A General State-Based Temporal Pattern Recognition

Aihua Zheng

A thesis submitted in partial fulfilment of the requirements of the University of Greenwich for the degree of Doctor of Philosophy

January 2012

The University of Greenwich

School of Computing and Mathematical Science

30 Park Row

Greenwich SE10 9LS

DECLARATION

I certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of Doctor of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations except where otherwise identified by references and that I have not plagiarised the work of others

X----------

Aihua Zheng

X----------

Dr. Jixin Ma

(Supervisor)

X----------

Prof. Miltos Petridis

(Supervisor)

ACKNOWLEDGEMENTS

Education is not the filling of a pail but the lighting of a fire. At the completion of this thesis, I am first and foremost grateful to my supervisors: Dr. Jixin Ma and Prof. Miltos Petridis.

Heartfelt thanks to my supervisors Dr. Jixin Ma and Prof. Miltos Petridis of the University of Greenwich. They gave me great help and inspiration during my overseas research in the United Kingdom. I was deeply affected by their indefatigable academic spirit and vast academic knowledge. They guided me through my academic endeavours step by step with their rigorous encouragement.

Thanks to all the members in QM165 who brought me so much happiness, especially Esther, who shared the same room with me for more than half a year. Friends forever!

Thanks to Professor Bin Luo and Associate Professor Jin Tang in Anhui University, for their great help and support during my successive postgraduate and doctoral programs of study. They not only helped me find the research direction, embark on the way of my cross-country study, but also helped build my confidence through continuous recognition and encouragement. Their noble character has influenced me all this time. Meanwhile, they understand students' economic and living difficulties and have given me great economic support. Also thanks to the Information Processing and Pattern Recognition research group in Anhui University for the loving care from all members.

Thanks to the positive impact of my mother. My mother works very hard and is a strong person.

Thanks to my fiancé who gave me quiet dedication and support throughout and always helped lift my spirits! Thanks for his thoughtful love and care.

Thanks everyone else who has helped me, given me full support but whom I have not had space to mention here. I would like to take this opportunity to express that I will remember your affection in my heart forever!

ABSTRACT

Time-series and state-sequences are ubiquitous patterns in temporal logic and are widely used to present temporal data in data mining. Generally speaking, there are three known choices for the time primitive: points, intervals, points and intervals. In this thesis, a formal characterization of time-series and state-sequences is presented for both complete and incomplete situations, where a state-sequence is defined as a list of sequential data validated on the corresponding time-series. In addition, subsequence matching is addressed to associate the state-sequences, where both non-temporal aspects as well as rich temporal aspects including temporal order, temporal duration and temporal gap should be taken into account.

Firstly, based on the typed point based time-elements and time-series, a formal characterization of time-series and state-sequences is introduced for both complete and incomplete situations, where a state-sequence is defined as a list of sequential data validated on the corresponding time-series. A time-series is formalized as a tetrad (T, R, T_{dur}, T_{gap}), which denotes: the temporal order of time-elements; the temporal relationship between time-elements; the temporal duration of each time-element and the temporal gap between each adjacent pair of time-elements respectively.

Secondly, benefiting from the formal characterization of time-series and state-sequences, a general similarity measurement (GSM) that takes into account both non-temporal and rich temporal information, including temporal order as well as temporal duration and temporal gap, is introduced for subsequence matching. This measurement is general enough to subsume most of the popular existing measurements as special cases. In particular, a new conception of temporal common subsequence is proposed. Furthermore, a new LCS-based algorithm named Optimal Temporal Common Subsequence (OTCS), which takes into account rich temporal information, is designed. The experimental results on 6 benchmark datasets demonstrate the effectiveness and robustness of GSM and its new case OTCS. Compared with binary-value distance measurements, GSM can

distinguish between the distance caused by different states in the same operation; compared with the real-penalty distance measurements, it can filter out the noise that may push the similarity into abnormal levels.

Finally, two case studies are investigated for temporal pattern recognition: basketball zone-defence detection and video copy detection.

In the case of basketball zone-defence detection, the computational technique and algorithm for detecting zone-defence patterns from basketball videos is introduced, where the Laplacian Matrix-based algorithm is extended to take into account the effects from zoom and single defender's translation in zone-defence graph matching and a set of character-angle based features was proposed to describe the zone-defence graph. The experimental results show that the approach explored is useful in helping the coach of the defensive side check whether the players are keeping to the correct zone-defence strategy, as well as detecting the strategy of the opponent side. It can describe the structure relationship between defender-lines for basketball zone-defence, and has a robust performance in both simulation and real-life applications, especially when disturbances exist.

In the case of video copy detection, a framework for subsequence matching is introduced. A hybrid similarity framework addressing both non-temporal and temporal relationships between state-sequences, represented by bipartite graphs, is proposed. The experimental results using real-life video databases demonstrated that the proposed similarity framework is robust to states alignment with different numbers and different values, and various reordering including inversion and crossover.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

GLOSSARY

$\overleftarrow{M}$	An inverse-ordered matching of M
δ, ε	thresholds
(A1), (A2), …	axioms
(F1), (F2), …	formula for fluents
(r1), (r2), …	deduction results in time theory
a, b, c, d,…	states
A_1, A_2, A_3	state-sequences
ACS	All Common Subsequence
B_1, B_2, B_3	state-sequences
$BG = <Q, SS, E>$	Bipartite graph between Q and SS
C_1, C_2, C_3	state-sequences
CA_{122}	Character-Angle of 1-2-2 zone-defence
CA_{131}	Character-Angle of 1-3-1 zone-defence
CA_{23}	Character-Angle of 2-3 zone-defence
CLCS	Compacted LCS
CN_{23}	the set of notes constructing CA_{23}
$D = [SS_1,\ldots, SS_L]$	A state-sequence in database
d	the movement distance
d_{max}	threshold in *kNN*
DTW	Dynamic Time Warping
E	set of edges between notes (defenders' position)
EDR	Edit Distance on Real Sequence
ERP	Edit Distance with Real Penalty
f	fluents
G	zone-defence graph
GSM	general similarity measurement

H	set of Holds
$I = [I_1, \ldots, I_n]$	key-frames sequence
i, j, k	recursion variances
kNN	the k nearest neighbours algorithm
LCS	Longest Common Subsequence
LCS'	length of the Longest Common Subsequence
LM	Laplacian Matrix-based graph matching algorithm
M	A normal matching in $MSM(Q, D)$
m, n	real numbers
MSM	Maximum Size Matching
$NN(Q, SS, d_{max})$	Set of nearest neighbours of all q_i in Q in SS
$NN(q_i, SS, d_{max})$	Set of nearest neighbours of q_i in SS
OED	Original Edit Distance
OTCS	Optimal Temporal Common Subsequence
P	a set of points
p, p_1, p_2	points
$Q = [q_1, q_2, \ldots, q_m]$	Query state-sequence
R	temporal relationship
R^d	denotes d-dimensional real number domain
S	state-sequence
$s_1, s_2, \ldots$	states
$S^1, S^2, \ldots$	state-sequence examples
SS	state-sequences
STD	standard deviation
$t_1, t_2, \ldots$	time-elements
TCS	temporal common subsequence
T_{dur}	temporal duration
T_{gap}	temporal gap
th	threshold in CLCS
T_n	vector of time-elements temporally well ordered

TWED	Time Warp Edit Distance
T-WLCS	Time-warped LCS
U	triple domain with
V	set of the notes (defenders' position)
X , Y	state-sequence
$x_1, \ldots, x_m$	states
$y_1, \ldots, y_n$	states
α, β	parameters between [0, 1]
γ	the acute angle

CHAPTER 1. INTRODUCTION

Section 1.1 The Motivation: Temporal Pattern Recognition

A term *temporal pattern* can be defined as a collection of states (events) that exist along some timeline. For instance, a temporal pattern could be a sequence of actions comprising of eating, walking, taking a shower and then going to sleep.

Temporal pattern recognition is the process of matching two temporal patterns with respect to the temporal properties.

Section 1.1.1 Characterization of Time-series and State-sequences

The notion of time is ubiquitous and vital in modelling natural phenomena and human activities. Time-series and state-sequences are important patterns in data mining and have attracted a lot of interest among researchers [BC1996, DGM1997, FRM1994, KP1998, YJF1998].

However, in most of the proposed formalisms, the fundamental time theories on which time-series and state-sequences are based are not usually explicitly specified. Time-series and sequences are simply expressed as lists in the form of t_1, t_2, …. t_n, or as sequences of collections of observations, and so on, where formal characterizations with respect to the temporal basis are neglected, leaving some critical issues unaddressed. For example:

- What sort of objects do these t_1, t_2, … and t_n belong to? In other words, are they time points, time intervals, or simply some absolute values from the set of real numbers or integers?

- What are the temporal order relationships between t_1, t_2, … and t_n, and/or between the sequence of collections? Are they well-ordered according to ordinal number sequenes, or are they relatively ordered by means of relations such as "Before", "Meets", "During", and so on?

- What are the associations between time-series/state-sequences and non-temporal data that represent various states of the world of discourse?

Therefore, a formal characterization of time-series and state-sequences is required.

Section 1.1.2 State-sequence Matching with Rich Temporal Aspects

The typical temporal pattern recognition is actually the state-sequence matching problem. State-sequence matching can be divided into two categories: whole matching (matching the state-sequences with the same length) and subsequence matching (match the state-sequences with different length). Obviously, the whole matching problem is in fact a special case of subsequence matching, which has been widely researched for many years. In this thesis, without losing generality, subsequence matching is the focus for the state-sequence matching problem. One of the most active and essential research topics in state-sequence matching is the similarity measurement. For general treatment, a versatile similarity measurement should be able to deal with both non-temporal similarity and temporal similarity for any two given state-sequences, where:

(1) **Non-temporal similarity** denotes the similarity between those states appearing in two given state-sequences according to the collection of state elements in the sets, ignoring any temporal issues. For instance in figure 1.1, there is no temporal information in the two state-sequences A_1 = {a, b, c, e, d} and A_2 = {a, b, b, d, d, d, e, g}. The only similarity we can identify is that both of them contain the state {a, b, d, e}.

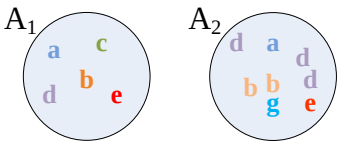


Figure 1.1 Non-temporal set in state-sequence

(2) **Temporal similarity** consists of 3 aspects:

i. *Temporal Order*:

- The temporal relation along the same time axis as shown in figure 1.2 and figure 1.3 where the axis denotes the temporal order. This issue has been well dealt with in most existing subsequence matching algorithms built through dynamic programming.

ii. *Temporal Duration*:

- The duration of each state. For instance, as shown in figure 1.2 where each column block denotes a single unit time interval, the two state-sequences, B_1 and B_2, have different temporal duration assignment functions T_{dur1} = [1, 1, 1, 1] and T_{dur2} = [1, 2, 3, 4] , respectively.

