$

or we can write

$$\begin{cases} -\psi \leq m' - m \leq \psi \\ -\psi \leq m' - (m + \Delta) \leq \psi. \end{cases} \tag{4.11}$$

According to the sign of Δ and using the result concluded from Equation 4.9, we divide the proof into two cases:

Case 1: when $\Delta \geq 0$,

$$\begin{aligned} m &\leq \hat{\beta}_1 \leq m + \Delta \\ \Rightarrow -(m + \Delta) &\leq -\hat{\beta}_1 \leq -m \\ \Rightarrow m' - (m + \Delta) &\leq m' - \hat{\beta}_1 \leq m' - m, \end{aligned}$$

implied by Equation 4.11, we can state

$$\begin{aligned} -\psi &\leq m' - \hat{\beta}_1 \leq \psi \\ \Rightarrow |m' - \hat{\beta}_1| &\leq \psi. \end{aligned}$$

Case 2: when $\Delta \leq 0$,

$$\begin{aligned} m + \Delta &\leq \hat{\beta}_1 \leq m \\ \Rightarrow -m &\leq -\hat{\beta}_1 \leq -(m + \Delta) \\ \Rightarrow m' - m &\leq m' - \hat{\beta}_1 \leq m' - (m + \Delta), \end{aligned}$$

implied by Equation 4.11, we can again state

$$\begin{aligned} -\psi &\leq m' - \hat{\beta}_1 \leq \psi \\ \Rightarrow |m' - \hat{\beta}_1| &\leq \psi. \end{aligned}$$

□

Lemma 2 *If two trendline pairs individually form a Symmetrical Triangles and both of them have a common trendline with slope m' , the resultant trendline pair after applying Merging Strategy 3 also form a Symmetrical Triangle, that is, the following criterion always holds:*

$$|m' + \hat{\beta}_1| \leq \psi. \quad (4.12)$$

Proof: If the two trendline pairs are classified as Symmetrical Triangles, the following set of inequalities must hold:

$$\begin{cases} |m' + m| \leq \psi \\ |m' + (m + \Delta)| \leq \psi \end{cases}$$

or we can write

$$\begin{cases} -\psi \leq m' + m \leq \psi \\ -\psi \leq m' + (m + \Delta) \leq \psi. \end{cases} \quad (4.13)$$

According to the sign of Δ and using the result concluded from Equation 4.9, we divide the proof into two cases:

Case 1: when $\Delta \geq 0$,

$$\begin{aligned} m &\leq \hat{\beta}_1 \leq m + \Delta \\ \Rightarrow m' + m &\leq m' + \hat{\beta}_1 \leq m' + (m + \Delta), \end{aligned}$$

implied by Equation 4.13, we can state

$$\begin{aligned} -\psi &\leq m' + \hat{\beta}_1 \leq \psi \\ \Rightarrow |m' + \hat{\beta}_1| &\leq \psi. \end{aligned}$$

Case 2: when $\Delta \leq 0$,

$$\begin{aligned} m + \Delta &\leq \hat{\beta}_1 \leq m \\ \Rightarrow m' + (m + \Delta) &\leq m' + \hat{\beta}_1 \leq m' + m, \end{aligned} \quad (4.14)$$

implied by Equation 4.13, we can again state

$$\begin{aligned} -\psi &\leq m' + \hat{\beta}_1 \leq \psi \\ \Rightarrow |m' + \hat{\beta}_1| &\leq \psi. \end{aligned}$$

□

Lemma 3 *If two trendline pairs individually formed Rectangles, the resultant trendline pair after applying Merging Strategy 3 also forms a Rectangle, that is, the following criterion always holds:*

$$|\hat{\beta}_1| \leq \sigma. \quad (4.15)$$

Proof: If the two trendline pairs are classified to the pattern Rectangle, the following inequalities must hold:

$$\begin{cases} |m| \leq \sigma \\ |m + \Delta| \leq \sigma \end{cases}$$

or we can write

$$\begin{cases} -\sigma \leq m \leq \sigma, \\ -\sigma \leq m + \Delta \leq \sigma. \end{cases} \quad (4.16)$$

According to the sign of Δ and using the result concluded from Equation 4.9, we divide the proof into two cases:

Case 1: when $\Delta \geq 0$,

$$m \leq \hat{\beta}_1 \leq m + \Delta, \quad (4.17)$$

implied by Equation 4.3.2, we can state

$$\begin{aligned} -\sigma &\leq \hat{\beta}_1 \leq \sigma \\ \Rightarrow \quad &|\hat{\beta}_1| \leq \sigma. \end{aligned}$$

Case 2: when $\Delta \leq 0$,

$$m + \Delta \leq \hat{\beta}_1 \leq m,$$

implied by Equation 4.3.2, we can again state

$$\begin{aligned} -\sigma &\leq \hat{\beta}_1 \leq \sigma \\ \Rightarrow \quad &|\hat{\beta}_1| \leq \sigma. \end{aligned}$$

□

Lemma 4 *If two trendline pairs individually forms Descending Triangles and either their upper trendline (slope < 0) or lower trendline (slope ≈ 0) is overlapped, the resultant trendline after applying Merging Strategy 3 also forms a Descending Triangle, that is, either one of the following two criteria holds:*

$$\begin{aligned} \hat{\beta}_1 < 0 \quad \text{if upper trendlines are merged} \\ \text{or} \\ \left| \hat{\beta}_1 \right| < \sigma \quad \text{if lower trendlines are merged.} \end{aligned}$$

Proof: If the two trendline pairs are classified as Descending Triangle, one of the following sets of inequalities must hold:

$$m < 0 \text{ and } m + \Delta < 0 \quad \text{if Upper Trendlines are merged,} \quad (4.18)$$

or

$$|m| \leq \sigma \text{ and } |m + \Delta| \leq \sigma \quad \text{if Lower Trendlines are merged.} \quad (4.19)$$

According to the sign of Δ and using the result concluded from Equation 4.9, we divide the proof into two cases:

Case 1: when $\Delta \geq 0$,

$$m \leq \hat{\beta}_1 \leq m + \Delta,$$

then implied by Equations 4.18 and 4.19, we can deduce

$$\begin{aligned} \hat{\beta}_1 < 0 \quad \text{if upper trendlines are merged} \\ \text{or} \\ \left| \hat{\beta}_1 \right| < \sigma \quad \text{if lower trendlines are merged.} \end{aligned}$$

Case 2: when $\Delta \leq 0$,

$$m + \Delta \leq \hat{\beta}_1 \leq m,$$

then implied by Equations 4.18 and 4.19, we can deduce

$$\begin{aligned} \hat{\beta}_1 < 0 & \text{ if upper trendlines are merged} \\ \text{or} \\ \left| \hat{\beta}_1 \right| < \sigma & \text{ if lower trendlines are merged.} \end{aligned}$$

□

Lemma 5 *If two trendline pairs individually form Ascending Triangle and either their upper trendline (slope < 0) or lower trendline (slope ≈ 0) is overlapped, the resultant trendline after applying Merging Strategy 3 also forms an Ascending Triangle, that is, either one of the following two criteria holds:*

$$\begin{aligned} \left| \hat{\beta}_1 \right| < \sigma & \text{ if Upper Trendlines are merged,} \\ \text{or} \\ \hat{\beta}_1 > 0 & \text{ if merging Lower Trendlines are merged.} \end{aligned}$$

Proof: The proof is similar to that of Lemma 4 and so it is left behind. □

The merged trendlines resulting from Merging Strategies 1, 2 and 3 do not necessary touch all the relative highs or relative lows. This allows certain extent of flexibility of pattern formation. However, one may want to restrict this by imposing an allowable distance between the merged trendline and the three relative highs or lows forming it. Therefore, depends on user's requirements or judgement, the pattern results from merging can also be rejected and more criteria can also be put on the resultant pattern.

4.3.3 Phase Three: Patterns Merging of Unclassified and Unmerged Trendline Pairs

In the last phase of patterns searching, an attempt to find meaningful chart patterns by merging those trendline pairs which are not classified in Phase One nor merged in Phase Two. Unlike Phase Two, in which the merging is carried out only if two trendline pairs are overlapped and also both classified to the same type of pattern, in Phase Three two trendline pairs are merged to see if the merged trendline pair can form a valid pattern. In this final phase, three types of chart patterns are searched in the following order:

- 1 Price Channel and Rectangle
- 2 Head and Shoulders (Top or Bottom)
- 3 Double (Top or Bottom) and Triple (Top or Bottom)

Price Channel and Rectangle are searched again in this phase because they can be potentially formed by a combination of unclassified trendline pairs and classified patterns such as Triangles and Wedges. Tables 4.6, 4.7 and 4.8 show the merging of unclassified and classified trendline pairs, and the resultant pattern. These trendline pairs (either classified or unclassified in Phase One) are not classified to Price Channel or Rectangle although they can potentially form such patterns by setting a larger value of the *zero-sloping threshold* σ and the *equality threshold* ψ . Although setting higher values of σ and ψ in the Phase One is feasible, trendline pairs belonging to patterns other than Price Channel or Rectangle may be wrongly classified. Therefore, smaller values of σ and ψ should be used in Phase One to ensure no potential patterns are missed. While in the final phase, we can attempt to merge two adjacent trendline pairs using Merging

Strategy 3 and then check whether the merged patterns form a Pricing Channel or Rectangle.

Head and Shoulders Top (or Bottom) are searched after no Price Channel or Rectangle pattern could be formed by two overlapped trendline pairs. The two shoulders in the pattern formation should be comparably equal which can be adjusted by user input parameter. Double Top (or Bottom) and Triple Top ((or Bottom) are simplest form of pattern which rely only on investigating either the upper trendlines or the lower trendlines to see if they are bounded by the *zero-sloping threshold* σ .

4.4 Results

Experiments on finding chart patterns were carried out using historical data (from 6/16/1999 to 6/15/2000) of S&P 500 stocks. During the trendline preparation, the trendline were drawn using relative highs and relative lows which are at least 10 days and at most 40 days apart. However, in finding (Triple or Double) top and bottom, those relative highs and relative lows are at most 60 days apart as they are usually formed in longer period of time. The equality threshold and zero-sloping threshold were set to 0.05 and 0.008 respectively. Part of the results are shown in this section while a full set of results can be obtained in Appendix A. Examples of chart patterns found in SP500 are shown in Figures 4.17 to 4.28.

Table 4.6: Merging unclassified and unmerged trendline pairs which can form Rectangle.

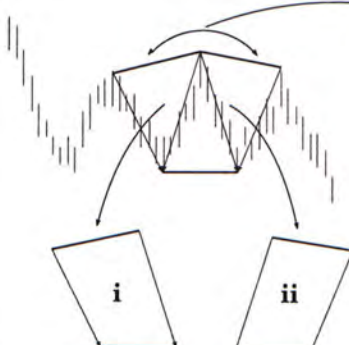
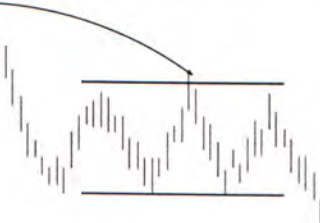
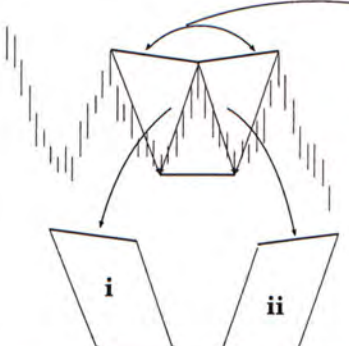

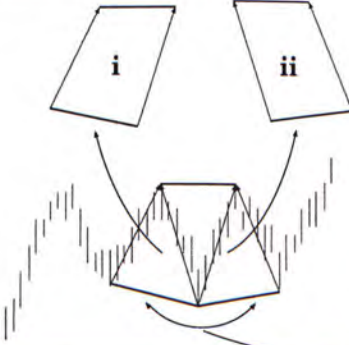

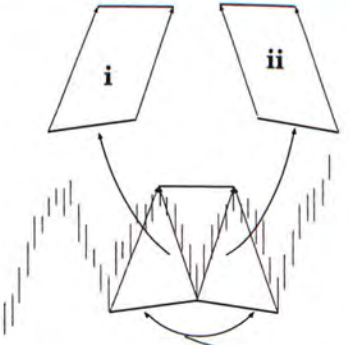

<i>Decomposed Trendline Pairs</i>	<i>Merged Pattern</i>	<i>Phase One Classification</i>
		<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Descending Triangle</i></p>
		<p>Trendline Pair: (i) <i>Descending Triangle</i> (ii) <i>Unclassified</i></p>
		<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Ascending Triangle</i></p>
		<p>Trendline Pair: (i) <i>Ascending Triangle</i> (ii) <i>Unclassified</i></p>

Table 4.7: Merging unclassified and unmerged trendline pairs which can form Price Channel (Slopes Up).

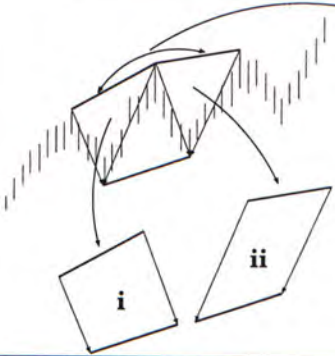
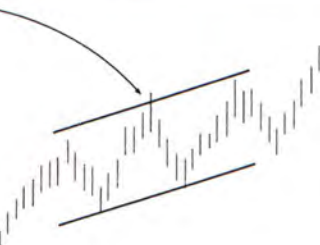
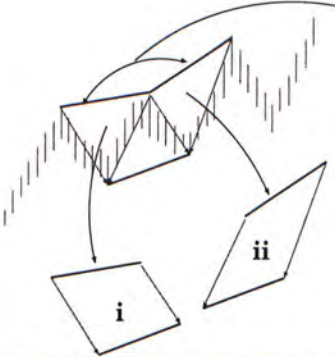
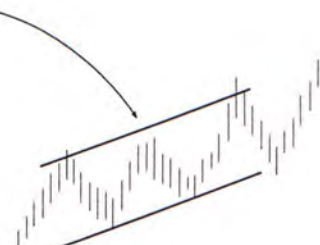
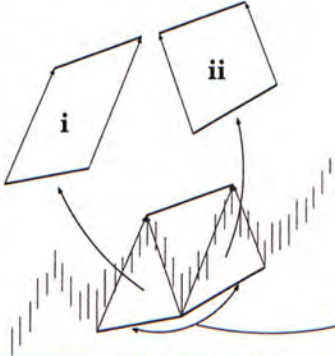

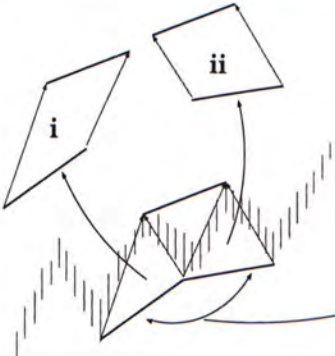

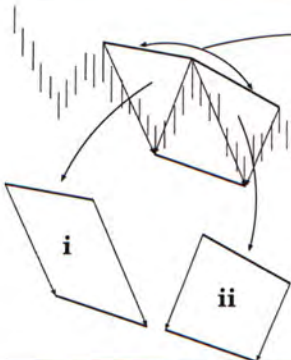
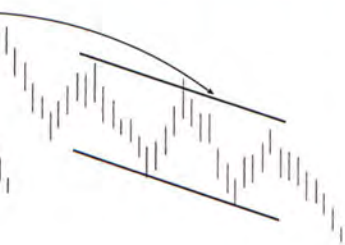
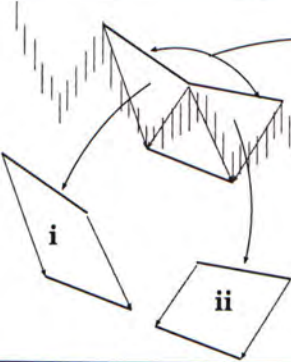
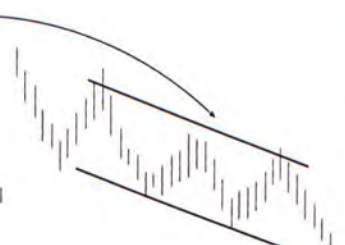


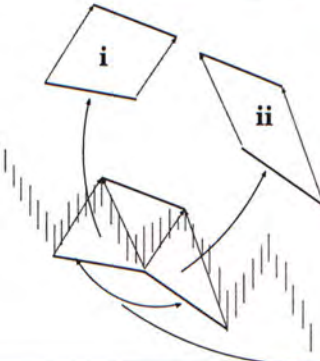
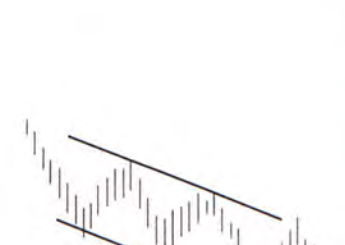
<i>Decomposed Trendline Pairs</i>	<i>Merged Pattern</i>	<i>Phase One Classification</i>
		<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Rising Wedge</i></p>
		<p>Trendline Pair: (i) <i>Rising Wedge</i> (ii) <i>Unclassified</i></p>
		<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Rising Wedge</i></p>
		<p>Trendline Pair: (i) <i>Rising Wedge</i> (ii) <i>Unclassified</i></p>

Table 4.8: Merging unclassified and unmerged trendline pairs which can form Price Channel (Slopes Down).