- The overall duration of continuous duplications of states. For instance, as shown in figure 1.2, for state-sequences B_1 and B_3, the common subsequence 'abcd' has different overall durations, T_{dur1} = [1, 1, 1, 1] and T_{dur3} = [2, 3, 4, 3], even if the duration of each unit state is identical to 1. This is because of the duplications of those unit states.

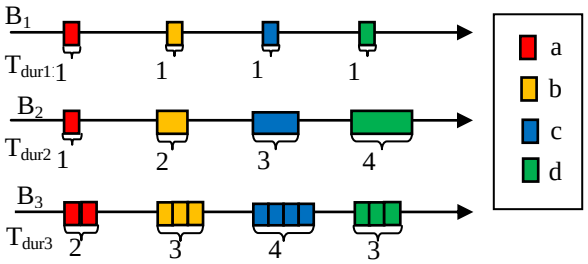


Figure 1.2 Various Temporal Durations in State-sequences

iii. *Temporal Gap*:

- The time element between two adjacent states as shown in figure 1.3. For the state-sequence 'abcd', C_1 and C_2, C_3 have different temporal gap values between ab, bc and cd, with T_{gap1} = [2, 2, 2], T_{gap2} = [1, 2, 3] and T_{gap3} = [1, 1, 3] respectively.

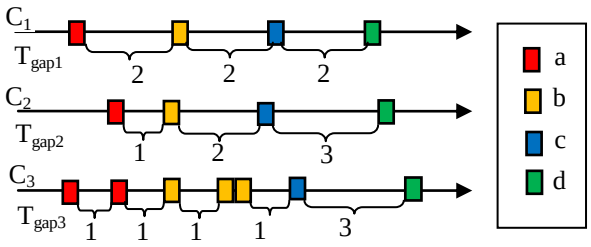


Figure 1.3 Various Temporal gap in state-sequences

Therefore, a general similarity measurement that takes into account both the non-temporal aspects and the rich temporal aspects is required.

Section 1.1.3 Similarity Measurement for State-sequence Matching

Plenty of similarity measurements have been developed in past decades. On one hand, from the point of view of similarity strategy, subsequence matching can be classified into two categories:

- *Edit Distance-based measurements*: match the state-sequences with least operations. Edit Distance (ED) [Lev1965] (also known as Levenshtein Distance) is an innovative distance measurement that has been widely and actively investigated and extended upon by many researchers. ED measures the distance between two state-sequences according to the number of operations (such as *insertion*, *deletion* and *substitution*) required to transform one state-sequence to the other. What follows are some representatives: [WF1973] developed an efficient ED with $O(mn)$ time complexity by

employing dynamic programming algorithm [Bel1957]. Dynamic Time Warping (DTW) [SC1978] allowed time warping such as stretching and shrinking by duplicating the previous state during matching, and was followed by variants such as PDTW [SC1978], SPRING method [SFY2007] and EDTW [APPK2008], and so on. [CN2004] developed the Edit Distance on Real Sequence (EDR). Subsequently, [COO2005] developed the Edit distance with Real Penalty (ERP), which takes the real penalty as the cost of each operation. Distinguishing from DTW, it adds a gap instead of duplicating the previous state while aligning two state-sequences. [MM2008a] extended ED (EDD) to take into account the different costs for different states in the operation and subsequently developed its Multi-Resolution for EDD (MREDD) in [MM2008b]. They distinguish the different unmatched states by adding a frequency function to the basic ED. Highlighting that none of the above measurements takes into account Temporal Gap difference during matching, [Mar2008] produced an elastic measurement, named Time-Warped Edit Distance (TWED), which takes into account the Temporal Gap difference in terms of the temporal index of states.

- *LCS-based measurements*: match the state-sequence according to the presence of common subsequences. The most successful measurement is the longest common subsequences (LCSS) [DGM1997]. The basic idea is to find the longest common sequence in all the sequences along the same temporal order. Several algorithms based on the original LCS have been proposed. Some representative variants of these are: Time-warped LCS (T-WLCS) [GS2004], which counts continuously duplicated common states in the spirit of the Dynamic Time Warping (DTW) [SC1978] algorithm; Compacted LCS (CLCS) [KC2005], where only the common subsequence, the continuous length of which is longer than the specified threshold, (*th*) is counted; All Common Subsequence (ACS) [Wan2007] which measures the similarity by means of counting the number of all common subsequences (including empty string in

actual algorithm) and taking the strategy that the more common subsequences a pair of state-sequences have, the more similar they are.

However, most of these existing similarity measurements characterize temporal similarity in terms of only the temporal order over the state-sequences, whilst other important temporal characters such as the temporal duration of each state itself and the temporal gap between two adjacent states have been neglected. The only noted exception is TWED, which addresses temporal gap similarity in terms of the simple temporal index of states, whilst temporal duration of states is not dealt with at all. According to the formal theory of time-series and state-sequences, a general matching measurement should take into account all of the temporal aspects illustrated above. All the existing measurements can be regarded as special cases of a General Similarity Measurement. Therefore, designing a general similarity measurement for state-sequence matching is a vital and attractive focus of my research.

On the other hand, with respect the ways in which the cost function is specified, similarity measurements can be classified into two alternative categories: (a) binary-value distance models, where the cost functions take binary value (0/1) as matching cost that is not sensitive to noise since they treat the noise and unmatched states with the same cost (1) and (b) real-penalty distance models, in which the cost functions take real difference as matching cost. Generally speaking, binary-value models are more robust since they are not sensitive to the outliers and noise but the real-penalty models are more rational since, in comparing with the logic binary values 0 and 1, the real distance refines the distance. The real-penalty distance models demonstrably outperform binary-value distance models. However, real-penalty distance models are much more sensitive to noise since the real difference between noise and non-noise states may push the overall distance to an abnormal degree.

For instance, take two state-sequences [a_1, a_2, c_0, b_1] and [a_3, a_4, b_0, b_4], as shown in figure 1.4, and suppose the distance between a_0, a_9, b_0, b_9, c_0, c_9 is 10 sequentially, whilst two states are matched if they start with the same character (i.e. a_1 matches a_3).

The matching cost during the two state-sequences can be calculated as 0 + 0 + 1 + 0 = 1, whilst in binary-value measurements it is calculated as 2 + 2 + 10 + 3 = 17. If we keep the characters the same and change the subscription of any state (s) in S_2, the matching cost will remain the same for binary-value measurements. This means the state-sequences with different subscriptions will not be distinguished in binary-value measurements, whilst real-penalty measurements will generate different matching costs. For example, the matching cost between [a_1, a_2, c_0, b_1] and [a_0, a_3, b_0, b_2] is 13, which is smaller than that between [a_1, a_2, c_0, b_1] and [a_3, a_4, b_0, b_4]. However, if noise exists (change b_0 into *$* which is 100 units away from c_9), the matching cost in binary-value measurements remains 1, whilst it becomes 117 in real-penalty measurements.

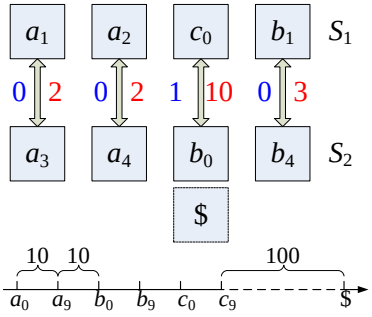


Figure 1.4 Temporal differences between two example state-sequences with binary-value model and real-penalty model

Therefore, the similarity measurement should be as reasonable as real-penalty measurements and also robust in the face of the noise.

Section 1.2 Objective: A General State-Based Framework for Temporal Pattern Recognition

This thesis aims to achieve the following tightly associated research goals:

1). A formal characterization of time-series and state-sequence: based on the typed point-based intervals, a formal characterization of time-series and state-sequence is required to describe the objects of time elements and states, the temporal relationships between them and the associations between

time-series/sequences and non-temporal data.

2). A general similarity measurement for subsequence matching: based on the formal characterization of time-series and state-sequence, it is necessary to design a general similarity measurement (GSM) to take into account both non-temporal and rich temporal aspects. The measurement should be able to tackle temporal order, temporal duration and temporal gap, and also be versatile enough to subsume most of the existing representative similarity measurements as special cases.

3). Investigation of basketball zone-defence detection: as a case study of temporal pattern recognition, the basketball zone-defence detection will be investigated to explore the structural relationship between the defenders.

4). Investigation of video copy detection: as will be demonstrated in another case study of temporal pattern recognition, it is important to design a model that is robust when faced with the re-ordering editing and noise which is ubiquitous in video clips. Furthermore, it is also necessary to design an accurate measurement to distinguish the possible video clips with identical similarity to the query video clip.

Section 1.3 Outline of the Main Contributions

In order to meet the goals outlined above, the following work has been carried out:

1). Based on the typed point based time-elements and time-series, a formal characterization of time-series and state-sequences was consummated with respect to the three temporal aspects including temporal order, temporal duration and temporal gap.

2). Based on the formal characterization of time-series and state-sequence, a general similarity measurement tackling both non-temporal and rich temporal

similarity was designed for state-sequence matching. It is versatile enough to subsume most of the existing representative similarity measurements. Experimental results on 6 benchmark datasets demonstrate that GSM can tackle the most general problems in matching time-series data with rich temporal information. In particular, a new LCS-based similarity measurement named the Optimal Temporal Common Subsequence (OTCS) has been proposed, where a new concept of common subsequence named 'temporal common subsequence' is proposed to describe the rich temporal similarity. In addition, it can release non-uniqueness problems and abnormal output problems in conventional LCS-based similarity measurement.

3). As a case-study of temporal pattern recognition, a system to detect the zone-defence strategy in basketball videos was investigated, where the detecting task was transferred into graph matching problem. An improved Laplacian Matrix-based graph matching algorithm was designed for basketball zone-defence detection. Meanwhile, due to the computational complexity of graph matching algorithms, an efficient feature descriptor, named Character-Angle based feature descriptor, was designed for zone-defence graphs.

4). As another case of temporal pattern recognition, a hybrid state-sequence matching framework was designed for video copy detection, where both the non-temporal and temporal similarities were taken into account. The non-temporal similarity was defined in the form of Euclidean distance whilst the temporal similarity was constructed with temporal order similarity, temporal alignment similarity and temporal concentration similarity.