<i>Decomposed Trendline Pairs</i>	<i>Merged Pattern</i>	<i>Phase One Classification</i>
 <p>Diagram showing two trendline pairs, labeled 'i' and 'ii', extracted from a price chart. Pair 'i' is a downward-sloping line, and pair 'ii' is a downward-sloping wedge. Arrows indicate their relationship to the original price movement.</p>	 <p>Diagram showing the two trendline pairs merged into a single price channel with a downward slope.</p>	<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Falling Wedge</i></p>
 <p>Diagram showing two trendline pairs, labeled 'i' and 'ii', extracted from a price chart. Pair 'i' is a downward-sloping wedge, and pair 'ii' is a downward-sloping line. Arrows indicate their relationship to the original price movement.</p>	 <p>Diagram showing the two trendline pairs merged into a single price channel with a downward slope.</p>	<p>Trendline Pair: (i) <i>Falling Wedge</i> (ii) <i>Unclassified</i></p>
 <p>Diagram showing two trendline pairs, labeled 'i' and 'ii', extracted from a price chart. Pair 'i' is a downward-sloping line, and pair 'ii' is a downward-sloping wedge. Arrows indicate their relationship to the original price movement.</p>	 <p>Diagram showing the two trendline pairs merged into a single price channel with a downward slope.</p>	<p>Trendline Pair: (i) <i>Unclassified</i> (ii) <i>Falling Wedge</i></p>
 <p>Diagram showing two trendline pairs, labeled 'i' and 'ii', extracted from a price chart. Pair 'i' is a downward-sloping wedge, and pair 'ii' is a downward-sloping line. Arrows indicate their relationship to the original price movement.</p>	 <p>Diagram showing the two trendline pairs merged into a single price channel with a downward slope.</p>	<p>Trendline Pair: (i) <i>Falling Wedge</i> (ii) <i>Unclassified</i></p>

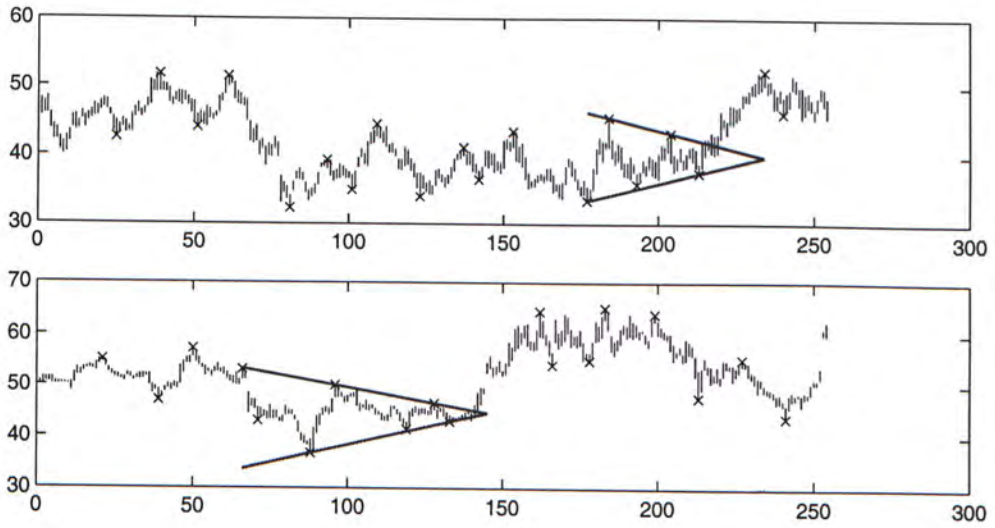


Figure 4.17: Examples of Symmetrical Triangle found in SP500. (Upper: Haliburton Co, Lower: Visteon Corp)

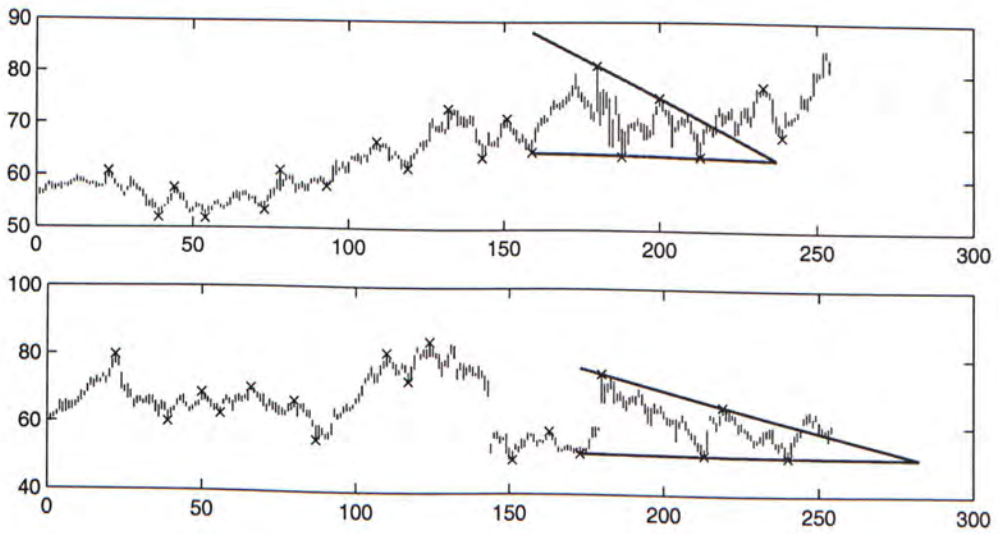


Figure 4.18: Examples of Descending Triangle found in SP500. (Upper: US West, Lower: Lucent Technology)

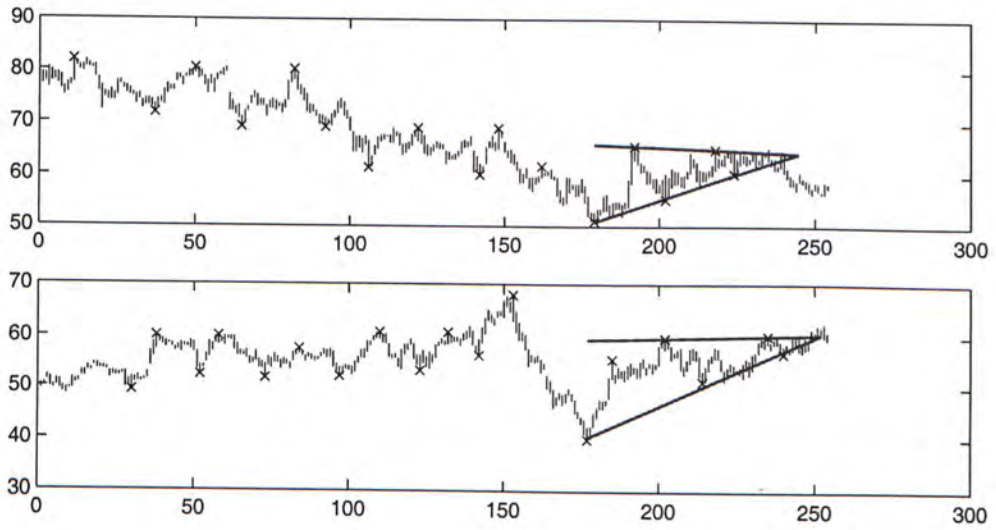


Figure 4.19: Examples of Ascending Triangle found in SP500. (Upper: Illinois Tool, Lower: Kerr Mcgee)

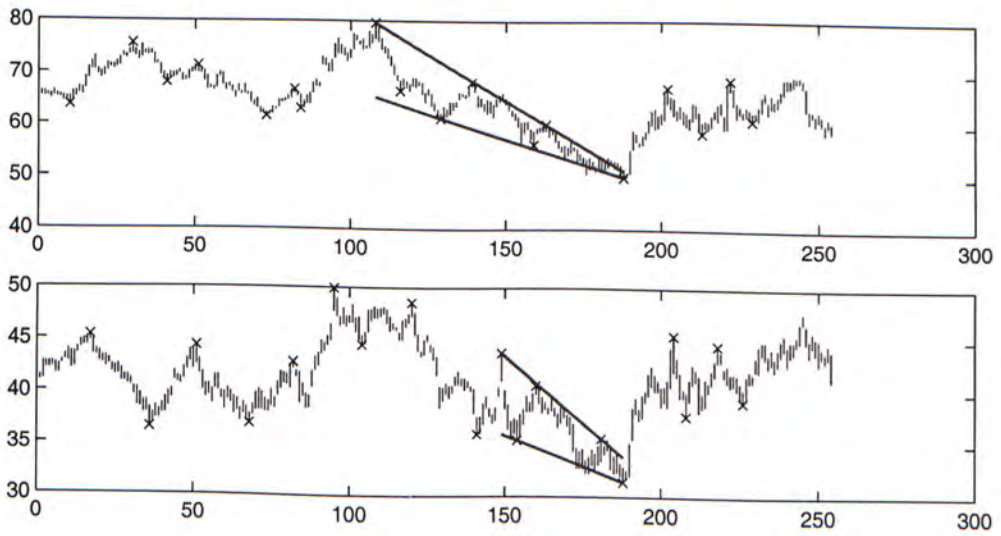


Figure 4.20: Examples of Falling Wedge found in SP500. (Upper: Jefferson Pilot, Lower: Wells Fargo)

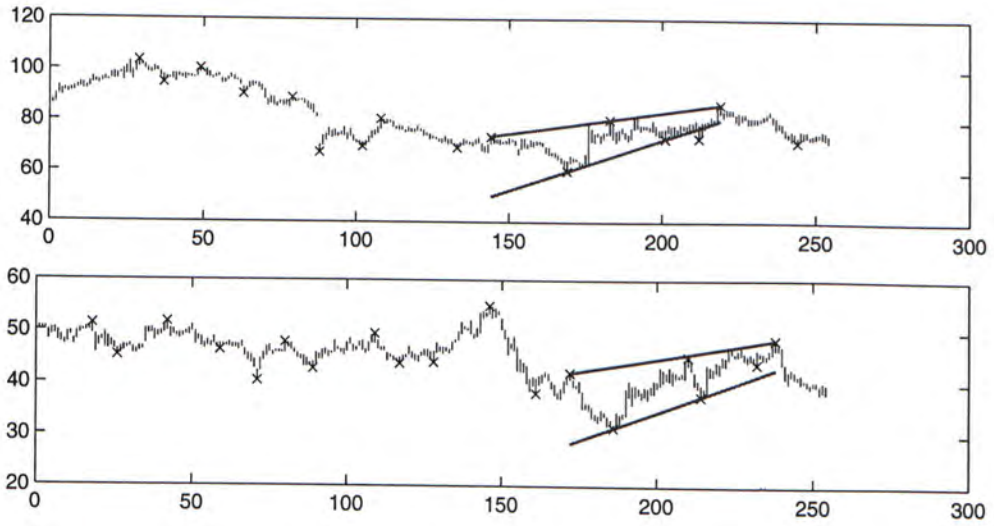


Figure 4.21: Examples of Rising Wedge found in SP500. (Upper: Eaton; Lower: Praxair, Inc.)

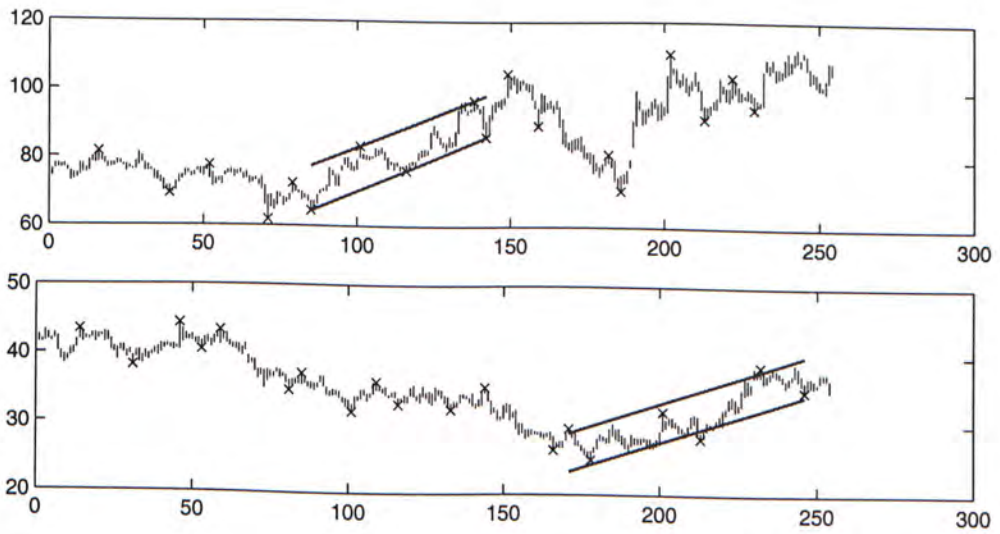


Figure 4.22: Examples of Price Channel found in SP500. (Upper: Marsh & McLennan; Lower: Unocal Corp.)

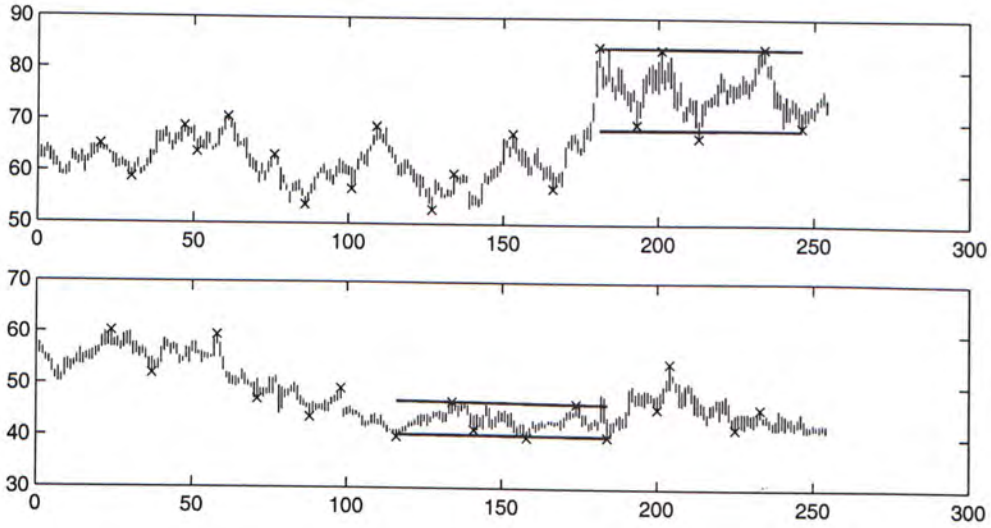


Figure 4.23: Examples of Rectangle found in SP500. (Upper: Schlumberger Ltd.; Lower: PACCAR Inc.)