Section 1.4 Thesis Structure

The rest of this thesis is organized as follow:

Chapter 2 is a comprehensive review of the representations of time-series and state-sequence as well as the popular existing measurements for state-sequence matching based on the typed point based time-elements and time-series. A formal characterizations of time-series and state-sequences in introduced for both complete and incomplete situations, where a state-sequence is defined as a list of sequential data validated on the corresponding time-series. While a state-sequence is formalized as the triple domain $U = S \times D \times G$, where: $S \subset R^d$ denotes d-dimensional domain of non-temporal data ordered in consequential (that is, "Meets or Before") temporal order and D, $G \subset R$ denote the domains of temporal duration and temporal gap respectively. In addition, the framework of the general similarity measurement is addressed to associate state-sequence matching, where both the non-temporal aspects and temporal aspects including temporal order, temporal duration and temporal gap should be taken into account.

In chapter 3, based on the general similarity measurement, a new conception of temporal common subsequence is first proposed, and then a new LCS-based algorithm named Optimal Temporal Common Subsequence (OTCS) that takes into account rich temporal information (including temporal order, temporal duration and temporal gap) between state-sequences is finally designed and tested on news video retrieval. The experimental results demonstrate the effectiveness and robustness of the new algorithm.

In chapter 4, a general similarity measurement (GSM), which takes into account both non-temporal and rich temporal information, including temporal order, as well as temporal gap and duration, is introduced for subsequence matching. Benefitting from a formal characterization of time-series and state-sequences, this measurement is general enough to subsume most of the popular existing measurements as special cases. In particular, compared with the binary-value similarity measurements, the GSM can distinguish the difference caused by various states in the same operation, whilst, compared with the real-penalty similarity measurements, it can also filter out the noise which may lead the similarity into a abnormal level.

In chapter 5, the basketball zone-defence detection is investigated as a case study of temporal pattern recognition. The Laplacian Matrix-based algorithm is extended to take account of the effects from zoom and single defender's translation in zone-defence graph matching. Furthermore, a set of character-angle based features are proposed to describe the structure relationship between defender-lines in the zone-defence graphs. Experimental results demonstrate the robust performance in both simulation and real-life applications, especially when disturbance exists.

In Chapter 6, video copy detection is investigated as another case study. A hybrid framework addressing both non-temporal and temporal relationships between state-sequences, which are represented by bipartite graphs, is proposed. The experimental results using real-life news video database demonstrate that the proposed similarity model is robust when faced with states alignment with different numbers and different values, and various reordering including inversion and crossover.

Finally, a summary of conclusion and recommendations for future work is presented in chapter 7.

There are also six appendices for this thesis. These are six of my published papers tightly associated with this research.

CHAPTER 2. LITERATURE REVIEW

Focusing on the two objectives of this thesis, a detailed review of related works will be presented in this chapter. First, we shall elaborate on the evolution of representations of primitive time and time-series, followed by the conventional existing measurements for state-sequence matching.

Section 2.1 The ontology of primitive time

There has been a longstanding debate in the literature on the issue of which sorts of objects should be taken as the time primitive. Commonsense, on one the hand, denotes that points are needed for both theoretical and practical modelling of temporal phenomena. For instance, it is intuitive and convenient to associate punctual events, such as "The database was updated at 0:00 midnight" etc., with instantaneous points rather than durative intervals. On the other hand, intervals also seem to be needed for representing temporal phenomena that take up time with positive duration, e.g., "He ran 3 hours yesterday morning".

Generally speaking, there are three known objects that may be taken as the time primitive:

- *points*, i.e., instants of time with no duration;
- *intervals*, i.e., periods of time with positive duration;
- *both points and intervals*

Section 2.1.1 Point-based Time Structure

The so-called point-based time structure was first proposed by Bruce [Bru1972]: a typical time structure based on points only as primitive is an ordering (P, ≤), where P is a set of points, and ≤ is a relation that (partially or totally) orders P. In point-based

systems, intervals may be defined as derived temporal objects, either as sets of points [DGM1997], or as ordered pairs of points [Gal1990, S1987, YJF1998].

Problems

Point-based time structure provides an efficient indexing method for temporal systems, but may suffer from the requirement that precise time values for all temporal data need to be available. Generally speaking, in many AI systems, temporal knowledge can be uncertain and incomplete. For instance, we may only know that event *A* happened before event *B*, without knowing their precise starting and finishing time, or what happened between them. Incomplete relative temporal knowledge such as this is typically derived from humans, where complete and absolute temporal information is rarely available and remembered for knowledge representation and reasoning.

It has been argued by some researchers that defining intervals as objects derived from points may lead to the so-called Dividing Instant Problem [AH1989, Bru1972, Lad1987], which is in fact an ancient historical puzzle encountered when attempting to represent what happens at the boundary point that divides two successive intervals. For instance, consider the fire example cited in [Ben1983]:

A fire that had been burning was later burnt out.

Intuitively, we can assume the two states, i.e., "The fire was burning" and "The fire was not burning" hold true throughout two successive point-based intervals, say $<p_1, p>$ and $<p, p_2>$, respectively. The question then becomes: "Was the fire burning or not burning at point *p*?" This, in terms of the *open* or *closed* nature of the involved point-based intervals, turns out to be the question of which of the two successive intervals, i.e., $<p_1, p>$ and $<p, p_2>$, is closed/open at the dividing point *p*? Virtually, there are four possible cases:

(a) The fire was burning rather than not burning at *p*;

(b) The fire was not burning rather than burning at *p*;

(c) The fire was both burning and not burning at *p*;

(d) The fire was neither burning nor was it not burning at *p*.

While both (c) and (d) are absurd, since they violate the *Law of Contradiction* and the *Law of Excluded Third* [Ben1983] respectively, the choice between (a) and (b) must be arbitrary and artificial. In fact, since we have no better reason, from the point of view of philosophy, for saying that the fire was burning than for saying that it was not burning at the dividing-instant, such an arbitrary approach has been criticized as unjustifiable and hence unsatisfactory [Ben1983, All1984, Gal1990, Vil1994].

Section 2.1.2 Interval-based Time Structure

The point-based structure of time has been challenged by many researchers who believe that time intervals are more suited for representing commonsense temporal knowledge, notably in the domain of linguistics and artificial intelligence. It is argued that intervals should be taken as the temporal primitive, where points may be constructed with a subsidiary status, e.g., as "*maximal nests*" of intervals that share a common intersection, or as "*meeting places*" of intervals [Bee1992, BC1996, Lad1987, Vil1994]. For instance, Allen's temporal theory [All1984, AH1989] is a representative example of the interval-based approach, which posits a set of intervals as the primitive temporal entities. Over the time intervals, Allen introduces thirteen temporal relations, including "Equal", "Before", "After", "Meets", "Met-by", "Overlaps", "Overlapped-by", "Starts", "Starts-by", "During", "Contains", "Finishes" and "Finished-by", which can be derived from the single *immediate predecessor* relation "Meets" [BC1996].

As Allen claims in his paper [All1984], the interval-based approach avoids the annoying question of whether or not a given point is part of, or a member of a given interval, and therefore can successfully overcome/bypass puzzles such as the *Dividing*

Instant Problem. Allen's contention is that nothing can be true at a point, for a point is not an entity at which things happen or are true. However, as Galton [Gal1990] shows in his critical examination of Allen's interval logic [All1984], a theory of time based only on intervals is inadequate for reasoning correctly about continuous change. Furthermore, instantaneous phenomena do exist in the real world and therefore make points necessary for general temporal reference. For instance, consider the following scenario:

A ball was thrown into the air from the east to the west.

By common sense, the state that the ball was at the east of and below its apex was immediately followed by the state that the ball was at its apex, and which, in turn, was immediately followed by the state that the ball was at the west of and below the apex. Also, the time by which the ball was at its apex – neither at the east of the apex nor at the west of the apex, should be a point with zero duration, rather than any interval or moment [AH1989], no matter how small it might be. In fact, during the process of the motion of the ball, the velocity of the ball became zero only at the time point when the ball was at its apex.

Problems

The interval-based time structure was proposed based on Allen's interval theory. However, it has been argued in [Gal1990] that Allen's interval theory lacks clarity in semantics and completeness. In addition, the corresponding matching algorithm proposed in [JAS2002] lacks in theoretical foundation. Therefore, a new matching algorithm that still uses the interval-based time structure is required.

Section 2.1.3 Point and Interval-based Time Structure

For the sake of general treatments, we shall take the time theory proposed previously in [MK1994] as the temporal basis, in which both points and intervals are

addressed as temporal primitives on an equal footing: points do not have to be defined as limits of intervals and intervals do not have to be constructed out of points.

The time theory, *T*, takes a nonempty sort, *T*, of primitive time elements, with a primitive order relation 'Meets' over time elements, and a function 'Dur' from time elements to non-negative real numbers. The basic set of axioms concerning the triad (*T*, Meets, Dur) is given as below:

(A1). $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_1, t_2) \wedge \text{Meets}(t_1, t_3) \wedge \text{Meets}(t_4, t_2) \Rightarrow \text{Meets}(t_4, t_3))$

That is, if a time element meets two other time elements, then any time element that meets one of these two must also meet the other. This axiom is actually based on the intuition that the "place" where two time elements meet is unique and closely associated with the time elements [AH1989].

(A2). $\forall t \exists t_1,t_2(\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$

That is, each time element has at least one immediate predecessor, as well as at least one immediate successor.

(A3). $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_1, t_2) \wedge \text{Meets}(t_3, t_4) \Rightarrow$

$\text{Meets}(t_1, t_4) \nabla \exists t'(\text{Meets}(t_1, t') \wedge \text{Meets}(t', t_4)) \nabla \exists t''(\text{Meets}(t_3, t'') \wedge \text{Meets}(t'', t_2)))$

where ∇ stands for "exclusive OR". That is, any two meeting places are either identical or there is at least a time element standing between the two meeting places if they are not identical.

(A4). $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_3, t_1) \wedge \text{Meets}(t_1, t_4) \wedge \text{Meets}(t_3, t_2) \wedge \text{Meets}(t_2, t_4)) \Rightarrow t_1 = t_2)$

That is, the time element between any two meeting places is unique.

N.B. In this thesis, for any two adjacent time elements, that is to say two time elements t_1 and t_2 such that $\text{Meets}(t_1, t_2)$, we shall simply use $t_1 \oplus t_2$ to denote their ordered

union. The existence of such an ordered union of any two adjacent time elements is guaranteed by axioms A2 and A3, whilst its uniqueness is guaranteed by axiom A4.