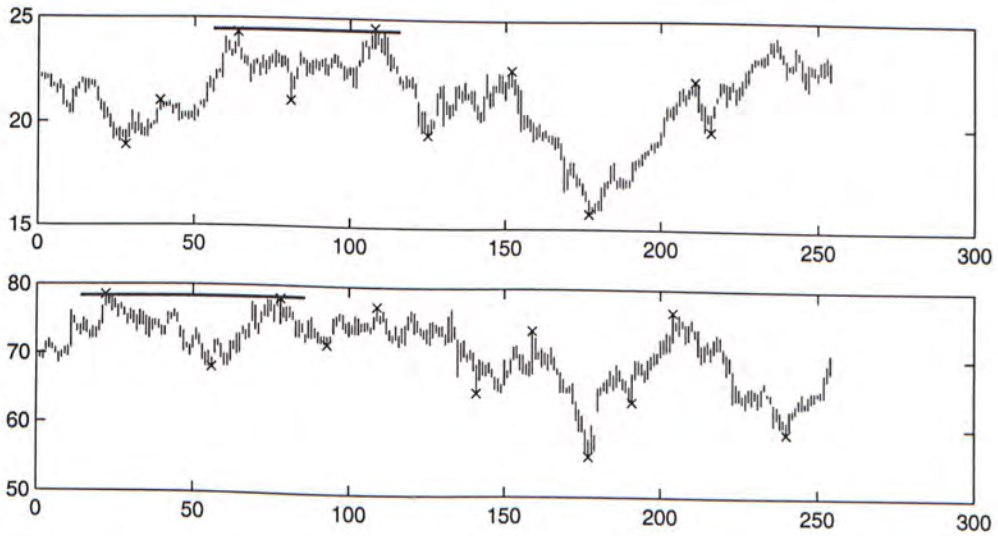


Figure 4.24: Examples of Double Top found in SP500. (Upper: Occidental Petroleum; Lower: GTE)

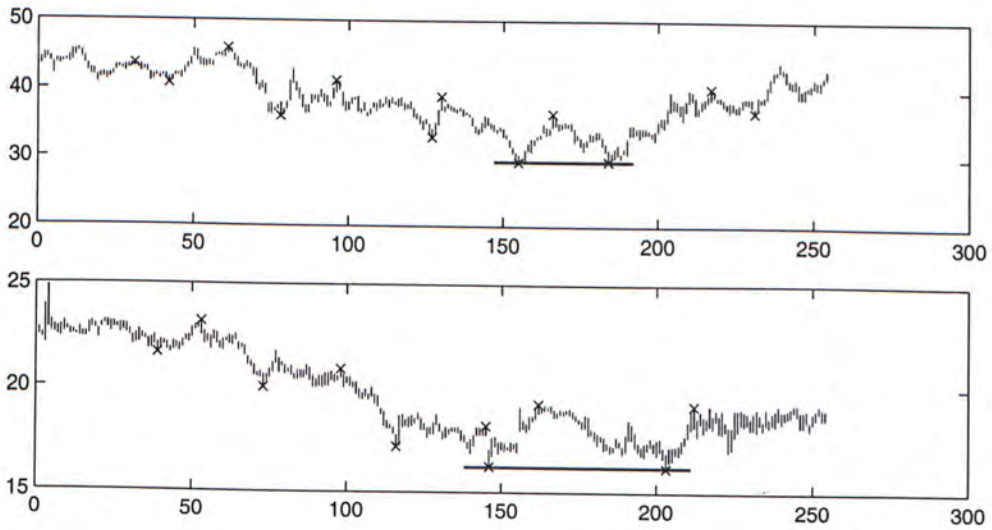


Figure 4.25: Examples of Double Bottom found in SP500. (Upper: Abbott Labs; Lower: Sempra Energy)

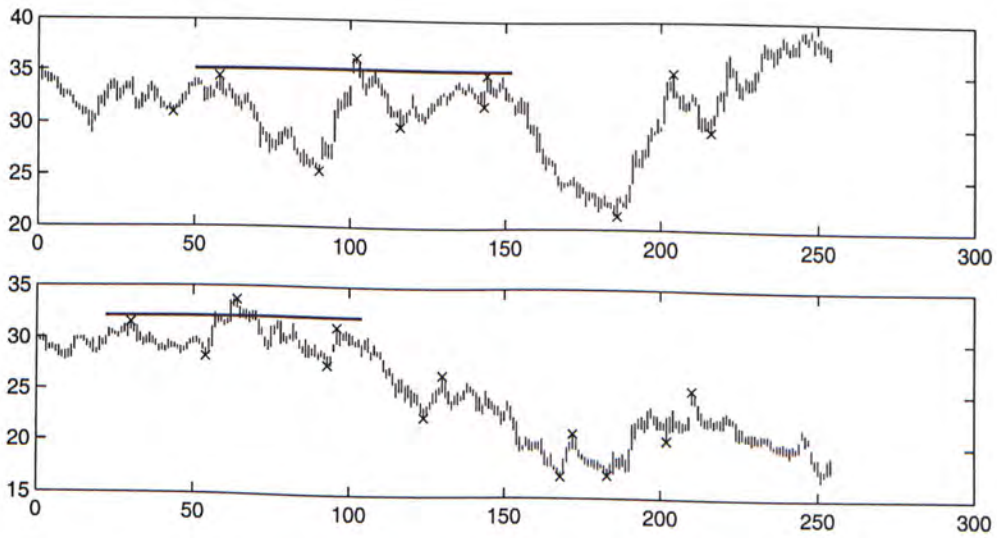


Figure 4.26: Examples of Triple Top found in SP500. (Upper: St Paul.; Lower: Masco Corp.)

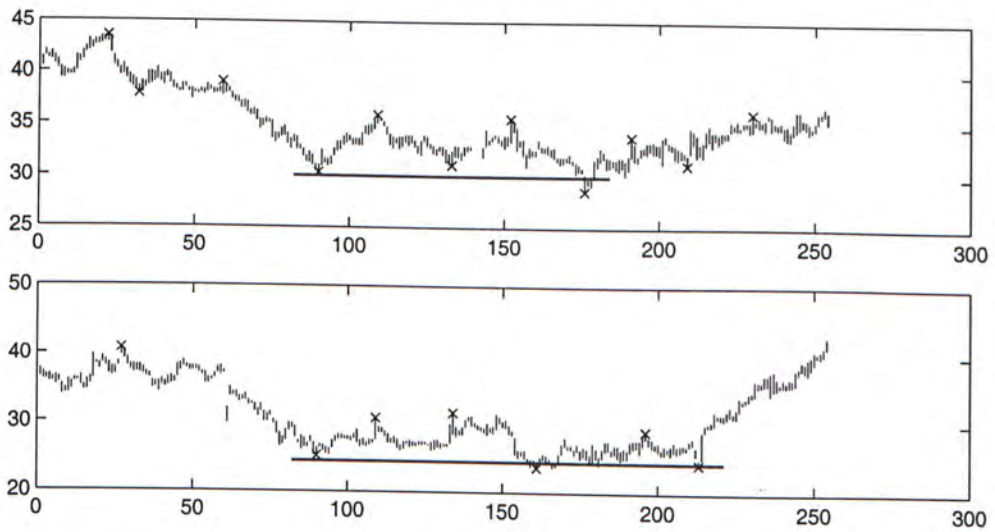


Figure 4.27: Examples of Triple Bottom found in SP500. (Upper: Ashland; Lower: St Jude Medical)

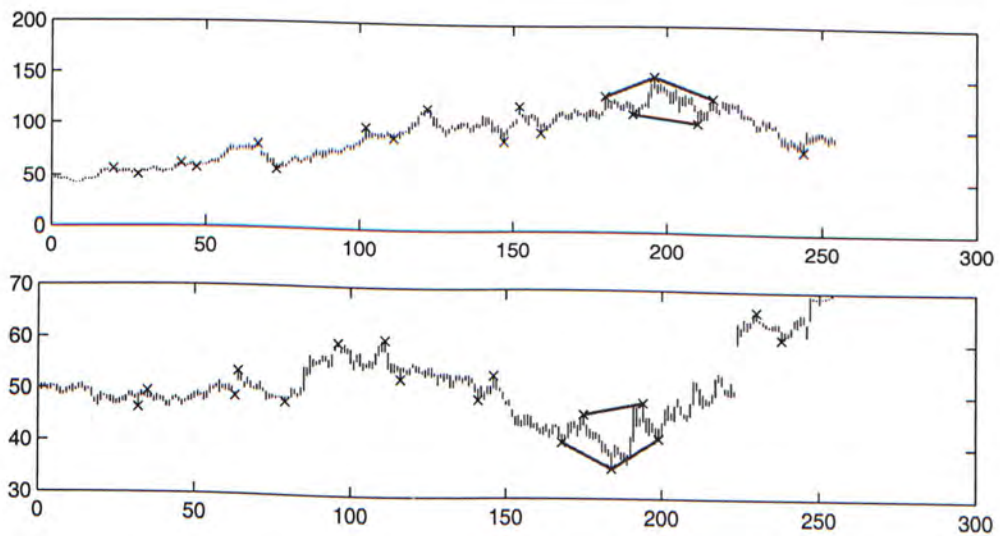


Figure 4.28: Examples of Head and Shoulders found in SP500. (Upper: Apply Computer; Lower: Bestfoods)

Chapter 5

Conclusion

Time series analysis is of growing importance as the observed patterns in historical time series data are useful for predicting future trends. Since time series data are usually large in size, people are seeking efficient methods for searching in time series databases. As sequence data usually contain observable trends and random noises, one ideal approach is to remove the noises and perform searching on approximated sequence data. To reduce noise while retaining the general shape in a time series is hence a challenge for developing an efficient time series pattern matching system.

Different patterns formed by time series data may be considered similar after appropriate scaling and shifting in time and amplitude domains. We should avoid applying brute force checking on different scaling and shifting factors to vast amount of data as performance will be degraded. Therefore, a lattice structure which encompasses layers of control points has been proposed to capture different resolutions of features in a time series. An algorithm to build a lattice structure and a method for similarity matching have also been proposed. An experiment on finding a specific pattern on historical stock data was conducted. The results

show that our algorithm can apply appropriate scaling factors and translation offsets to locate similar patterns.

To further improve the pattern matching performance, we have integrated the lattice structure with the indexing scheme proposed in [CW99]. The time domain values are first temporarily set aside while the amplitude domain values are indexed with an R -tree. This may result in false alarms but experimental results show that the indexing scheme still outperforms sequential search.

While the technique of control point identification can handle pattern matching in uni-value time series, it was successfully extended to find chart patterns which are bi-value time series. Specifically, we have focused on finding continuation and reversal patterns which are formed over a certain period of time. The formations of these patterns are arbitrary, so one could not find a query template for any of these patterns. Therefore, traditional methods of time series pattern matching cannot be applied. We observed that those chart patterns are supported by significant turning points. In other words, those significant turning points form trendlines which contribute to the pattern formation. Therefore, as what we have done in uni-value time series, we spot out remarkable control points through a ranking process. Trendlines are then continuously defined along the time line by always selecting the most representational control point within a user-defined period. As there is no restriction on the length of the chart patterns, we propose to slice the trendlines into pairs and use a bottom-up approach to identify patterns through progressive merging and rejection. An experiment was carried out using 500 different stock charts and thirteen types of chart patterns are successfully located.

Finally, we conclude that using control points can minimize the overhead of processing a massive time series while preserving its original features. In this dissertation, we have only considered local maxima and minima as control points

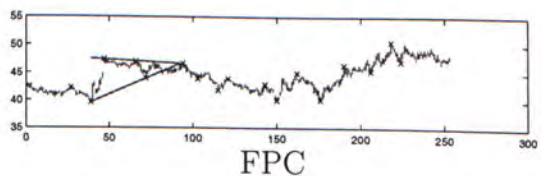
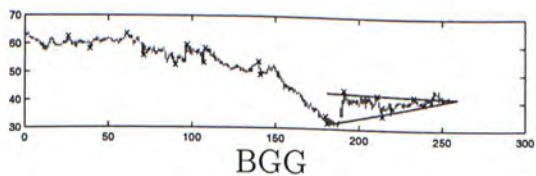
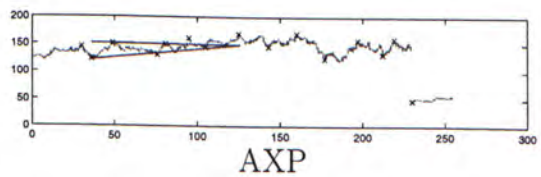
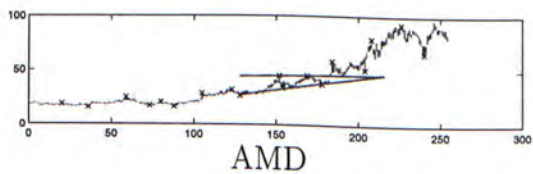
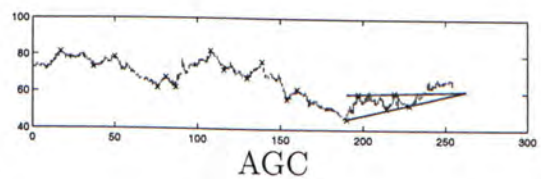
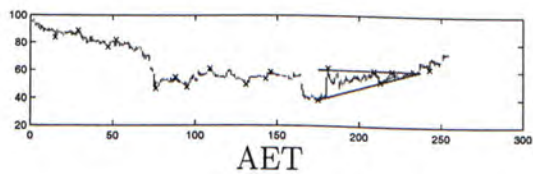
which are more suitable for volatile time series like stock data. For slowly changing time series in which local maxima and minima are far apart from each other, inflection points or other points whose higher order derivatives are zero may be suitable candidates of control points.

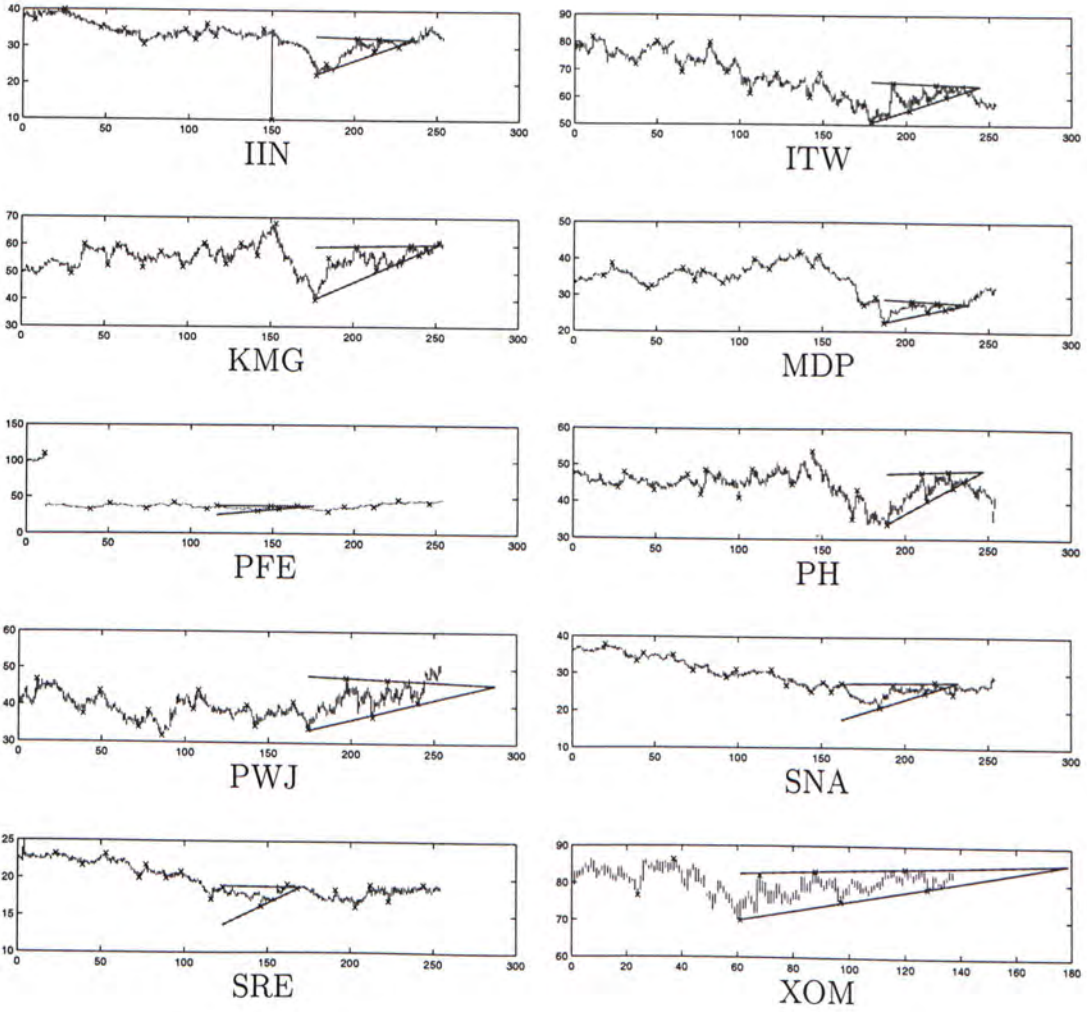
Appendix A

Supplementary Results

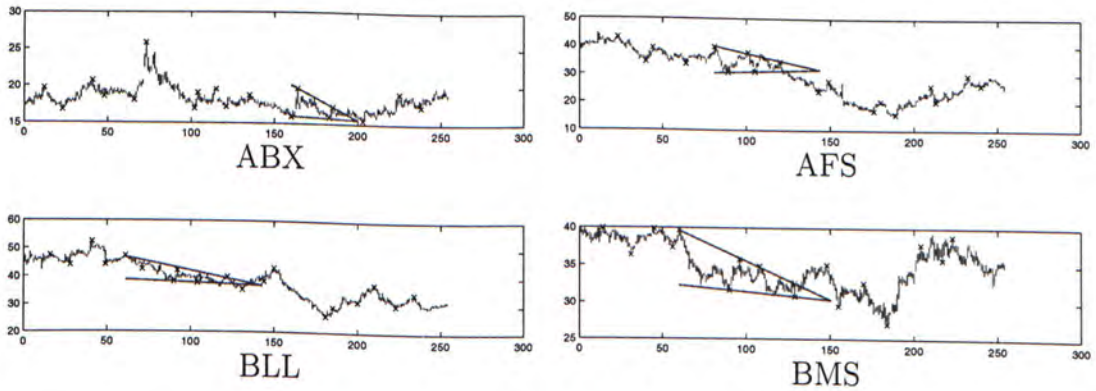
In this appendix, we show a full set of the chart patterns found in stock charts of S&P 500 Stocks (daily high and daily low price from 6/16/1999 to 12/14/1999) using the algorithm proposed in Chapter 4.

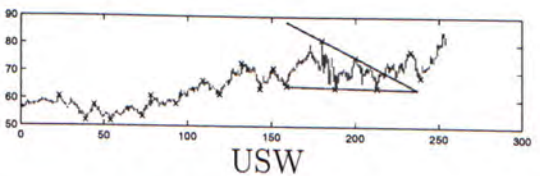
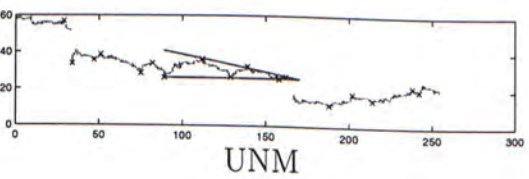
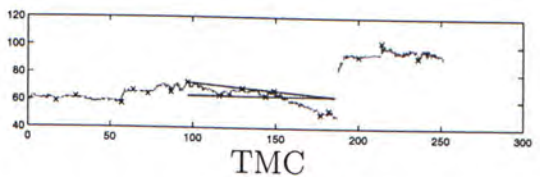
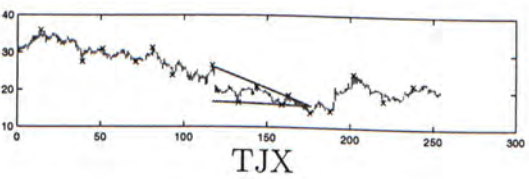
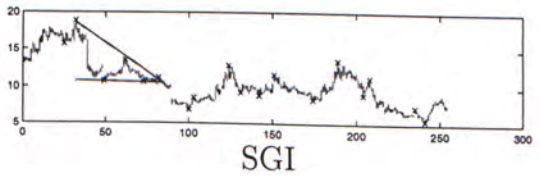
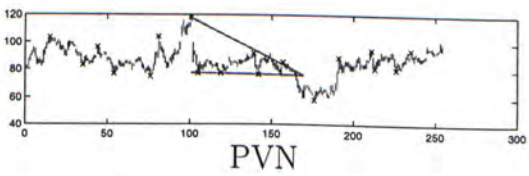
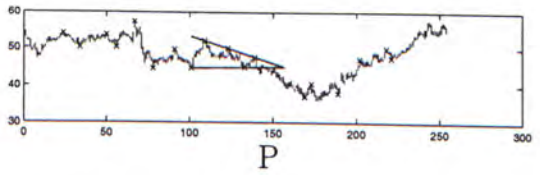
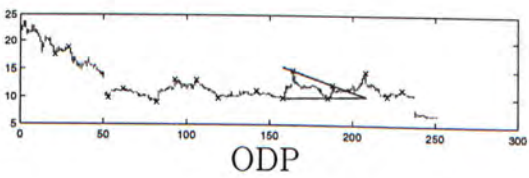
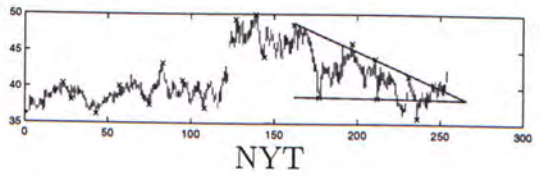
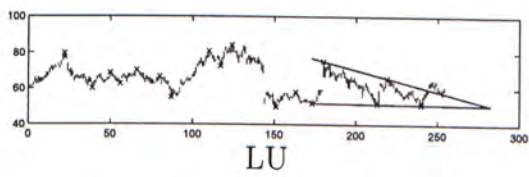
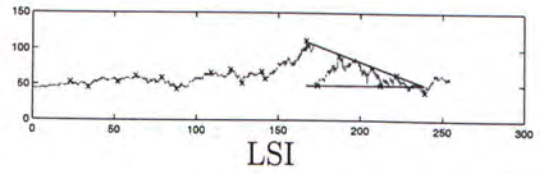
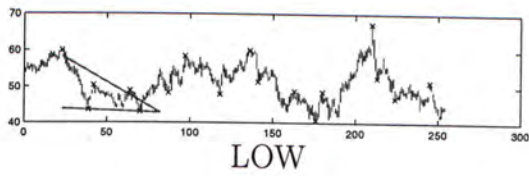
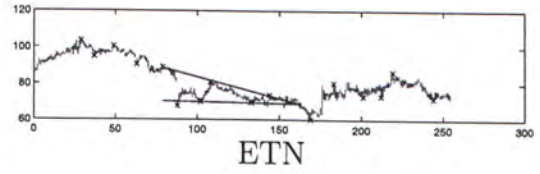
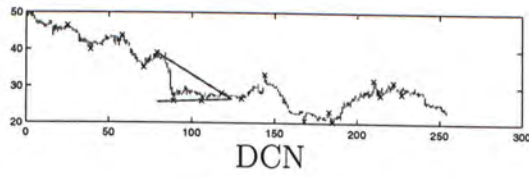
A.1 Ascending Triangle

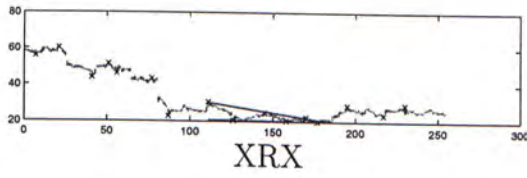




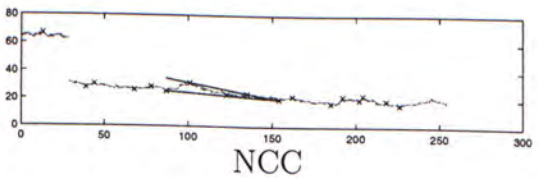
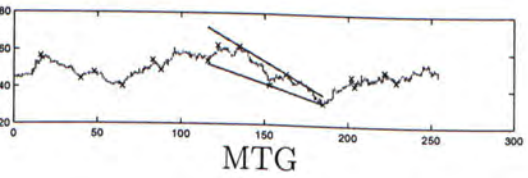
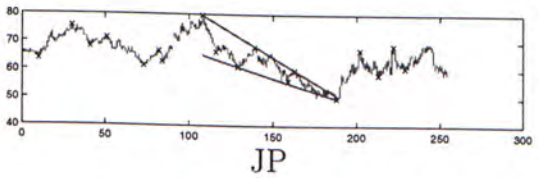
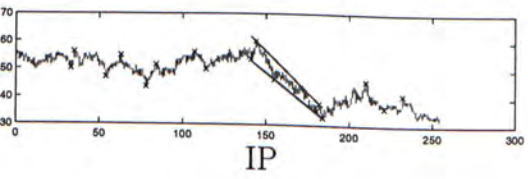
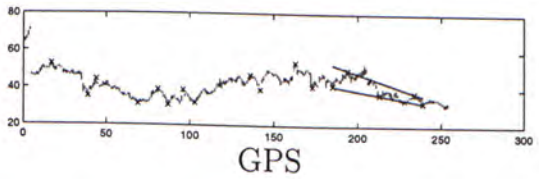
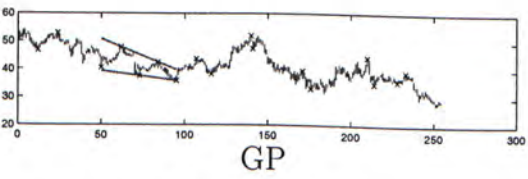
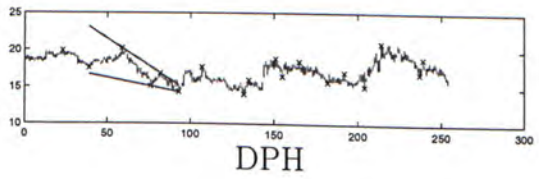
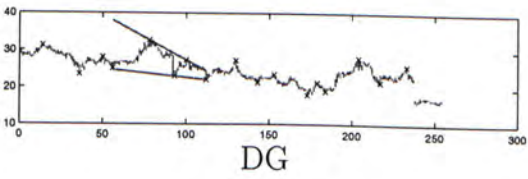
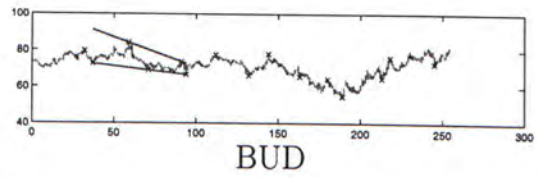
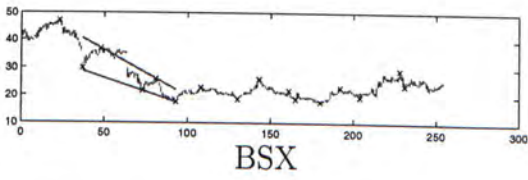
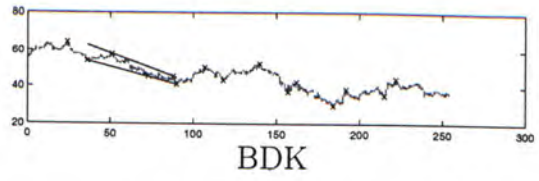
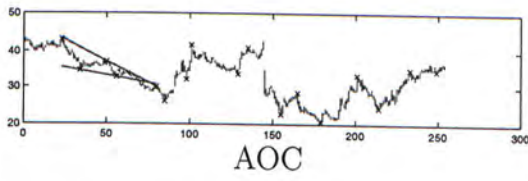
A.2 Descending Triangle