(A5). $\forall t_1,t_2(\text{Meets}(t_1, t_2) \Rightarrow \text{Dur}(t_1) > 0 \vee \text{Dur}(t_2) > 0)$

That is to say, time elements with zero duration cannot meet each other.

(A6). $\forall t_1,t_2(\text{Meets}(t_1, t_2) \Rightarrow \quad \text{Dur}(t_1 \oplus t_2) = \text{Dur}(t_1) + \text{Dur}(t_2))$

Thus, the "ordered union" operation over time elements is consistent with the conventional "addition" operation over the duration assignment function, i.e., 'Dur'.

N.B. In the time theory T introduced here, we adopt the following results of real number theory:

(r1) The set of real numbers is totally ordered by the less-than-or-equal-to relation '$\leq$', where '$>$' is the "bigger than" relation, that is, not($\leq$).

(r2) '$+$' is the conventional addition operator over (non-negative) real numbers.

In terms of the 'Meets' relation, other exclusive order relations over time elements can be derived as below:

$\text{Equal}(t_1, t_2) \Leftrightarrow \exists t',t''(\text{Meets}(t', t_1) \wedge \text{Meets}(t', t_2) \wedge \text{Meets}(t_1, t'') \wedge \text{Meets}(t_2, t''))$

$\text{Before}(t_1, t_2) \Leftrightarrow \exists t(\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$

$\text{Overlaps}(t_1, t_2) \Leftrightarrow \exists t,t_3,t_4(t_1 = t_3 \oplus t \wedge t_2 = t \oplus t_4)$

$\text{Starts}(t_1, t_2) \Leftrightarrow \exists t(t_2 = t_1 \oplus t)$

$\text{During}(t_1, t_2) \Leftrightarrow \exists t_3,t_4(t_2 = t_3 \oplus t_1 \oplus t_4)$

$\text{Finishes}(t_1, t_2) \Leftrightarrow \exists t(t_2 = t \oplus t_1)$

After(t_1, t_2) $\Leftrightarrow$ Before(t_2, t_1)

Overlapped-by(t_1, t_2) $\Leftrightarrow$ Overlaps(t_2, t_1)

Started-by(t_1, t_2) $\Leftrightarrow$ Starts(t_2, t_1)

Contains(t_1, t_2) $\Leftrightarrow$ During(t_2, t_1)

Finished-by(t_1, t_2) $\Leftrightarrow$ Finishes(t_2, t_1),

Met-by(t_1, t_2) $\Leftrightarrow$ Meets(t_2, t_1)

On one hand, the completeness of the 13 possible exclusive order relations (the above 12 plus Meets) between any two time elements can be simply characterized by a single axiom as below:

$\forall t_1,t_2$(Equal(t_1, t_2) $\vee$ Before(t_1, t_2) $\vee$ After(t_1, t_2) $\vee$ Meets(t_1, t_2) $\vee$ Met-by(t_1, t_2)

$\vee$ Overlaps(t_1, t_2) $\vee$ Overlapped-by(t_1, t_2) $\vee$ Starts(t_1, t_2) $\vee$ Started-by(t_1, t_2)

$\vee$ During(t_1, t_2) $\vee$ Contains(t_1, t_2) $\vee$ Finishes(t_1, t_2) $\vee$ Finished-by(t_1, t_2))

On the other hand, the exclusiveness of these 13 order relations needs to be characterized by 78 axioms of the following form:

$\forall$t$_1$,t$_2$($\neg$Relation1(t_1, t_2) $\vee$ $\neg$Relation2(t_1, t_2))

where Relation1 and Relation2 are two distinct relations from the above 13 relations.

N.B. In the above, 78 is the combinational number C_{13}^2 = 13!/2!11!.

For convenience of expression, we shall extend Allen's non-exclusive relation 'In', which is defined for intervals alone [All1984], to accommodate both time intervals and points, and in addition, to introduce another temporal relation, 'Part', as below:

$$In(t_1, t_2) \Leftrightarrow Starts(t_1, t_2) \vee During(t_1, t_2) \vee Finishes(t_1, t_2)$$

$$Part(t_1, t_2) \Leftrightarrow Equal(t_1, t_2) \vee In(t_1, t_2)$$

Problems

The point & interval-based time structure seems to be general and efficient enough for temporal representation. However, the following two issues still exist and solving them is a motivation of this thesis.

1) The fundamental temporal theory of the point & interval-based time structure is the temporal theory of Ma and Knight in [MK1994, MK1996]. However, only temporal order and temporal relationship is specified. The other temporal aspects such as temporal duration and temporal gap were neglected.

2) Only the basic temporal representation is illustrated whilst the corresponding matching algorithm, especially with respect to the rich temporal aspects, is required.

Section 2.2 The notion of time and time-series

Data mining is the process of finding trends and patterns in data [Gro1999]. Generally speaking, data mining requires some historical knowledge about the internal temporal relationships of certain patterns such as those in Decision Support, Diagnosis and Explanation, Forecasting/Prediction, Planning/Scheduling, and History Reconstruction, etc. In particular, time-series and state-sequences are important patterns in data mining and have attracted the interest of many researchers [BC1996, DGM1997, FRM1994, KP1998, YJF1998].

Section 2.2.1 The notion of time

1) What is time?

Even today we still cannot define “time” as we define any real thing. We can measure time, yet do not know what time is, although we hang "time" on the wall or on the wrist. According to Einstein's theory of relativity, we know that time can be extended or shortened. That is why the physicists set the time simply as a sequence of events and mark them with time, such as the person's birthday or the shelf life of food.

2) Is time like a river flow, or with intermittency as a replacement?

Unfortunately, no theory or experiment has confirmed that time is flowing in a continuous manner or like every frame in a movie screen, giving a continuous picture of intermittency. Research on the continuity and the intermittency of time is ubiquitous and vital in modelling natural phenomena and human activities.

Now we are back to the "continuity" of time. The strange thing is that it can approach a continuous or intermittent flow, yet the smallest calculable time interval is the same as "Planck time". In short, time is a continuous tape, and physicists regard it as an interlocking, non-continuous necklace.

3) For everyone, is time passing in the same way?

Einstein's theory suggests that the answer is no. In fact, the same as space, time is also relative. What does “relative” mean in this context? That is, in order to completely and unambiguously describe an event, this event should be placed in a reference system. For example, if I meet someone at the end of the road, then the "end" might just be the beginning of another person’s road. If I add "at the end of the road behind the plaza," then the event "meet" is accurate. If I said

10 years later, then I must point out to which reference system 10 years has passed. Obviously, in everyday life there is no need to be detailed. However, detail is vital in time-series analysis.

Section 2.2.2 The notion of time-series and state-sequence

A time-series is a chronological series of observations made. In accordance with different phenomena or problems studied, one can obtain all kinds of time-series. For example, some economists observe fluctuations in the price index; a meteorologist studies the rainfall in some location and electrical engineers study electronic receiver's internal noise. All of them will observe a string of data measured by some unit of measurement. The natural order is the chronological order of appearance of data in the time-series. The typical essential characteristic of time-series is the dependency between adjacent observations. This dependence has great practical significance. Time-series analysis is addressed in the techniques of this dependence analysis. The new method of prediction of time-series data not only provides a effective prediction method for time-series data produced from for example the national economy, agriculture, biology, meteorology, hydrology and other fields, but also enables researchers to exercise math skills and programming techniques.

Broadly speaking, a state is the way something is with respect to its main attributes. A state-sequence is defined as a list of states together with corresponding time-series.

In order to analyze time-series and state-sequences, formalism is required. However, in most proposed formalisms, the fundamental time theories upon which time-series and state-sequences are built up are usually not explicitly specified. Time-series and sequences are simply expressed as lists in the form of t_1, t_2, …. t_n, or as sequences of collection of observations, and so on, where formal characterizations with respect to the temporal basis are neglected, leaving some critical issues unaddressed. For example:

- What sorts of objects are t_1, t_2 …and t_n? In other word, what sorts of objects should be taken as the time primitive? Are they time points, time intervals, or simply some absolute values from the real numbers, integers, or the clock?
- What are the temporal order relationships between t_1, t_2…and t_n, and/or between the sequence of collections? Are they simply well-ordered as according to natural numbers, or might they be relatively ordered by means of relations such as "Before", "Meets", "During", and so on?
- What are the associations between time-series/state-sequences and non-temporal data that represent various states of the world in discourse?

Section 2.3 LCS-Based Subsequence Matching

The Longest Common Subsequence (LCS) is a typical similarity measurement for subsequence matching. Recently, a group of LCS-like measurements were proposed for subsequence matching. Given two state-sequences $X = [x_1, \ldots, x_m]$ and $Y = [y_1, \ldots, y_n]$, several algorithms based on the original LCS have been proposed to match these two state-sequences. Some representative variants of these are: (i) Compacted LCS (CLCS) [KC2005] where only the common subsequences, the continuous length of which is longer than the specified threshold (*th*), is counted; (ii) All Common Subsequence (ACS) [Wan2007] which measures the similarity by means of counting the number of all common subsequences (including empty strings) and taking the strategy that the more common subsequences a pair of state-sequences have, the more similar they are; and (iii) Time-Warped LCS (T-WLCS) [GS2004], which counts continuously duplicated common states in the spirit of the Dynamic Time Warping (DTW) [SC1978] algorithm. Each of these is discussed in further detail in the following four sub-sections.

Section 2.3.1 Original Longest Common Subsequence (LCS)

The basic idea of the original LCS algorithm [BHR2000] is to find the longest

subsequence common to two state-sequences (X and Y) along the same temporal order. Then the length of the common subsequence is counted as the similarity between the two given state-sequences. We shall now explain the solution of LCS in what follows. Suppose the current state pair is x_i and y_j, a table with size of $(m+1) \times (n+1)$ is designed to store the process of LCS computation. An empty state is added in front of each state-sequence. The procedure of finding the longest common subsequence can be illustrated by the following 3 rules:

1) Setup rule: i = 0 or j = 0

In this case, we are comparing the empty state with another state-sequence. Obviously, the common state between an empty state-sequence and any state-sequence is the empty state as well. Therefore, $\mathrm{LCS}(X_0, Y_j) = \mathrm{LCS}(X_i, Y_0) = \phi$.