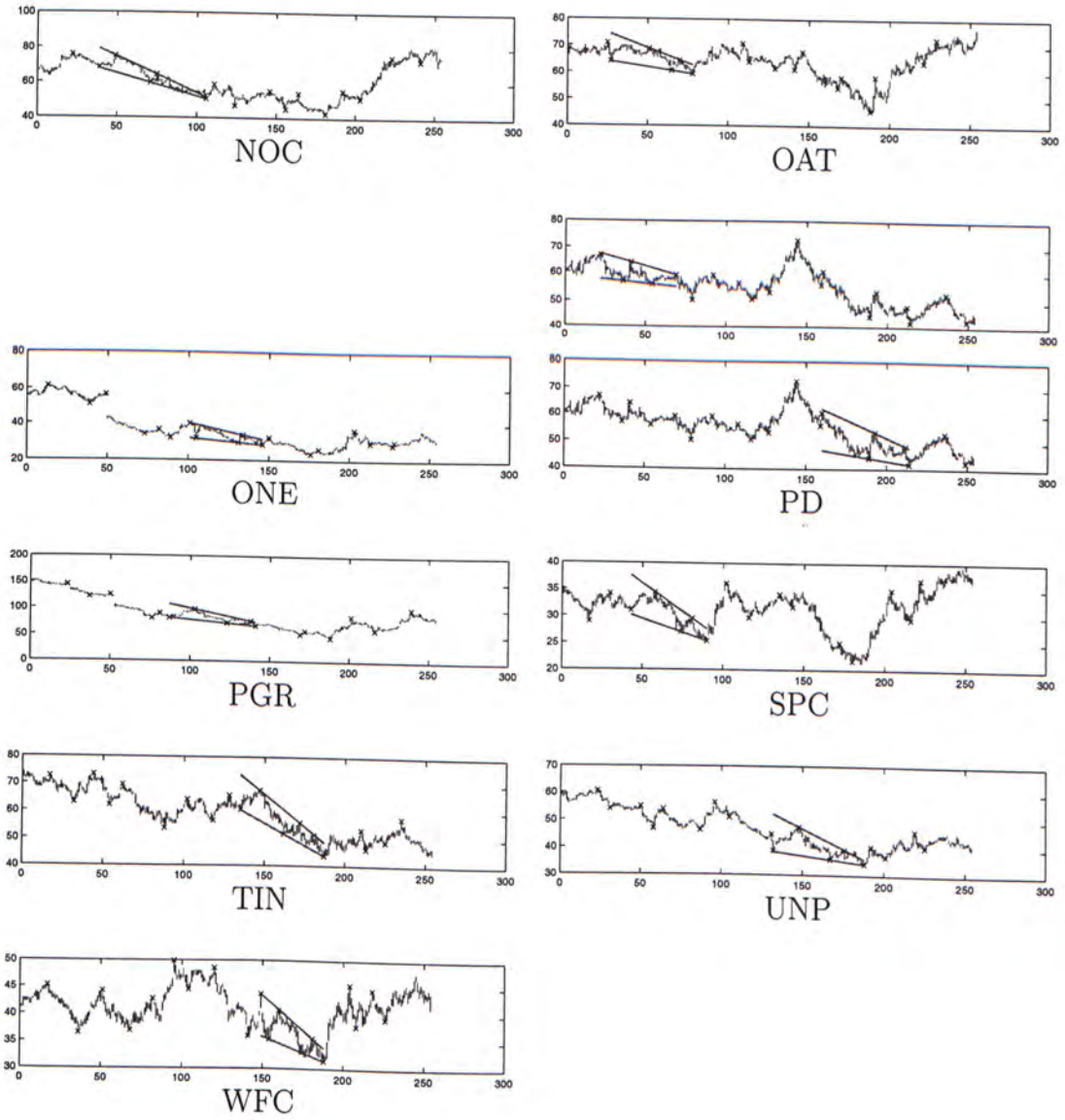




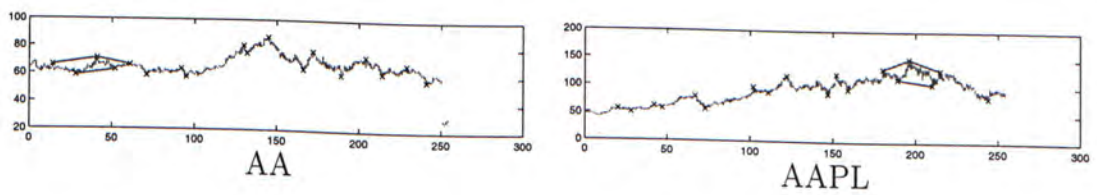


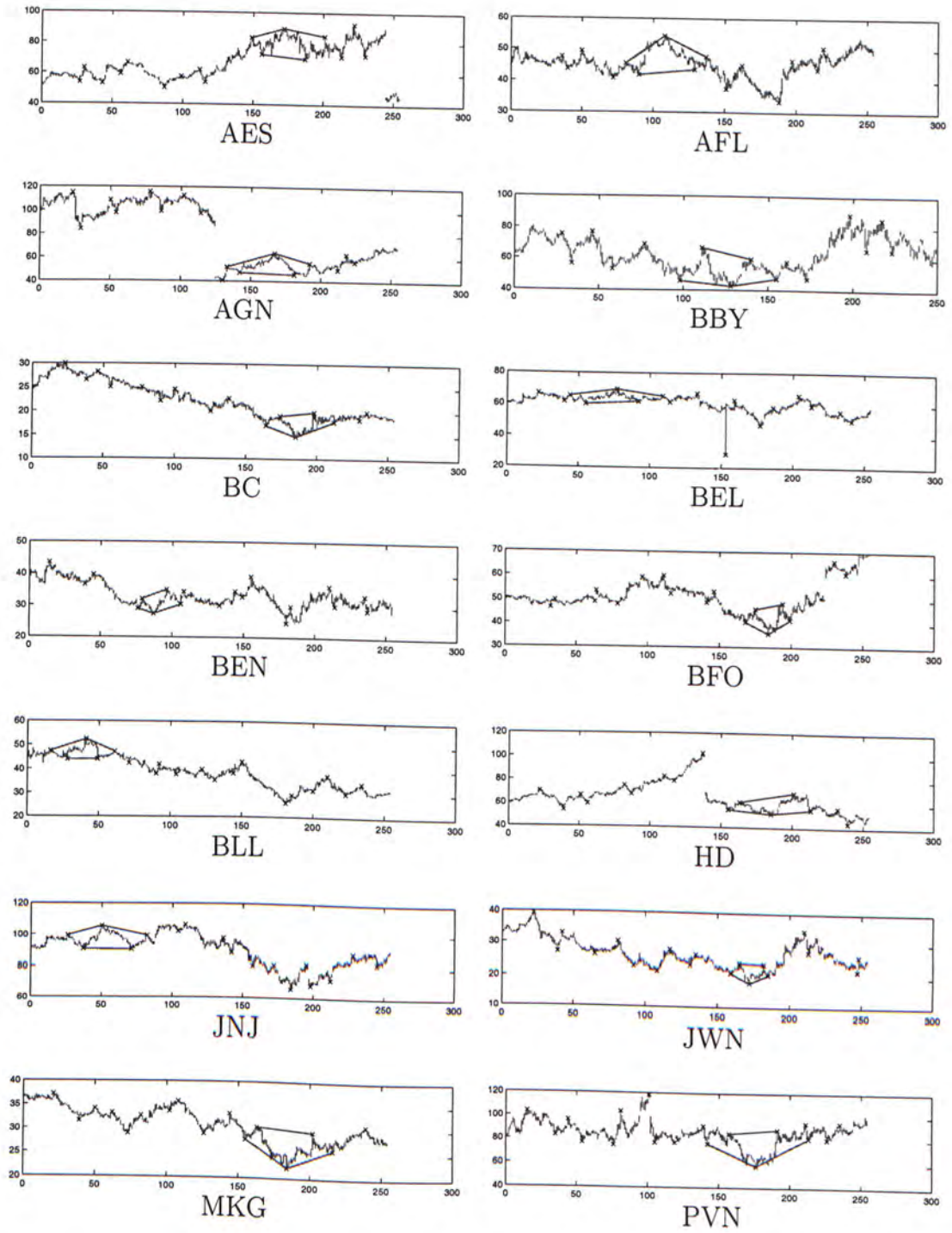
A.3 Falling Wedge

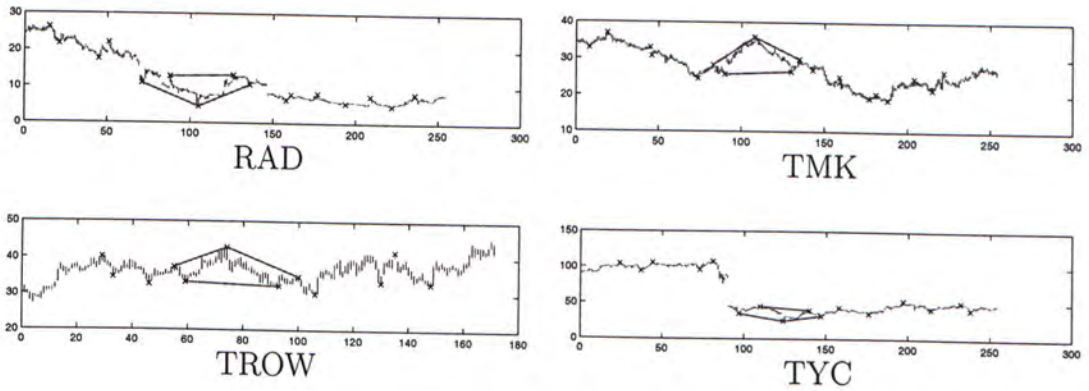




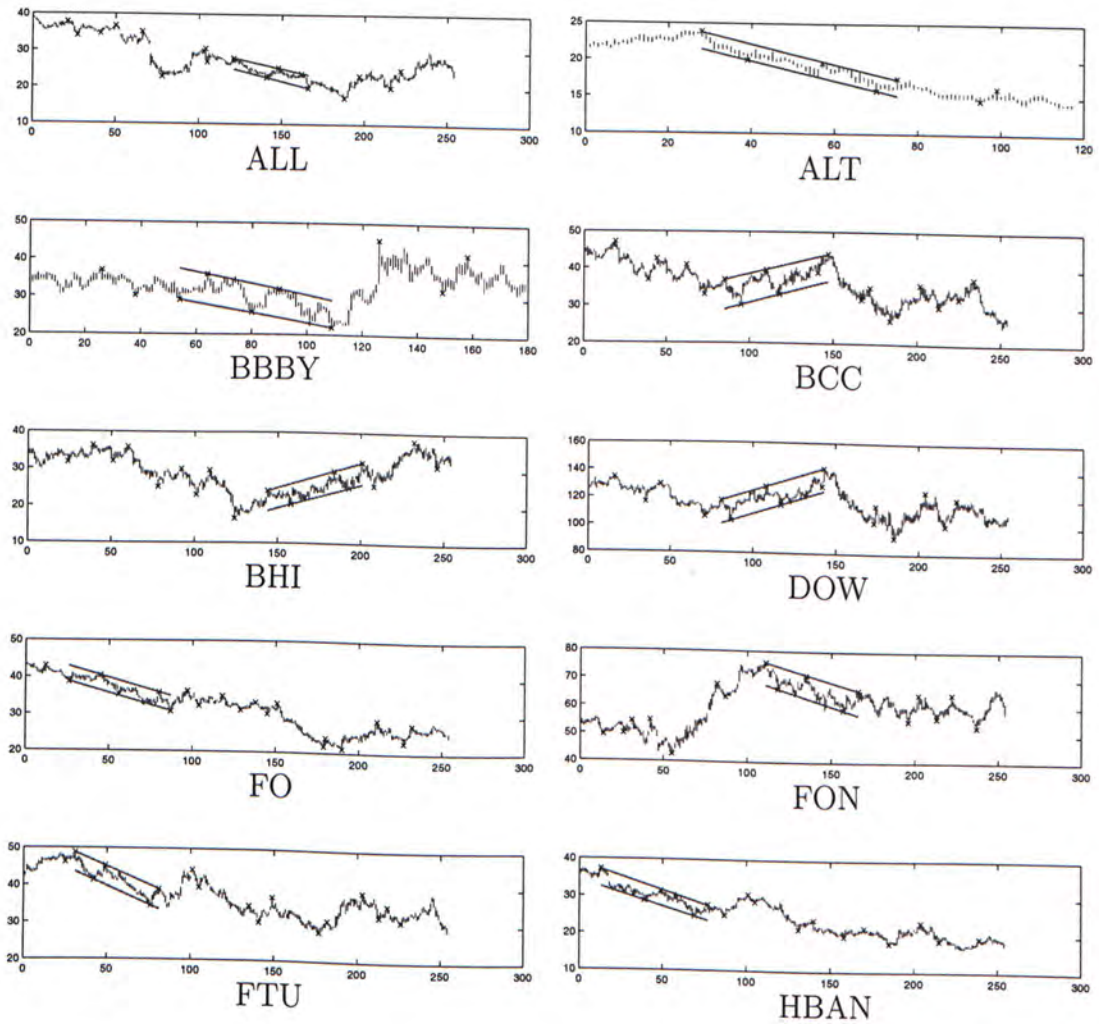
A.4 Head and Shoulders

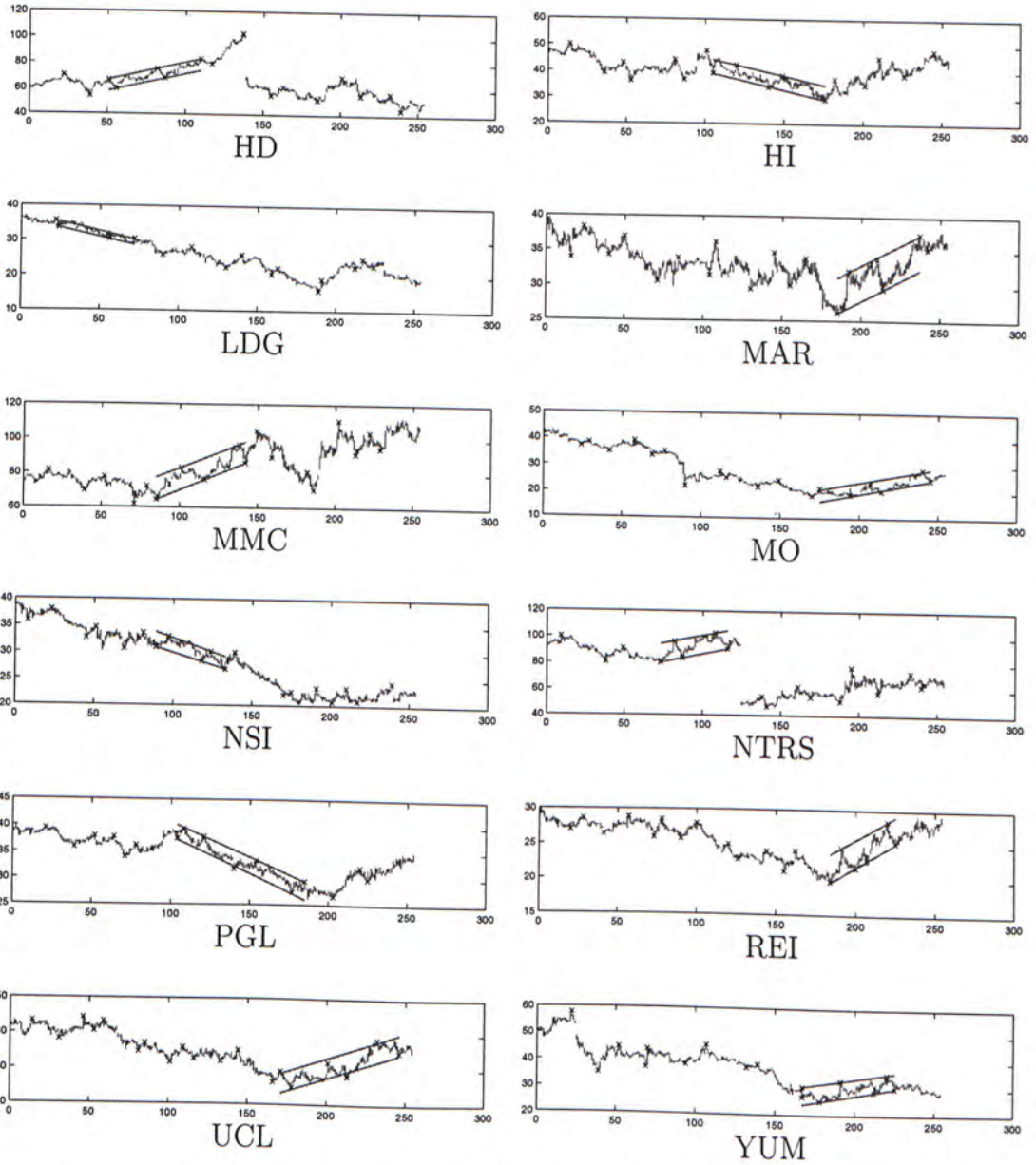




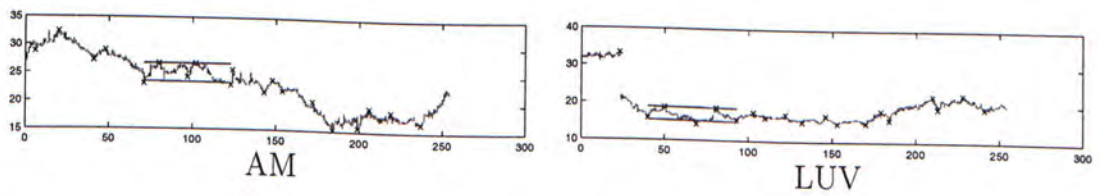


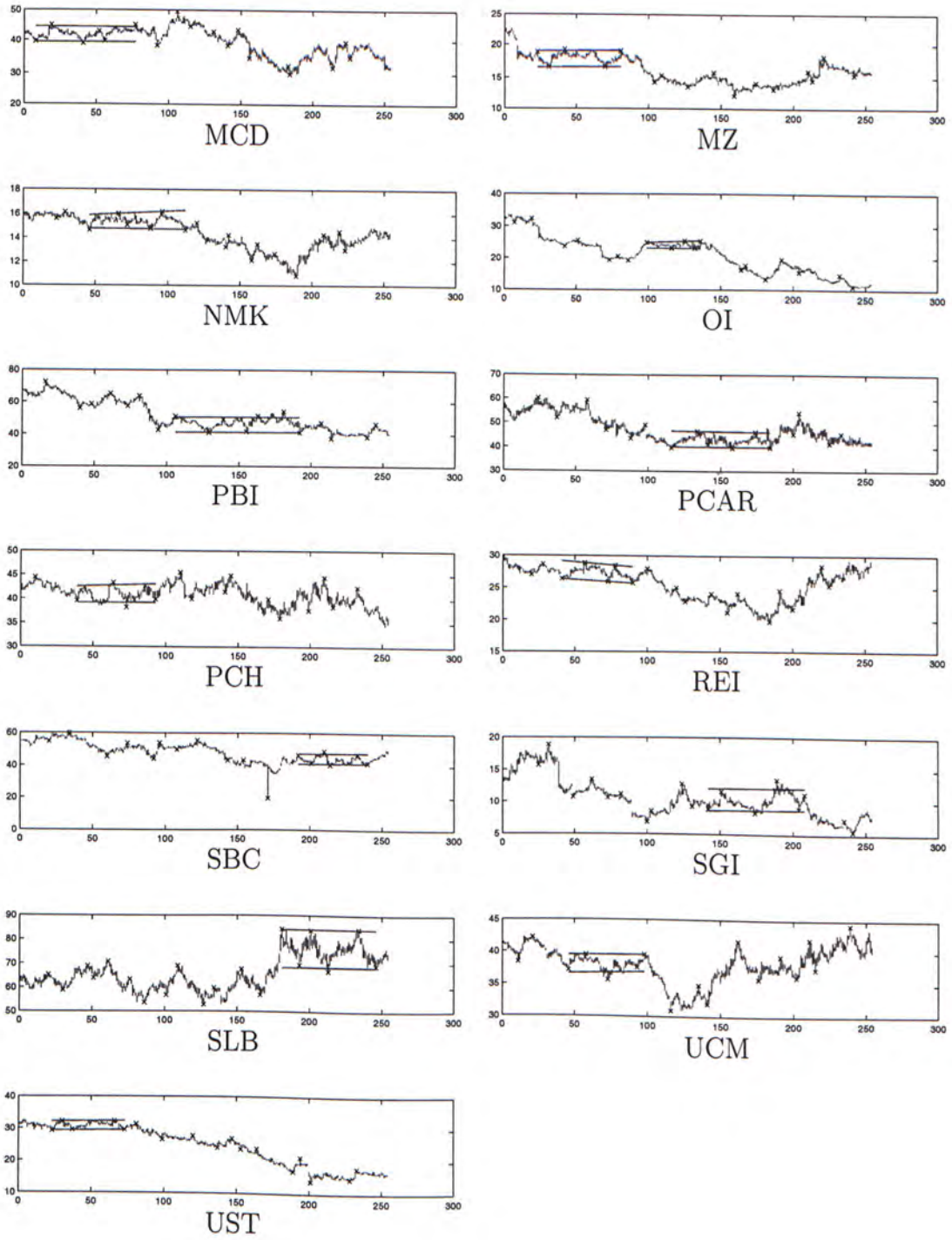
A.5 Price Channel



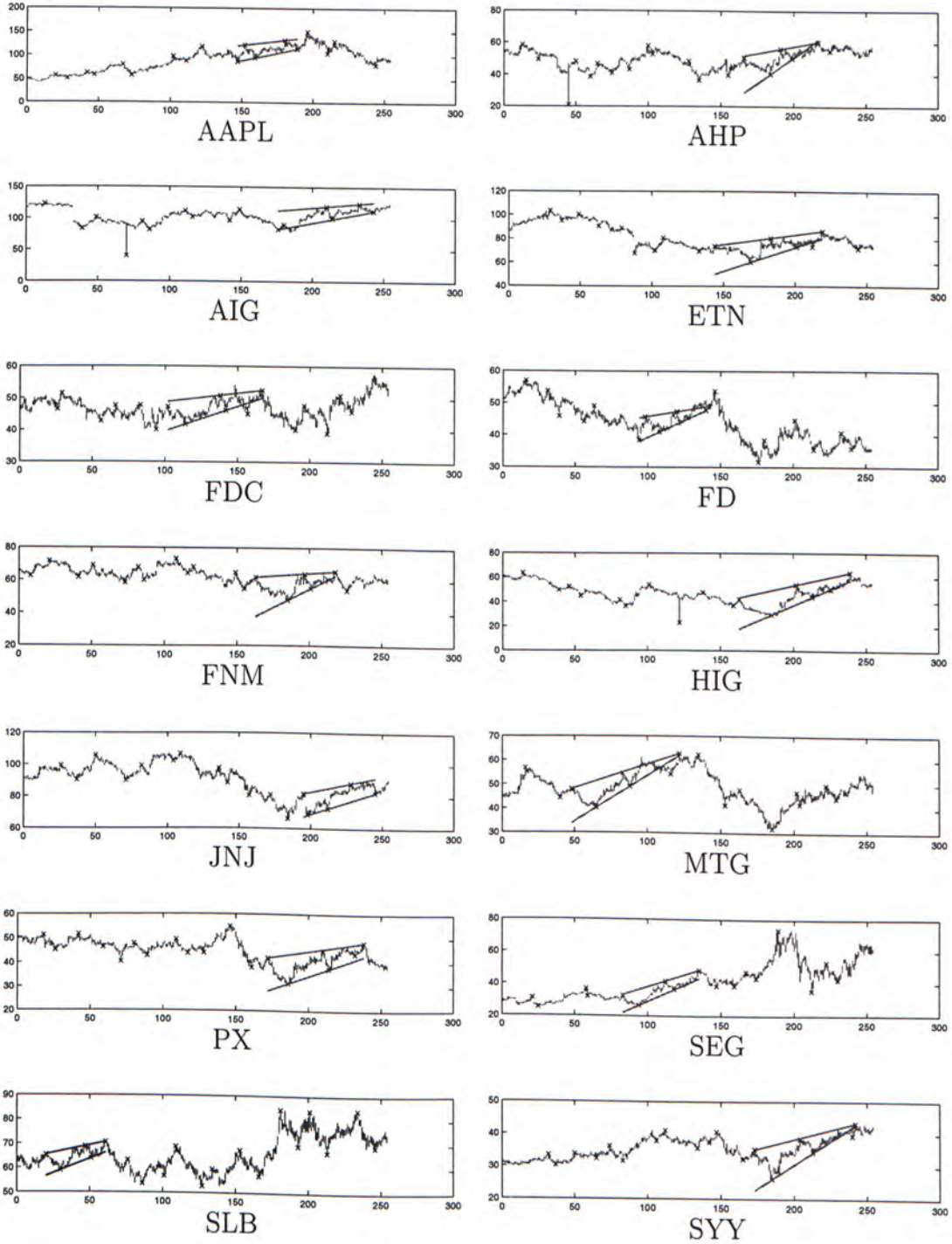


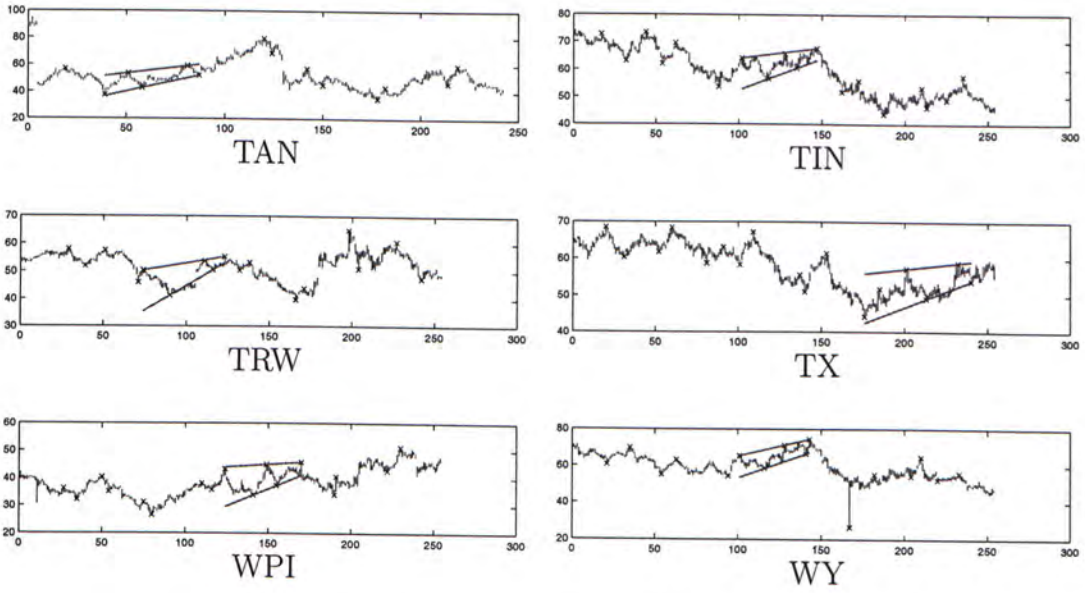
A.6 Rectangle



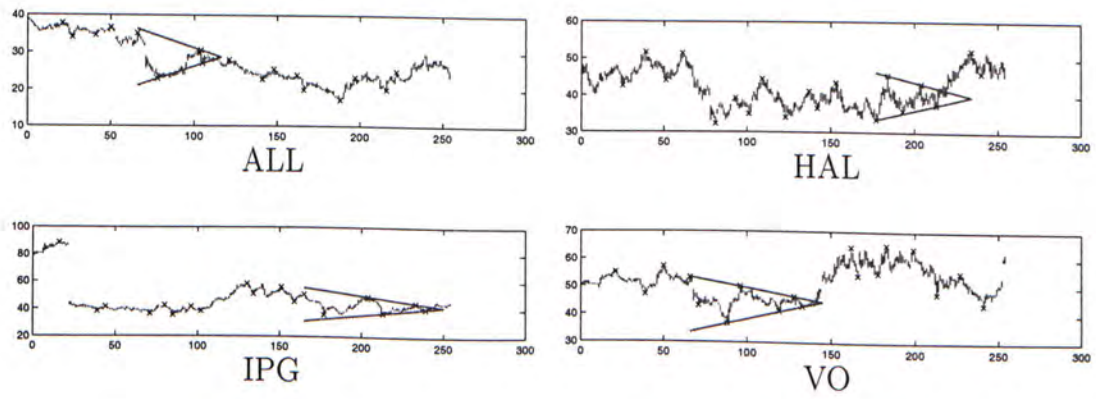


A.7 Rising Wedge

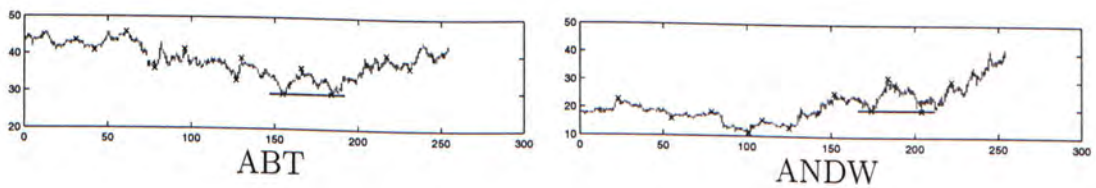


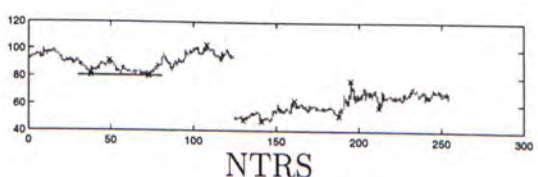
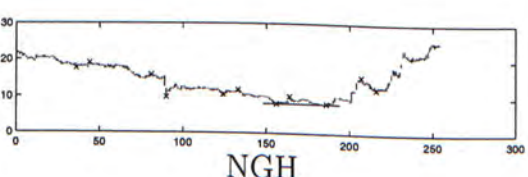
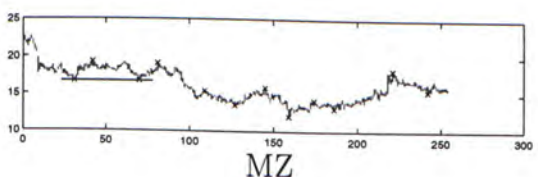
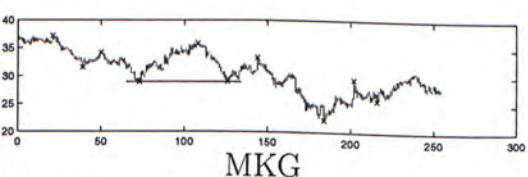
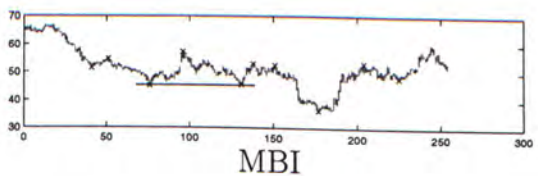
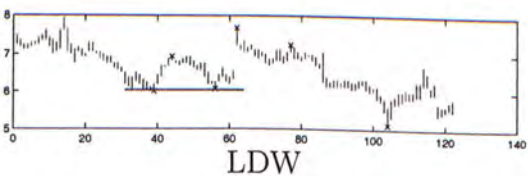
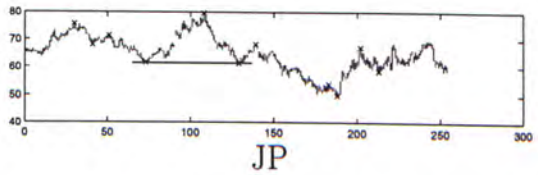
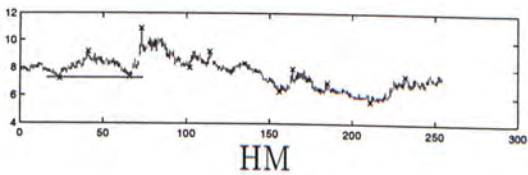
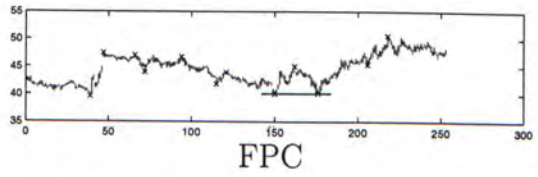
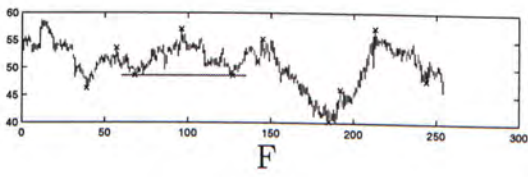
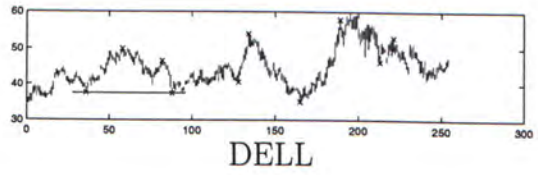
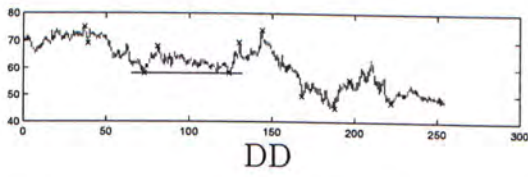
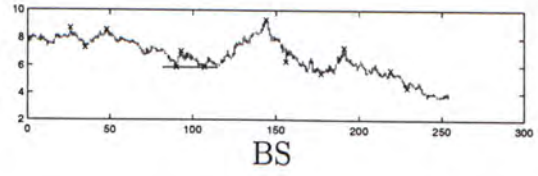
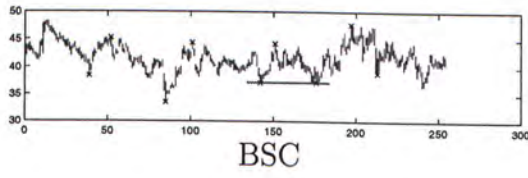