2) Matching rule: $x_i = y_j$

In this case, the two state-sequences match each other by ending in the same state. Shorten each state-sequence by removing states x_i and y_j from state-sequences X and Y respectively. The longest common subsequence would be the LCS of the shortened sequences appended by the removed state (x_i or y_j). In terms of prefixes:

$$LCS(X_i,\ Y_j) = (LCS(\ X_{i-1},\ Y_{j-1});\quad x_i) \text{ or } (LCS(\ X_{i-1},\ Y_{j-1});\quad y_j) \qquad (2\text{-}1)$$

where X_i and Y_j indicates the substring $[x_1, x_2, \ldots, x_i]$ and $[y_1, y_2, \ldots, y_j]$ for $1 \le i \le m$, $1 \le j \le n$, the semicolon indicates that the following element, x_i, is appended to the sequence.

3) Unmatching rule $x_i \neq y_j$

In this case, X and Y do not end in the same state. Then the LCS of X and Y is the longer of the two sequences $\mathrm{LCS}(X_i, Y_{j-1})$ and $\mathrm{LCS}(X_{i-1}, Y_j)$.

"Dynamic programming can be thought of as being the reverse of recursion. Recursion is a top-down mechanism – we take a problem, split it up, and solve the

smaller problems that are created. Dynamic programming is a bottom-up mechanism- we solve all possible small problems and then combine them to obtain solutions for bigger problems. The reason that this may be better is that, using recursion, it is possible that we may solve a small subproblem many times. Using dynamic programming, we solve it once."[1]

According to the above, the recursive function of LCS can be defined as follows:

Definition 2-1: the longest common subsequence of given state-sequences $X = [x_1, \ldots, x_m]$ and $Y = [y_1, \ldots, y_n]$ is:

$$LCS(X_i, Y_j) = \begin{cases} \phi & if\ i = 0\ or\ j = 0 \\ LCS((X_{i-1}, Y_{j-1}), x_j) & if\ x_i = y_j \\ longer(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & if\ x_i \neq y_j \end{cases} \quad (2\text{-}2)$$

where $0 \leq i \leq m$, $0 \leq j \leq n$.

In order to measure the similarity between two state-sequences, the length of LCS is defined as below:

Definition 2-2: the length of the LCS of two given state-sequences X and Y is:

$$LCS'(X_i, Y_j) = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ LCS'(X_{i-1}, Y_{j-1}) + 1 & if\ x_i = y_j \\ \max(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & if\ x_i \neq y_j \end{cases} \quad (2\text{-}3)$$

The original LCS is designed for 1-dimention state-sequences. In order to cope with multi-dimension situations, [VHGK2003] extended the original LCS to 2-dimention situations:

[1] http://www.ics.uci.edu/~dan/class/161/notes/6/Dynamic.html

$$LCSS_{\delta,\varepsilon}(X_i, Y_j) = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ LCSS_{\delta,\varepsilon}(X_{i-1}, Y_{j-1}) + 1 & if\ |x_{ik} - y_{jk}| < \varepsilon\ \ for\ all\ k \\ & \quad and\ |j - i| \le \delta \\ \max(LCSS_{\delta,\varepsilon}(X_i, Y_{j-1}), LCSS_{\delta,\varepsilon}(X_{i-1}, Y_j)) & otherwise \end{cases} \quad (2\text{-}4)$$

where the constant δ and ε denote the controller in space and time respectively.

Example evolution:

For the given state-sequence X = [*abcd*] and Y = [*adcbd*], the procedure of the longest common subsequence is illustrated next. Assuming that the LCS function starts with zero, two empty states are inserted as prefixes of two state-sequences respectively. $X_0 = x_0 = Y_0 = y_0 = \emptyset$ is placed as shown in table 2.1 with the size 4×5, where $LCS(X_i, Y_j)$ denotes the longest common subsequence between X_i and Y_j, and the arrow directs to the source cell of current longest common subsequence, for example ←↑ indicates the current cell $LCS(X_i, Y_j)$ is generated by $longer(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j))$ from Eq.(2-2).

Table 2.1 LCS subsequence table

LCS(X, Y)	Ø	*a*	*d*	*c*	*b*	*d*
Ø	Ø	Ø	Ø	Ø	Ø	Ø
a	Ø	↖ ***a***	← *a*	← *a*	← *a*	← *a*
b	Ø	↑ *a*	←↑ *a*	←↑ *a*	↖ ***ab***	← *ab*
c	Ø	↑ *a*	↑ *a*	↖ ***ac***	←↑ *ac/ab*	←↑ *ac/ab*
d	Ø	↑ *a*	↖ ***ad***	←↑*ac/ad*	←↑ *ac/ad*	↖ ***acd/abd***

The LCS table is designed to store the step of LCS calculation between X and Y placed in the first column and the first row while LCS(X_i, Y_j) indicates the longest common subsequence of X_i and Y_j.

LCS(X_0, Y_j) is always Ø for j = 0, 1, …, n since the longest common sequence between the empty sequence and any other sequence is considered as empty. Likewise, LCS(X_i, Y_0) = Ø for i = 0, 1, …, m (*setup rule*).

LCS(X_1, Y_1) is indicated by states '*a*' (x_1) and '*a*' (y_1). According to the *matching rule*, LCS(X_1, Y_1) = (LCS(X_0, Y_0), '*a*') = '*Øa*', simplified as 'a'.

LCS(X_1, Y_2) is indicated by states '*a*' (x_1) and '*d*' (y_2). According to the *unmatching rule*, LCS(X_1, Y_2) = longer (LCS(X_1, Y_1), LCS(X_0, Y_2)) = longer('*a*', '*Ø*') = '*a*'.

LCS(X_1, Y_3) is indicated by states '*a*' (x_1) and '*c*' (y_3). *Unmatching rule*, LCS(X_1, Y_3) = longer (LCS(X_1, Y_2), LCS(X_0, Y_3)) = longer('*a*', '*Ø*') = '*a*'.

LCS(X_1, Y_4) is indicated by states '*a*' (x_1) and '*b*' (y_4). *Unmatching rule*, LCS(X_1, Y_4) = longer (LCS(X_1, Y_3), LCS(X_0, Y_4)) = longer('*a*', '*Ø*') = '*a*'.

LCS(X_1, Y_5) is indicated by states '*a*' (x_1) and '*d*' (y_5). *Unmatching rule*, LCS(X_1, Y_5) = longer (LCS(X_1, Y_4), LCS(X_0, Y_5)) = longer('*a*', '*Ø*') = '*a*'.

Analogously, the rest of the table can be filled. The corresponding length of LCS is stored in the table 2.2.

Table 2.2 LCS length table

LCS'(X, Y)	*Ø*	*a*	*d*	*c*	*b*	*d*
Ø	0	0	0	0	0	0
a	0	**1**	←1	←1	←1	←1
b	0	↑1	↑1	↑1	**2**	←2
c	0	↑1	↑1	**2**	↑2	↑2
d	0	↑1	**2**	↑2	↑2	**3**

Problem of LCS

In order to visually demonstrate the performance of LCS-based measurements, five state-sequences are defined as follows: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*]. According to Eq.(2-3), the similarity table can be obtained as follows:

Table 2.3 LCS' table between five example state-sequences

Similarity		S^1	S^2	S^3	S^4	S^5
LCS'	S^1	**4**	3	**4**	**4**	2
	S^2	3	7	**4**	4	**5**
	S^3	4	4	8	8	4
	S^4	4	4	8	11	4
	S^5	2	5	4	4	7

"Non-uniqueness" problem: different state-sequences have the same similarity to the query state-sequence. For instance, for a given three state-sequence pairs (S^1, S^1), (S^1, S^3) and (S^1, S^4) with the same longest common subsequence 'abcd', we shall get LCS'(S^1, S^1) = LCS'(S^1, S^3) = LCS'(S^1, S^4) = 4, which means S^1 has the same similarity to S^3 and S^4 as well as to S^1 itself, whereas in real-life application the measurement should distinguish the similarity as clear as possible.

"Unreasonable problem": some other abnormal or unreasonable results occur when continuously duplicated common states exist frequently in state-sequences. For example, LCS'(S^2, S^5) > LCS'(S^2, S^3). The reason is that the continuously duplicated common states are counted without distinguishing from the non-duplicated common states. However, according to the definition of temporal common subsequences, the similarity degree between S^3 and S^2 should in fact be higher than that between S^5 and S^2.

Section 2.3.2 Compacted LCS (CLCS)

In contrast to the original LCS, in Compacted LCS (CLCS) [KC2005] only the common subsequence, the continuous length of which is longer than the specified threshold (*th*), is counted. The procedure for CLCS is as the following 4 steps:

Step 1: calculate the Matching Matrix. Without lose of generality, assume that X is the query state-sequence and Y denotes one of the state-sequences in the database. The Matching Matrix is defined as: for i=1, 2, … m.

$$m(i)=\begin{cases}1, & if\ i-th\ state\ is\ matched\\ 0, & if\ i-th\ state\ is\ unmatched\end{cases} \tag{2-5}$$

In fact, the length of the original LCS can be obtained by computing $\sum_{i=1}^{n} m(i)$.

Step 2: the length of continuously matched subsequence:

$$LC(i)=\begin{cases}LC(i-1)+m(i), & if\ m(i)=1\\ m(i) & if\ m(i)=0\end{cases} \tag{2-6}$$

Step 3: the length of continuously matched subsequence separated by unmatched subsequence: for *i*=1, 2, … *n*-1

$$SLC(i)=\begin{cases}LC(i), & if\ m(i+1)=0\\ 0, & if\ m(i+1)=1\end{cases} \tag{2-7}$$

In this situation, the length of LCS can be repressed by $\sum_{i=1}^{n} SLC(i)$

Step 4: calculate the compacted-LCS (CLCS) where only the length of continuous matched common subsequence is longer than the threshold (*th*) is counted:

$$CLCS(X,Y)=\sum_{i=1}^{n} MLC(i)\begin{cases}MLC(i)=SLC(i), & if\ SLC(i)>th\\ MLC(i)=0, & if\ SLC(i)<th\end{cases} \tag{2-8}$$

where $th=k\cdot n, 0<k<1$.