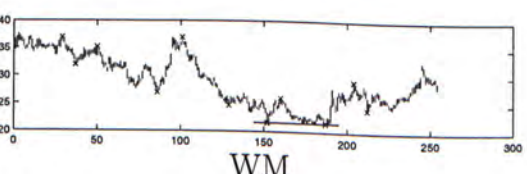
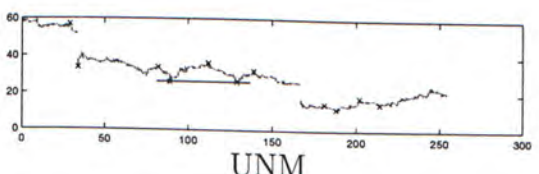
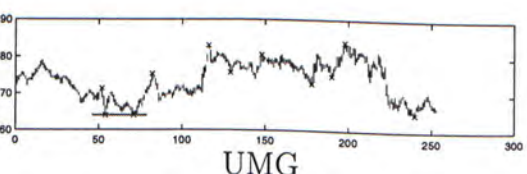
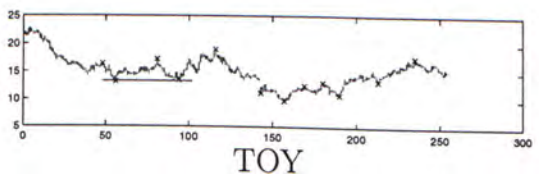
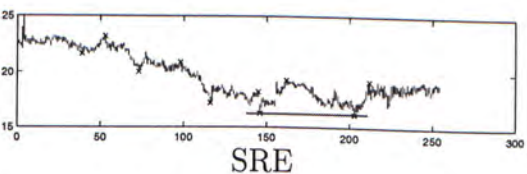
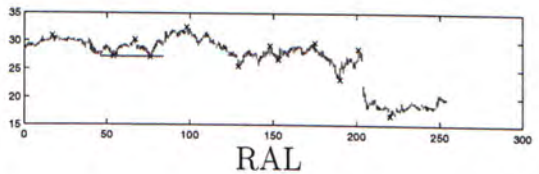
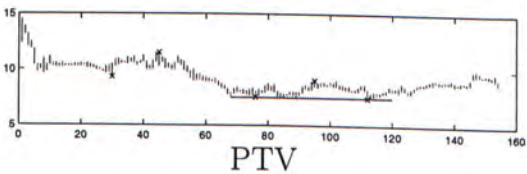
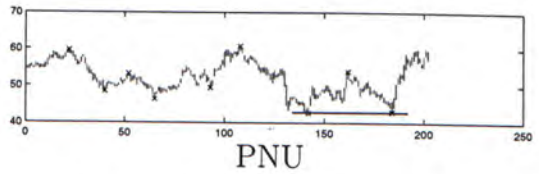
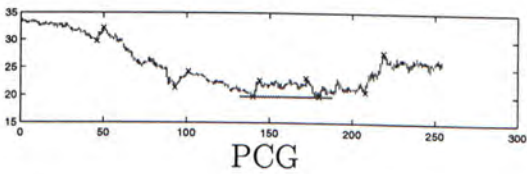
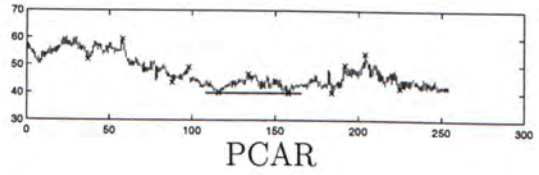
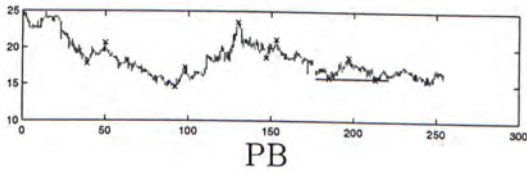
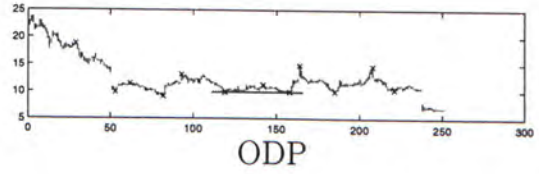
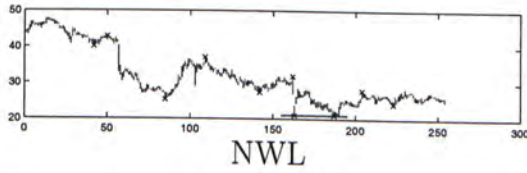
A.8 Symmetric Triangle



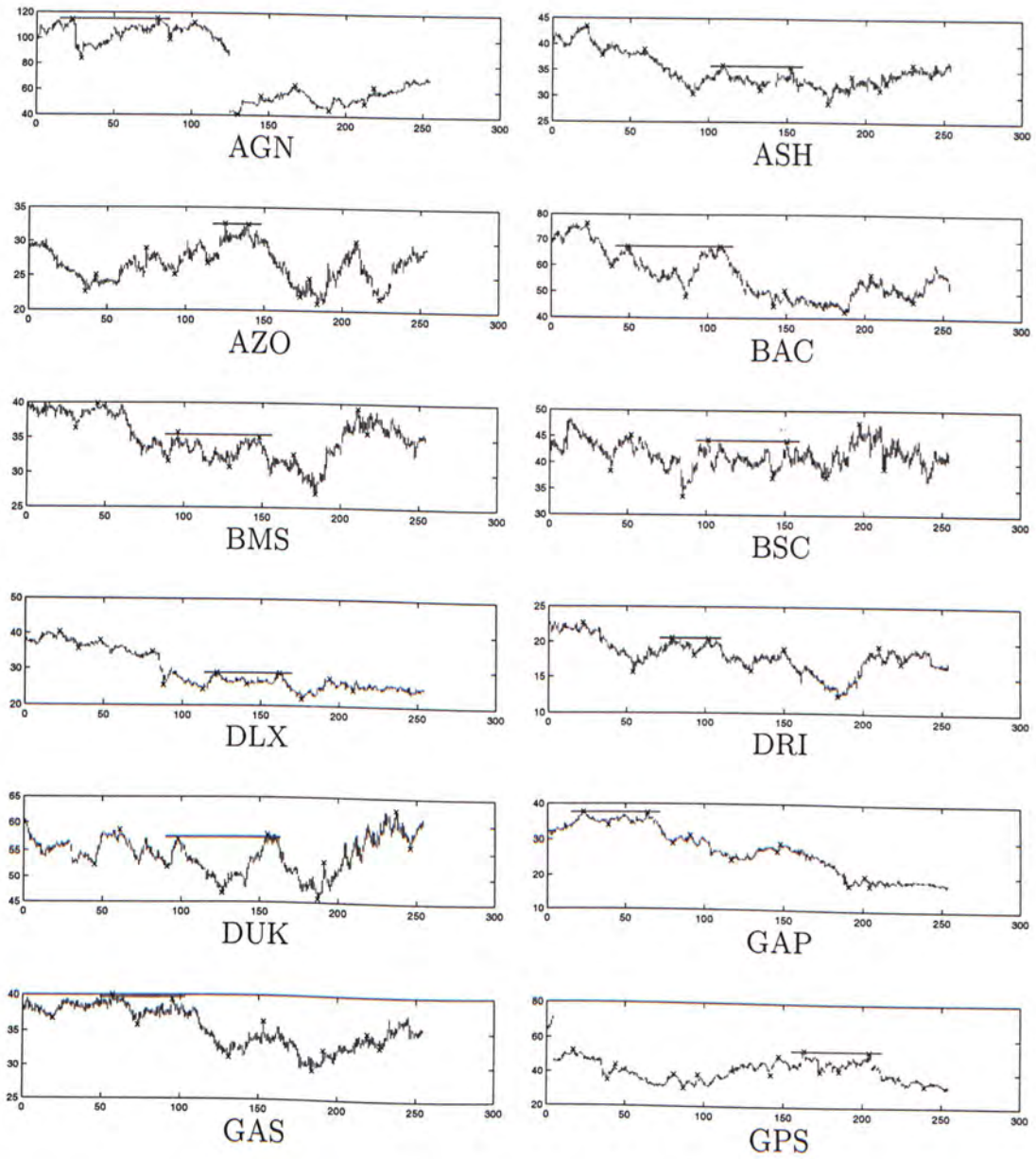
A.9 Double Bottom

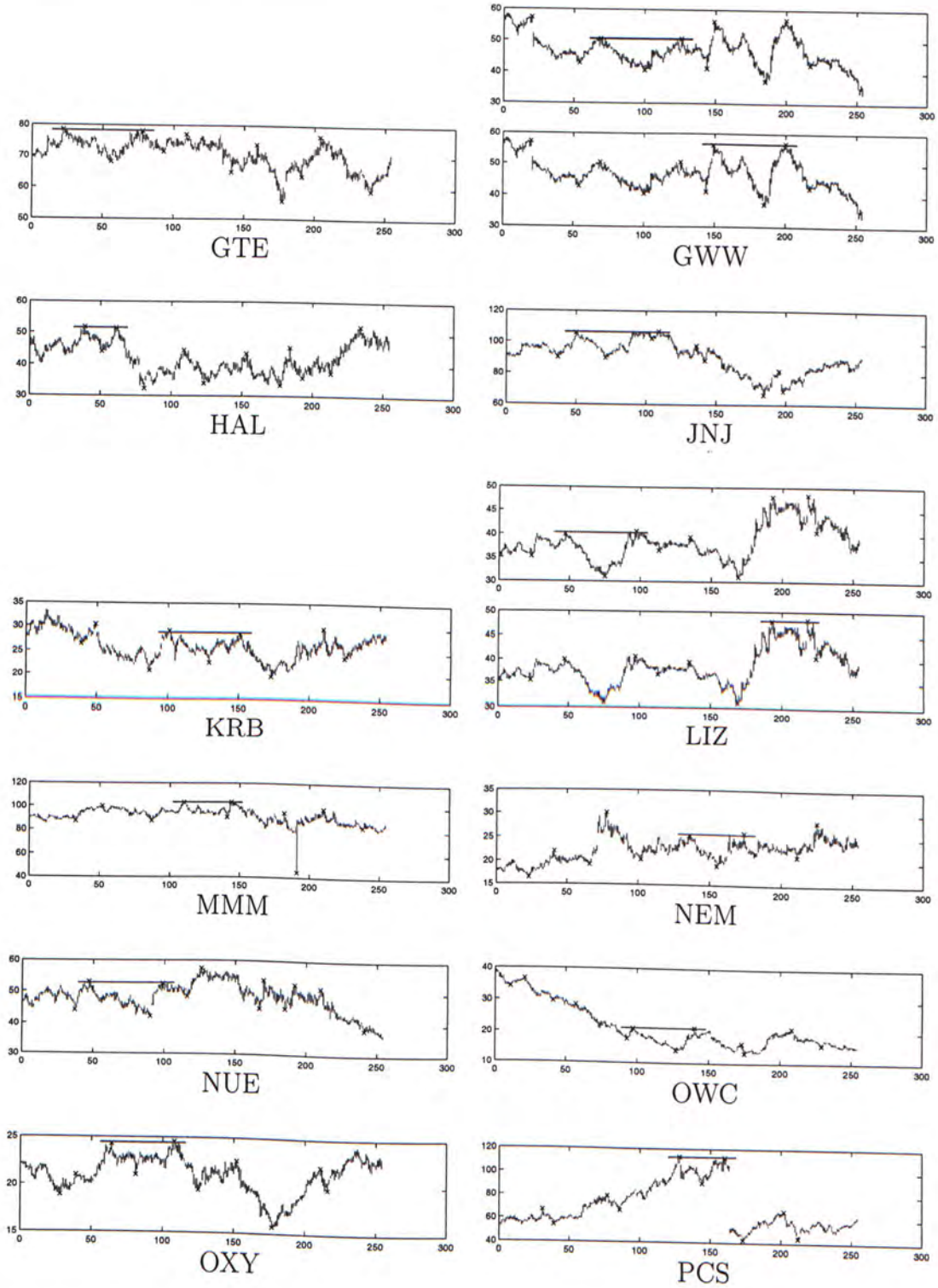


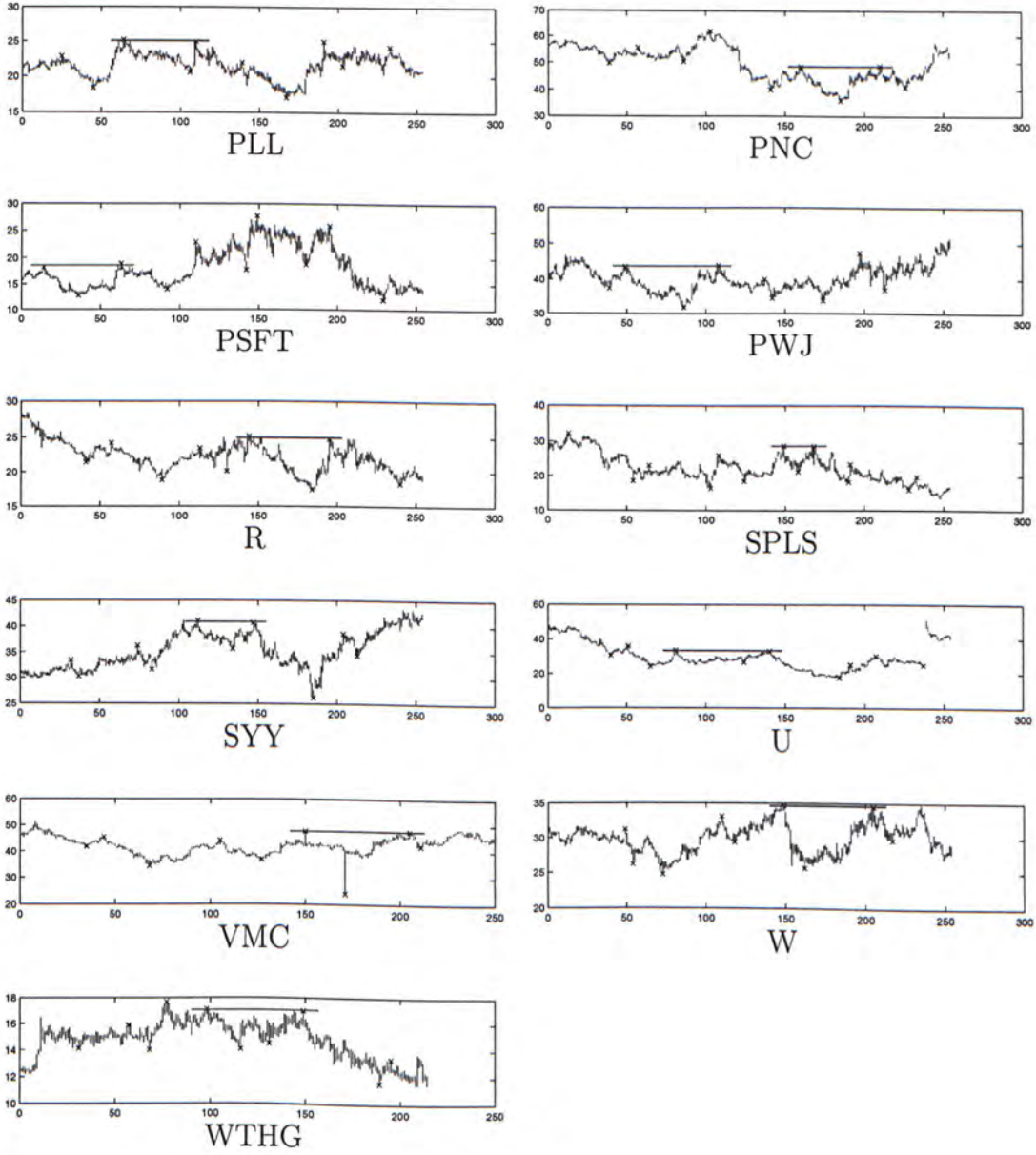




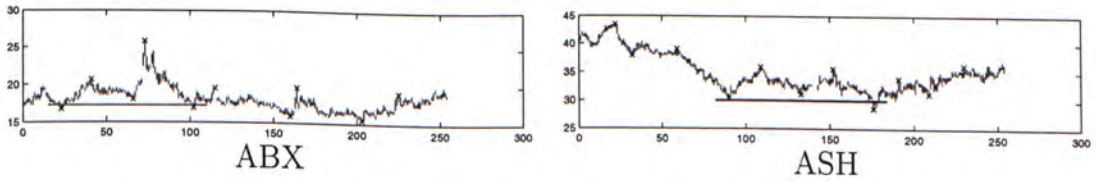
A.10 Double Top

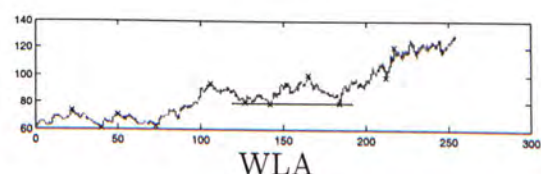
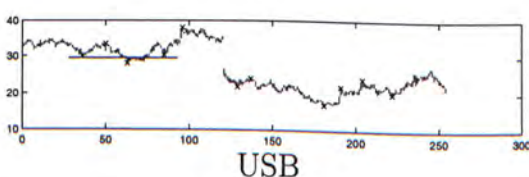
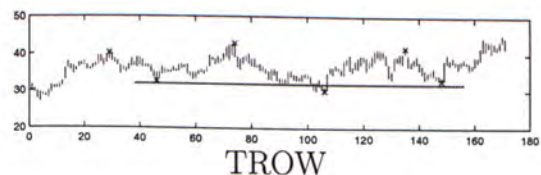
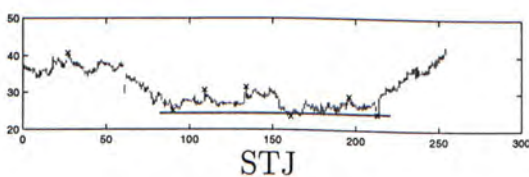
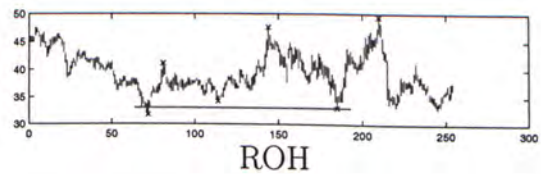
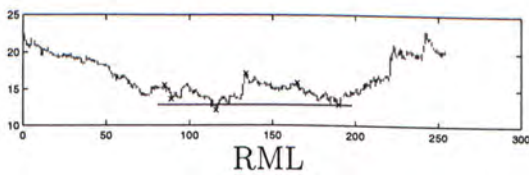
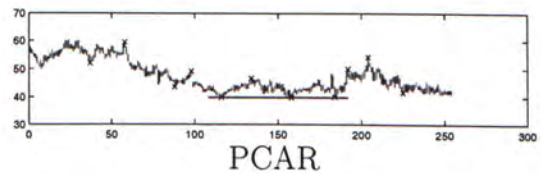
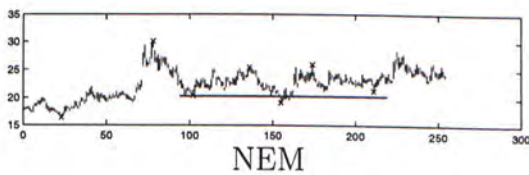
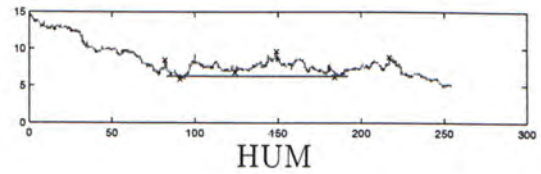
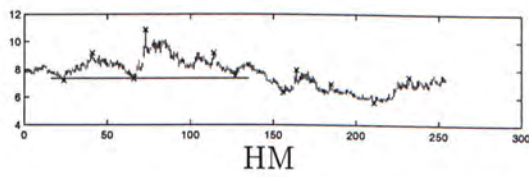
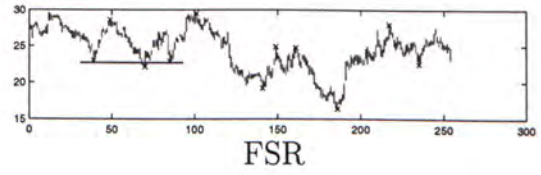
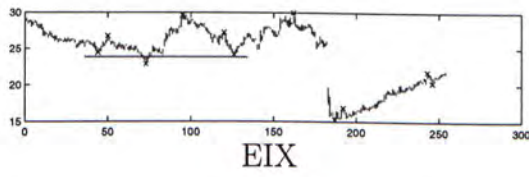
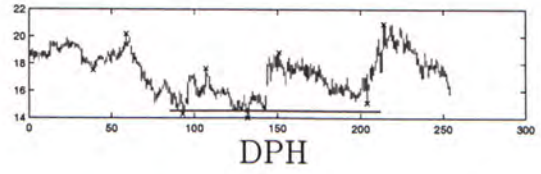
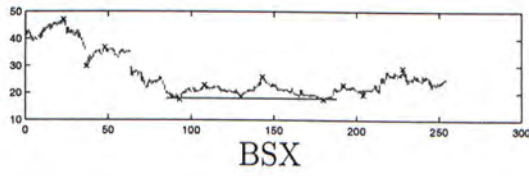




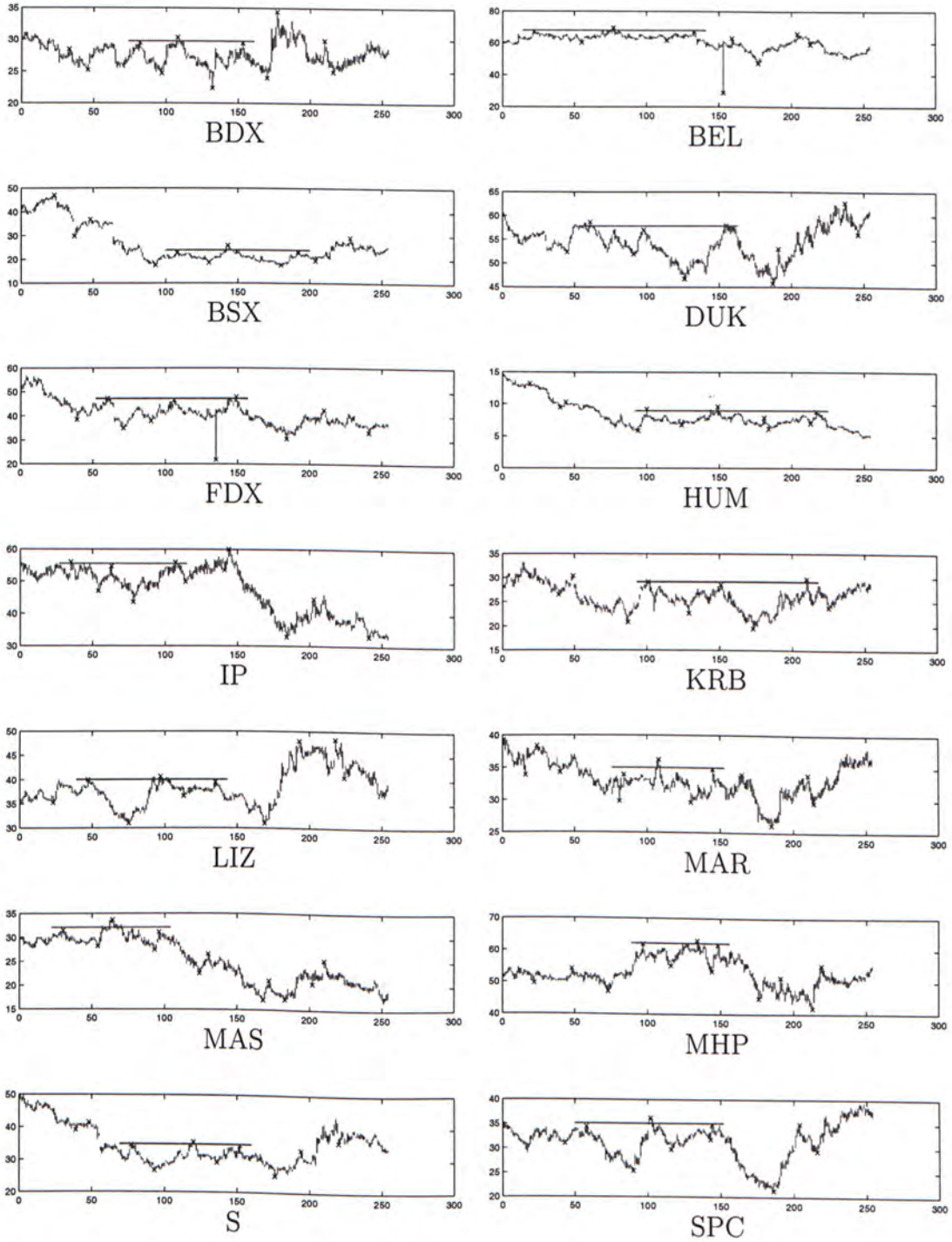


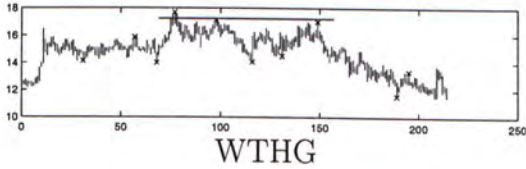
A.11 Triple Bottom





A.12 Triple Top





Bibliography

- [AFS93] Rakesh Agrawal, Christos Faloutsos, and Arn Swami. Efficient similarity search in sequence databases. In *Proceedings of the Fourth Intl. Conf. On Foundations of Data Organization and Algorithm*, pages 69–84, 1993.
- [ALSS95] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databses. In *Proceedings of the 21st VLDB Conference*, pages 490–501, 1995.
- [APWZ95] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying shapes of histories. In *Proceedings of the 21st VLDB Conference*, pages 502–514, 1995.
- [BC96] Donald J. Berndt and James Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. 1996.
- [CF99] Kin Pong Chan and Wai Chee Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering*, pages 126–133, 1999.

- [CLW98] Kam Wing Chu, Sze Kin Lam, and Man Hon Wong. An efficient hash-based algorithm for sequence data searching. *The Computer Journal*, pages 402–415, 1998.
- [CW99] Kam Wing Chu and Man Hon Wong. Fast time-series searching with scaling and shifting. In *PODS*, pages 237–348, 1999.
- [Dem94] Thomas Demark. *The New Science of Technical Analysis*. John Wiley & Sons Inc, New York, 1994.
- [Edw76] Allen L. Edwards. *An introduction to linear regression and correlation*. San Francisco : W. H. Freeman, 1976.
- [EM97] Robert D. Edwards and John F. Magee. *Technical Analysis of Stock Trends*. Chicago, Ill. :John Magee; New York :AMACOM, 1997.
- [FRM94] Christos Faloutsos, M. Ranganthan, and Yannis Manolopoulos. Fast subsequence matching in time-series databses. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 419–429, 1994.
- [Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching, 1984.
- [KAAS99] K. V. Ravi Kanth, Divyakant Agrawal, Amr El Abbadi, and Ambuj Singh. Dimensionality reduction for similarity searching in dynamic databases. *Computer Vision and Image Understanding: CVIU*, 75:59–72, 1999.
- [Keo97] Eamonn Keogh. A fast and robust method for pattern matching in time series database. In *Proceedings of 9th International Conference on Tools with Intelligence (TAI)*, 1997.

- [KJF97] Flip Korn, H.V. Jagadish, and Christos Faloutsos. Efficient supporting adhoc queries in large datasets of time sequences. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 289–300, 1997.
- [KP98] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *In AAAI Workshop on AI Approaches to Time-Series Problems*, pages 239–243, 1998.
- [KP99a] Eamonn J. Keogh and Michael J. Pazzani. An indexing scheme for fast similarity search in large time series databases. In *In Proceedings Eleventh International Conference on Scientific and Statistical Database Management*, pages 239–241, 1999.
- [KP99b] Eamonn J. Keogh and Michael J. Pazzani. Relevance feedback retrieval of time series data. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 183–190, 1999.
- [KPC01] Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, 2001. <http://www.citeseer.nj.nec.com/377269.html>.
- [KS97] Eamonn Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proceedings of the third conference on Knowledge Discovery in Databases and Data Mining*, pages 24–30, 1997.

- [KS01] Tamer Kahveci and Ambuj K. Singh. Variable length queries for time series data. In *ICDE*, 2001. <http://citeseer.nj.nec.com/kahveci01variable.html>.
- [LW98] Sze Kin Lam and Man Hon Wong. A fast projection algorithm for sequence data searching. *Data Knowledge Engineering*, 28(3):321–339, 1998.
- [LYC96] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences. In *In Proceedings of the 12th International Conference on Data Engineering*, pages 546–553, 1996.
- [PCYH00] Sanghyun Park, Wesley W. Chu, Jeehee Yoon, and Chihcheng Hsu. Efficient search for similar subsequences of different lengths in sequence databases. In *Proceedings of the 15th International Conference on Data Engineering*, pages 23–32, March 2000.
- [PH74] T. Pavlidis and S.L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, 23:860–870, 1974.
- [PWZP00] Chang Shing Perng, Haixun Wang, Sylvia R Zhang, and D.Stott Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. In *Proceedings of the 15th International Conference on Data Engineering*, pages 33–42, March 2000.
- [RL90] L. Rabiner and S. Levinson. Isolated and connected word recognition — theory and selected applications. In *Speech Recognition*, Eds. A. Waibel and K.F. Lee, Morgan Kaufmann Publishers, Inc. San Mateo, CA, pages 115–153, 1990.

- [RM97] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD)*, pages 13–25, 1997.
- [Sch98] Jack Schwager. *Schwager on Futures, Technical Analysis*. John Wiley & Sons, 1998.
- [sto] Stock Charts. <http://www.stockcharts.com/>.
- [STZ00] Cyrus Shahabi, Xiaoming Tian, and Wugang Zhao. TSA-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In *Statistical and Scientific Database Management*, pages 55–68, 2000.
- [SZ96] H. Shatkay and S. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of 12th IEEE International Conference on Data Engineering.*, pages 546–553, 1996.
- [TCL98] Alexander Thomasian, Vittorio Castello, and Chung-Sheng Li. Clustering and singular value decomposition for approximate indexing in high dimensional spaces. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management*, pages 201–207, 1998.
- [WSB98] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity search method in high dimensional spaces. In *In Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*, pages pp. 194–205, 1998.
- [WW00] Changzhou Wang and X. Sean Wang. Supporting content-based searches on time series via approximation. In *Proceedings of 12th*

International Conference on Scientific and Statistical Database Management, pages 69–81, 2000.

- [YJC98a] Byoung Kee Yi, H.V. Jagadis, and C.Faloutsos. Supporting fast search in time series for movement patterns in multiple scale. In *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*, pages 251–258, 1998.
- [YJC98b] Byoung Kee Yi, H.V. Jagadish, and C.Faloutsos. Efficient retrieval of similar time sequences under time warping. In *International Conference on Data Engineering*, pages 201–208, 1998.

Publications

- **P. M. Wan** and M. H. Wong, "Efficient and Robust Feature Extraction and Pattern Matching of Time Series by a Lattice Structure.", In Proceedings of the 10th International Conference on Information and Knowledge Management (ACM CIKM), Atlanta, USA, Nov 2001.
- **P. M. Wan** and M. H. Wong, "Finding Chart Patterns using Control Points", Technical Report, 2001.

CUHK Libraries



003924504