Example evolution:

For the same five state-sequence: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*]. The examples of CLCS are calculated in table 2.4:

Table 2.4 Example evolution of CLCS length

(a) CLCS(S^1, S^1) table

CLCS(S^1, S^1)	Ø	*a*	*b*	*c*	*d*
Ø	0	0	0	0	0
a	0	0	0	0	0
b	0	0	0	0	0
c	0	0	0	1	1
d	0	0	0	1	**2**

(b) CLCS(S^1, S^2) table

CLCS(S^1, S^2)	Ø	*a*	*a*	*a*	*a*	*a*	*b*	*c*
Ø	0	0	0	0	0	0	0	0
a	0	**0**	0	0	0	0	0	0
b	0	0	0	0	**0**	0	0	0
c	0	0	0	0	0	0	0	**3**
d	0	0	0	0	0	0	0	3

(c) CLCS(S^1, S^3) *table*

CLCS(S^1, S^3)	Ø	*a*	*a*	*b*	*b*	*c*	*c*	*d*	*d*
Ø	Ø	0	0	0	0	0	0	0	0
a	0	**1**	1	1	1	1	1	1	1
b	0	1	1	**2**	2	2	2	2	2
c	0	1	1	2	2	**3**	3	3	3
d	0	1	1	2	2	3	3	**4**	**4**

(d) CLCS(S^1, S^4) table

CLCS(S^1, S^4)	Ø	*a*	*a*	*e*	*b*	*b*	*f*	*c*	*c*	*g*	*d*	*d*
Ø	Ø	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	0	0	0	0	0

(e) CLCS(S^1, S^5) table

CLCS(S^1, S^5)	Ø	*a*	*a*	*a*	*a*	*b*	*b*	*b*
Ø	Ø	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	0

Problem of CLCS:

"Non-uniqueness" problem: this problem is ubiquitous in CLCS as shown in the tables, where CLCS(S^1, S^3)= CLCS(S^1, S^4)= CLCS(S^2, S^4)= CLCS(S^3, S^4)= CLCS(S^5, S^4) = 0. The non-uniqueness problem must be more serious than the original LCS since the threshold smoothes the difference of the two state-sequences: the length of continuous matched states will be smoothed to be the same level (0) if it varies from 1 to *th*-1.

"Unreasonable problem": one will also get the unreasonable phenomenon CLCS(S^2, S^5) > CLCS(S^2, S^3). The reason is that the matched states "c" are separated and the length is 1, which is smaller than the threshold *th*=2.

Particularly, CLCS is very fluctuant since the continuity of matched common subsequences may be destroyed easily by the unmatched states (for example, resulting as CLCS(S^4, S^1) = CLCS(S^4, S^2) = CLCS(S^4, S^3) = CLCS(S^4, S^5) = 0) or by the continuously duplicated common states (for example., resulting as CLCS(C^1, C^3) = 0), which in turn means that for real applications, it will be very sensitive to noise which will be taken as unmatched states in state-sequence matching.

Table 2.5 CLCS table between five example state-sequences

Similarity		S^1	S^2	S^3	S^4	S^5
CLCS (*th*=2)	S^1	4	3	0	0	2
	S^2	3	7	3	0	5
	S^3	0	3	8	0	4
	S^4	0	0	0	11	0
	S^5	2	5	8	0	7

Section 2.3.3 All Common Subsequence (ACS)

From above LCS-like measurements, we can see that only the longest common subsequence may not be sufficient to distinguish the difference (similarity) between state-sequences. It is necessary therefore to explore the information in the second longest common subsequence, the third longest common subsequence and so on.

Different from the CLCS, which discards the short continuously matched subsequence, the All Common Subsequence (ACS) [Wan2007] takes into account the information of the second, third, … longest subsequences by counting the number of all common subsequences. For instance, let us take the three state-sequences {*A*, *B*, *C*} = {*cbabca*, *bcabac*, *abcade*}. Obviously, LCS(*A*, *B*) = {*caba*} and LCS(*A*, *C*) = {*abca*}, therefore, LCS'(*A*, *B*) = LCS'(*A*, *C*) = 4, which means we cannot distinguish the state-sequence *B* and *C* when compared to *A*. The set of all common subsequences of *A* and *B* is (ignoring the empty sequence): {*a*, *aa*, *ab*, *aba*, *abc*, *ac*, *b*, *ba*, *baa*, *bab*, *baba*, *babc*, *bac*, *bb*, *bba*, *bbc*, *bc*, *bca*, *c*, *ca*, *caa*, *cab*, *caba*, *cabc*, *cac*, *cb*, *cba*, *cbac*, *cbc*, *cc*}. The set of all common subsequences of A and C is: {*a*, *aa*, *ab*, *aba*, *abc*, *abca*, *ac*, *aca*, *b*, *ba*, *bc*, *bca*, *c*, *ca*}. ACS(*A*, *B*) = 31, ACS(*A*, *C*) = 15, suggesting that state-sequence *B* is more similar to state-sequence *A* than to state-sequence *C*.

Theorem 2.1. Given two state-sequences, $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_n)$. $N(i, j)$ denotes the number of common subsequences of $(x_1, \ldots, x_i)$ and $(y_1, \ldots, y_j)$, i.e., the prefixes of sequences *X* and *Y* of lengths *i* and *j*. Then:

$$N(i,j)=\begin{cases}1, & \text{if } i \text{ or } j=0\\ N(i-1,j-1)\times 2, & \text{if } x_i=y_j\\ N(i-1,j)+N(i,j-1)-N(i-1,j-1), & \text{if } x_i\neq y_j\end{cases} \tag{2-9}$$

Consequently ACS(*X*, *Y*) = *N*(*m*, *n*).

Proof: Let *A*(*i*-1, *j*-1) be the set of all common subsequences between $(x_1, \ldots, x_{i-1})$ and $(y_1, \ldots, y_{j-1})$. So *N*(*i*-1, *j*-1)=|*A*(*i*-1, *j*-1)|. If $x_i = y_j$, then $A(i, j) = A(i-1, j-1) \cup A(i-1, j-1)x_i$. where $A(i-1, j-1)x_i = \{ax_i \mid \forall a \in A(i-1, j-1)\}$. Therefore $N(i, j) = N(i-1, j-1) \times 2$. If $x_i \neq y_j$, then some new common subsequences may be added to $A(i, j-1)$ or $A(i-1, j)$ on top of $A(i-1, j-1)$. Therefore, $A(i, j) = A(i, j-1) \cup A(i-1, j) - A(i-1, j-1)$. Consequently we have $N(i, j) = N(i, j-1) + N(i-1, j) - N(i-1, j-1)$.

Algorithm 2.1: Calculation of all common subsequence

Input: Two sequences X and Y.

Output: The number of all common subsequences ACS(X, Y).

for $i = 0$ to $|X|$ do $N(i, 0) = 1$

for $j = 0$ to $|Y|$ do $N(0, j) = 1$

for $i = 1$ to $|X|$ do

  for $j = 1$ to $|Y|$ do

    if $x_i = y_j$ then

      $N(i, j) = N(i-1, j-1) \times 2$

    else

      $N(i, j) = N(i-1, j) + N(i, j-1) - N(i-1, j-1)$

    end

    end

  ACS(X, Y) = $N(|X|, |Y|)$

End

Example evolution:

For the same five state-sequences: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*]. Examples of ACS calculation are evaluated in the following tables.

Table 2.6 Example evolution of ACS

(a) ACS(S^1, S^1) table

ACS(S^1, S^1)	Ø	*a*	*b*	*c*	*d*
Ø	0	0	0	0	0
a	0	1	1	1	1
b	0	1	3	3	3
c	0	1	3	7	7
d	0	1	3	7	**15**

(b) ACS(S^1, S^2) table

ACS(S^1, S^2)	Ø	*a*	*a*	*a*	*a*	*a*	*b*	*c*
Ø	0	0	0	0	0	0	0	0
a	0	**1**	1	1	1	1	1	1
b	0	1	1	1	**1**	1	3	3
c	0	1	1	1	1	1	3	**7**
d	0	1	1	1	1	1	3	7

(c) ACS(S^1, S^3) table

ACS(S^1, S^3)	Ø	*a*	*a*	*b*	*b*	*c*	*c*	*d*	*d*
Ø	0	0	0	0	0	0	0	0	0
a	0	**1**	1	1	1	1	1	1	1
b	0	1	1	**3**	3	3	3	3	3
c	0	1	1	3	3	**7**	7	7	7
d	0	1	1	3	3	7	7	**15**	**15**

(d) ACS(S^1, S^4) table

ACS(S_1, S_4)	Ø	*a*	*a*	*e*	*b*	*b*	*f*	*c*	*c*	*g*	*d*	*d*
Ø	0	0	0	0	0	0	0	0	0	0	0	0
a	0	**1**	1	1	1	1	1	1	1	1	1	1
b	0	1	1	1	**3**	3	3	3	3	3	3	3
c	0	1	1	1	3	3	2	**3**	7	7	7	7
d	0	1	1	1	3	3	2	7	7	7	**15**	**15**

(e) ACS(S^1, S^5) table

ACS(S_1, S_5)	Ø	*a*	*a*	*a*	*a*	*b*	*b*	*b*
Ø	0	0	0	0	0	0	0	0
a	0	**1**	1	1	1	1	1	1
b	0	1	1	1	**1**	3	3	3
c	0	1	1	1	1	3	3	**3**
d	0	1	1	1	1	3	3	3

Problem:

"Non-uniqueness" problem: this problem is ubiquitous in ACS as well. One will get ACS(S^1, S^1)=ACS(S^1, S^3)= ACS(S^1, S^4)=16.

"Unreasonable" problem: the unreasonable phenomenon ACS(S^2, S^5) > ACS(S^2, S^3) still exists. In particular, in ACS, the similarity becomes extremely large (such as S^3 and S^4) when continuously duplicated common states exist frequently in

state-sequences and this will therefore underestimate the high similarity between S^3 and S^1.

Table 2.7 ACS table between five example state-sequences

Similarity		S^1	S^2	S^3	S^4	S^5
ACS	S^1	15	7	15	15	3
	S^2	7	127	15	15	31
	S^3	15	15	255	255	15
	S^4	15	15	255	2047	15
	S^5	3	31	15	15	127

Section 2.3.4 Time-Warped LCS (T-WLCS)

If we consider two state-sequences '*aabbcc*' and '*abc*', the output of LCS'('*aabbcc*', '*abc*') would be 3 since LCS('*aabbcc*', '*abc*') = '*abc*'. What about '*adbecf*' and '*abc*'? The output of LCS'('*adbecf*', '*abc*') would also be 3 since LCS('*adbecf*', '*abc*') = '*abc*' as well. However, considering that in the first pair ('*aabbcc*' and '*abc*'), '*aabbcc*' is just the extension version of '*abc*', which should be considered as more similar to the second pair ('*adbecf*' and '*abc*'). The main reason is that in the first pair, the unmatched states ('*a*', '*b*', '*c*') are regarded and discarded in the same way as that in the second pair ('*d*', '*e*', '*f*') in LCS. In the spirit of the Dynamic Time Warping (DTW) [SC1978] algorithm, the Time-Warped LCS (T-WLCS)[GS2004] was proposed. The recurrence formula for T-WLCS is:

$$T-WLCS(X_i,Y_j)=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ \max[T-WLCS(X_i,Y_{j-1}),T-WLCS(X_{i-1},Y_j), & \text{if } i,j>0\\ T-WLCS(X_{i-1},Y_{j-1})]+1 & \text{and } x_i=y_j\\ \max[T-WLCS(X_i,Y_{j-1}),T-WLCS(X_{i-1},Y_j)] & \text{if } i,j>0\\ & \text{and } x_i\neq y_j\end{cases} \quad (2\text{-}10)$$

where *T-WLCS*(X_i, Y_j) denotes the maximum length of a time warped common subsequence (we name the common subsequence plus the continuously duplicated

common subsequences as time warped common subsequence, distinguishing from the traditional conception of common subsequence in the original LCS). The length of longest time-warped common subsequence can be read in *T-WLCS*(X_m, Y_n).

Example evolution:

For the same five state-sequences: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*]. The examples of T-WLCS calculation is evaluated in the following tables:

Table 2.8 Example evolution of T-WLCS

(a) T-WLCS (S^1, S^1) table

T-WLCS(S^1, S^1)	Ø	*a*	*b*	*c*	*d*
Ø	0	0	0	0	0
a	0	1	1	1	1
b	0	1	2	2	2
c	0	1	2	3	3
d	0	1	2	3	**4**

(b) T-WLCS (S^1, S^2) table

T-WLCS(S^1, S^2)	Ø	*a*	*a*	*a*	*a*	*a*	*b*	*c*
Ø	0	0	0	0	0	0	0	0
a	0	**1**	2	3	4	5	5	5
b	0	1	2	3	**4**	5	6	6
c	0	1	2	3	4	5	6	**7**
d	0	1	2	3	4	5	6	**7**

(c) T-WLCS (S^1, S^3) table

T-WLCS(S_1, S_3)	Ø	*a*	*a*	*b*	*b*	*c*	*c*	*d*	*d*
Ø	0	0	0	0	0	0	0	0	0
a	0	**1**	2	2	2	2	2	2	2
b	0	1	2	**3**	4	4	4	4	4
c	0	1	2	3	4	**5**	6	6	6
d	0	1	2	3	4	**5**	6	7	8

(d) T-WLCS (S^1, S^4) table

T-WLCS(S^1, S^4)	Ø	*a*	*a*	*e*	*b*	*b*	*f*	*c*	*c*	*g*	*d*	*d*
Ø	0	0	0	0	0	0	0	0	0	0	0	0
a	0	**1**	2	2	2	2	2	2	2	2	2	2
b	0	1	2	2	3	4	4	4	4	4	4	4
c	0	1	2	2	3	4	4	5	6	6	6	6
d	0	1	2	2	3	4	4	5	6	6	7	8

(e) T-WLCS (S^1, S^5) table

T-WLCS(S^1, S^5)	Ø	*a*	*a*	*a*	*a*	*b*	*b*	*b*
Ø	0	0	0	0	0	0	0	0
a	0	**1**	2	3	4	4	4	4
b	0	1	2	3	4	5	6	7
c	0	1	2	3	4	5	6	7
d	0	1	2	3	4	5	6	7

Problem:

For the same five state-sequences: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*].

"Non-uniqueness" problem: this problem is ubiquitous in T-WLCS as well. One will get T-WLCS(S^1, S^3)= T-WLCS(S^1, S^4)=8 and T-WLCS(S^1, S^2)= T-WLCS(S^1, S^5)=8.

"Unreasonable" problem: the unreasonable phenomenon T-WLCS(S^2, S^5) > T-WLCS(S^2, S^3) still exists. Even T-WLCS cannot guarantee that the query state-sequence has the highest similarity with itself: for instance, T-WLCS(S^1, S^1) < T-WLCS(S^1, S^2), T-WLCS(S^1, S^3), T-WLCS(S^1, S^4), T-WLCS(S^1, S^5), which means the state-sequence S^1 has least similarity to itself comparing with the other state-sequences S^2, S^3, S^4, S^5. Such a problem becomes absurd if, for instance, we have $S^{2'}$ = '*aaaaaaaaaaa*', which will lead to T-WLCS(S^1,$S^{2'}$) = 12 due to the unreasonable treatment of continuously duplicated common states.

Table 2.9 T-WLCS table between five example state-sequences

Similarity		S^1	S^2	S^3	S^4	S^5
T-WLCS	S^1	4	7	8	8	7
	S^2	7	11	10	10	11
	S^3	8	10	12	12	9
	S^4	8	10	12	15	9
	S^5	7	11	9	9	12

Section 2.4 ED-Based Subsequence Matching

The Edit Distance is a popular measurement for subsequence matching besides the longest common subsequence-based measurements. Various distance models based on Edit Distance have been developed over the past half century for state-sequence matching, including: Dynamic Time Warping (DTW) [SC1978]; Edit Distance [Lev1965] and its variants such as Edit Distance on Real Sequence (EDR) [CN2004]; Edit Distance with Real Penalty (ERP) [COO2005] and Time Warp Edit Distance (TWED) [Mar2008] etc. However, most of these existing distance models characterize temporal distance only in terms of the temporal order over the state-sequences, whereas other important temporal features such as the temporal gap between two adjacent states, and the temporal duration of each state itself have been neglected.

Section 2.4.1 Original Edit Distance (OED)

The edit distance between two state-sequences is defined as the cost of transforming one state-sequence into the other state-sequence using operations such as substitution, deletion, insertion, transposition and so on. Four examples are demonstrated as follows for transforming one word into another one.

1) "*night*"→ "*light*": substitute "*n*" in "*night*" with "*l*", obtain "*light*"

2) "*knight*" → "*night*": delete "*k*" at the beginning of "*knight*", obtain "*night*"

3) "*discover*" → "*discovery*": insert "*y*" into the end of "*discover*", obtain "*discovery*"

4) "*quiet*" → "*quite*": transpose "*et*" in "*quiet*" into "*te*", obtain "*quite*"

From the examples, we can also conclude that the deletion operation and insertion operation are reciprocally inversed: we can also transform "*night*" into "*knight*" by inserting "*k*" into the front of "*night*" or transform "*discovery*" into "*discover*" by deleting the last character "*y*" of "*discovery*".

There are many algorithms to calculate the Edit Distance, including: Hamming Distance, Levenshtein Distance, Damerau-Levenshtein Distance, Jaro-Winkler Distance and Ukkonen's Algorithm. The Levenshtein Distance, which is named after Vladimir Levenshtein from 1965, is a widely used specification of the Edit Distance that calculates the minimum number of operations of substitution, deletion and insertion. In most applications, Edit Distance is referred to as Levenshtein Distance. Therefore, we shall refer to the Levenshtein Distance as the original Edit Distance if not specified.

For the two state-sequences X and Y, the edit distance between them can be defined as the following recursion:

Definition 2.3: the Edit Distance of given state-sequences X and Y is

$$ED(X_i, Y_j) = \begin{cases} j & if\ i = 0 \\ i & if\ j = 0 \\ ED(X_{i-1}, Y_{j-1}) & if\ x_i = y_j \\ \min(ED(X_i, Y_{j-1}), ED(X_{i-1}, Y_j), \\ \qquad ED(X_{i-1}, Y_{j-1})) + 1 & if\ x_i \neq y_j \end{cases} \tag{2-11}$$

Therefore, the Edit Distance (ED) between X and Y can be read as ED(X_m, Y_n). What follows are further explanations of the recursion: the problem can be summarized as transforming X_i into Y_j using a minimum operations ED(X_i, Y_j). The procedure of edit distance can be illustrated as:

1) Setup rule: i = 0 or j = 0

- i=0 and j=0: the number of operations to transform one empty state-sequence into another empty state-sequence is zero.

- $i \neq 0$ and j=0: the length of the first state-sequence is non-zero. The operations transforming X_i into Y_0 is deleting i states in the first state-sequence X_i.

- i=0 and $j \neq 0$: the length of the second state-sequence is non-zero. The operations transforming X_0 into Y_j is inserting j states in the first state-sequence X_i.

In general, we can conclude that $ED(X_0, Y_j) = j$ and $ED(X_i, Y_0) = i$. Typically, $ED(X_0, Y_0) = 0$.

2) Matching rule: $x_i = y_j$ ($i \neq 0$ and $j \neq 0$)

In this case, the current two states match each other. Suppose the number of operations required to transform X_{i-1} into Y_{j-1} is $ED(X_{i-1}, Y_{j-1})$, so that no additional operations are required to transform X_i into Y_j. Therefore, $ED(X_i, Y_j) = ED(X_{i-1}, Y_{j-1})$.

3) Unmatching rule: $x_i \neq y_j$ ($i \neq 0$ and $j \neq 0$)

In this case, the current two states are not matched. There are three ways to transform the first state-sequence into the second state-sequence:

- Substitution: if we can transform X_i into Y_j by exchanging x_i for y_j, and the number of operations required to transform X_{i-1} to Y_{j-1} is $ED(X_{i-1}, Y_{j-1})$, then the total number of operations is $ED(X_{i-1}, Y_{j-1}) + 1$.
- Deletion: if we can transform X_i into Y_j by removing x_i at the end of X_i, and the number of operations required to transform X_{i-1} to Y_j is $ED(X_{i-1}, Y_j)$, then the total number of operations is $ED(X_{i-1}, Y_j) + 1$.

- Insertion: if we can transform X_i into Y_j by adding y_j at the end of X_i, and the number of operations required to transform X_i to Y_{j-1} is ED(X_i, Y_{j-1}), then the total number of operations is ED(X_i, Y_{j-1}) + 1.

Therefore, the number of operations required to transform X_i into Y_j is obviously the minimum of the above three sub-cases: ED(X_i, Y_j) = min(ED(X_i, Y_{j-1}) + 1, ED(X_{i-1}, Y_j) + 1, ED(X_{i-1}, Y_{j-1})+1) = min(ED(X_i, Y_{j-1}), ED(X_{i-1}, Y_j), ED(X_{i-1}, Y_{j-1})) + 1. In view of the above analysis, the following conclusion can be reached for the original ED:

- The non-temporal distance is not considered.
- For the temporal distance, only the temporal order is considered in terms of Dynamic Programming.
- It is a binary-value distance model. Therefore, it is not sensitive to outliers and noise, and therefore arguably.

Section 2.4.2 Edit Distance on Real sequence (EDR)

The original Edit Distance was designed for string sequence matching where states are presented in the form of characters. However, in many real-life applications, states are not characters. Therefore, more practical distance measurements are required. The Edit Distance on Real Sequence, as an important extension of original Edit Distance, has been shown to be effective with respect to real-life state-sequence matching. Distinguishing the character states in the original Edit Distance, the multi-dimensional state is referred to as a state vector. The matching between two multi-dimensional real-life states is first defined:

Definition 2.4. d-dimensional state vectors x_i and y_j from two state-sequence X and Y are matched if and only if $|x_{it} - y_{jt}| \leq \varepsilon$ for all $1 \leq t \leq d$, where ε is the matching threshold.

Definition 2.5. For two given state-sequence X_m and Y_n, the Edit Distance on Real sequence (EDR) between them is defined as the following recursion:

$$EDR(X_i, Y_j) = \begin{cases} j & if\ i = 0 \\ i & if\ j = 0 \\ \min\{EDR(X_{i-1}, Y_{j-1}) + \text{subcost}, & i \neq 0, and \\ \quad EDR(X_{i-1}, Y_j) + 1, EDR(X_i, Y_{j-1}) + 1\} & j \neq 0 \end{cases} \quad (2\text{-}12)$$

where *subcost* = 0 if x_i and y_j are matched and *subcost* = 1 if x_i and y_j are unmatched. The procedure of edit distance on real sequence can be illustrated as:

1) Setup rule: i = 0 or j = 0

Analogous to the original Edit Distance, we can conclude that EDR(X_0, Y_j) = j and EDR(X_i, Y_0) = i. Typically, EDR(X_0, Y_0) =0.

2) Edition rule: i ≠ 0 and j ≠ 0

We also consider the three ways to transform the first state-sequence into the second state-sequence:

- *Substitution*: In this case, we first need to justify whether the current two states match each other or not. if we can transform X_i into Y_j by exchanging x_i for y_j, and the number of operations required to transform X_{i-1} to Y_{j-1} is EDR(X_{i-1}, Y_{j-1}), then the total number of operations is EDR(X_{i-1}, Y_{j-1}) + *subcost*, where the *subcost* is specified in following two sub-cases:

  - $| x_{it} - y_{jt} | \leq \varepsilon$ for all $1 \leq t \leq d$: the current two state vectors are matched. According to the definition, *subcost* = 0, which means no additional operation is required to transform X_i into Y_j.

  - $| x_{it} - y_{jt} | > \varepsilon$ for some $1 \leq t \leq d$: the current two state vectors are unmatched. According to the definition, *subcost* = 1, which means the substitution operation is required to transform X_i into Y_j.

- *Deletion*: if we can transform X_i into Y_j by removing x_i at the end of X_i, and the number of operations required to transform X_{i-1} to Y_j is EDR(X_{i-1}, Y_j), then the total number of operations is EDR(X_{i-1}, Y_j) + 1.
- *Insertion*: if we can transform X_i into Y_j by adding y_j at the end of X_i, and the number of operations required to transform X_i to Y_{j-1} is EDR(X_i, Y_{j-1}), then the total number of operations is EDR(X_i, Y_{j-1}) + 1.

Therefore, the number of operations required to transform X_i into Y_j is obviously the minimum of the above three sub-cases: EDR(X_i, Y_j) = min(EDR(X_i, Y_{j-1}) + subcost, EDR(X_{i-1}, Y_j) + 1, EDR(X_{i-1}, Y_{j-1})+1).

- The non-temporal distance is not considered.
- For the temporal distance, only the temporal order is considered in terms of the Dynamic Programming.
- It is a binary-value distance model. Therefore, it is not sensitive to the outliers and noise, and therefore not realistic.

Section 2.4.3 Edit distance with Real Penalty (ERP)

As a binary-value model, EDR is robust for the outliers and noise but not realistic since the distance between two states is not refined. Edit distance with Real Penalty (ERP) was proposed as another important extension of the original Edit Distance from the point of view of real penalty, where the real distance between two states was counted instead of a simple 0 or 1 being given. The ERP copes with the local time shifting in terms of adding a gap g. for example, X = [1, 2], Y = [1, 3, 6], X may be aligned into [1, 2, *g*] for alignment purposes. Therefore, the cost of an insertion operation (or a deletion operation if swaping *X* and *Y*) can be regarded as the real distance between the current state (‘6’ in *Y*) and the gap g (normally specified as zero).

$$dist_{ERP}(x_i, y_j) = \begin{cases} |x_i - y_j| & substitution \\ |x_i - g| & deletion \\ |g - y_j| & inseartion \end{cases} \quad (2\text{-}13)$$

The recursion of Edit distance with Real Penalty can be defined as:

$$ERP(X_i, Y_j) = \begin{cases} \sum_1^j |y_j - g| & if\ i = 0 \\ \sum_1^i |x_i - g| & if\ j = 0 \\ \min\{ERP(X_{i-1}, Y_{j-1}) + dist_{ERP}(x_i, y_j), & \\ \quad ERP(X_{i-1}, Y_j) + dist_{ERP}(x_i, g), & otherwise \\ \quad ERP(X_i, Y_{j-1}) + dist_{ERP}(g, y_j)\} & \end{cases} \quad (2\text{-}14)$$

The procedure of Edit distance with Real Penalty can be illustrated as:

1) Setup rule: i = 0 or j = 0

- i=0 and j=0: the cost to transform one empty state-sequence into another empty state-sequence is zero.

- $i \neq 0$ and j=0: the length of the first state-sequence is non-zero. The operations transforming X_i into Y_0 is deleting i states in the first state-sequence X_i. Therefore the cost is the sum of the real distance between the first i states and the gap g.

- i=0 and $j \neq 0$: the length of the second state-sequence is non-zero. The operations transforming X_0 into Y_j is inserting j states into the first state-sequence X_i. Therefore the cost is the sum of the real distance between the first j states and the gap g.

In general, we can conclude that ERP(X_0, Y_j) = $\sum_1^j |y_j - g|$ and ERP(X_i, Y_0) = $\sum_1^i |x_i - g|$. Typically, ERP(X_0, Y_0) =0.

2) Edition rule: $i \neq 0$ and $j \neq 0$

There are three ways to transform the first state-sequence into the second state-sequence:

- *Substitution*: different from the EDR, we do not need to justify whether the current two states match each other or not. Since the real distance between the two current states reflects the matching cost of substitution. If we can transform X_i into Y_j by exchanging x_i for y_j, and the real cost of operations required to transform X_{i-1} to Y_{j-1} is ERP(X_{i-1}, Y_{j-1}), then the total cost of operations is ERP(X_{i-1}, Y_{j-1}) + $dist_{ERP}(x_i, y_j)$, where $dist_{ERP}(x_i, y_j) = | x_i - y_j |$ denotes the substitution cost between x_i and y_j.
- *Deletion*: if we can transform X_i into Y_j by removing x_i at the end of X_i, and the real cost of operations required to transform X_{i-1} to Y_j is ERP(X_{i-1}, Y_j), then the total cost of operations is ERP(X_{i-1}, Y_j) + $dist_{ERP}(x_i, g)$, where $dist_{ERP}(x_i, g) = | x_i - g |$ denotes the deletion cost of x_i.
- *Insertion*: if we can transform X_i into Y_j by adding y_j at the end of X_i, and the real cost of operations required to transform X_i to Y_{j-1} is ERP(X_i, Y_{j-1}), then the total cost of operations is ERP(X_i, Y_{j-1}) + $dist_{ERP}(g, y_j)$, where $dist_{ERP}(g, y_j) = | g - y_j |$ denotes the insertion cost of y_j.

Therefore, the number of operations required to transform X_i into Y_j is obviously the minimum of the above three sub-cases: ERP(X_i, Y_j) = min(ERP(X_{i-1}, Y_{j-1}) + $dist_{ERP}(x_i, y_j)$, ERP(X_{i-1}, Y_j) + $dist_{ERP}(x_i, g)$, ERP(X_i, Y_{j-1}) + $dist_{ERP}(g, y_j)$).

- The non-temporal distance is not considered.
- For the temporal distance, only the temporal order is considered in terms of the Dynamic Programming.
- It is a real-penalty distance model. Therefore, it is realistic, but not sensitive to the outliers and noise.

Section 2.4.4 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW), as the most important variant of original Edit Distance, is defined as a real-penalty distance model. Different from the previous real-penalty distance model (ERP), Dynamic Time Warping (DTW) copes with the time shifting by duplicating the previous state. For instance, using the same example as demonstrated in ERP: X = [1, 2], Y = [1, 3, 6], X may be aligned into [1, 2, 2] for alignment purpose in DTW. Therefore, the cost of the insertion operation (or deletion operation if swap X and Y) can be regarded as the real distance between the current state ('6' in Y) and the duplicated state ('2' in X).

$$DTW(X_i, Y_j) = \begin{cases} 0 & \text{if } i = j = 0 \\ \infty & \text{if } i = 0 \text{ or } j = 0 \\ dist_{DTW}(x_i, y_j) + \min\{DTW(X_{i-1}, Y_{j-1}), \\ \quad DTW(X_{i-1}, Y_j), DTW(X_i, Y_{j-1})\} & \text{otherwise} \end{cases} \quad (2\text{-}15)$$

where the $dist_{DTW}(x_i, y_j)$ is normally specified as the Lp Norm. For instance, $dist_{DTW}(x_i, y_j) = | x_i - y_j |$ for L1 Norm and $dist_{DTW}(x_i, y_j) = \sqrt{(x_i^2 - y_j^2)}$ for L2 Norm. The procedure of Dynamic Time Warping can be illustrated as:

1) Setup rule: i = 0 or j = 0

- i=0 and j=0: the cost of transforming one empty state-sequence into another empty state-sequence is zero. Therefore, $DTW(X_0, Y_0) = 0$.

- i=0 or j=0: $DTW(X_0, Y_j) = DTW(X_i, Y_0) = \infty$. The cost of transforming one empty state-sequence into another non-empty state-sequence is infinite.

2) Edition rule: i ≠ 0 and j ≠ 0

There are three ways to transform the first state-sequence into the second state-sequence:

- *Substitution*: similar to the ERP, the real distance between the two current states reflects the matching cost of substitution. If we can transform X_i into Y_j by exchanging x_i for y_j, and the real cost of operations required to transform X_{i-1} to Y_{j-1} is DTW(X_{i-1}, Y_{j-1}), then the total cost of operations is DTW(X_{i-1}, Y_{j-1}) + $\text{dist}_{\text{DTW}}(x_i, y_j)$, where $\text{dist}_{\text{DTW}}(x_i, y_j) = \text{dist}_{\text{Lp}}(x_i, y_j)$ denotes Lp Norm distance between the current two states x_i and y_j.

- *Deletion*: if we can transform X_i into Y_j by removing x_i at the end of X_i, according to the spirit of DTW, the state y_j will be duplicated for alignment purpose. Suppose the real cost of operations required to transform X_{i-1} to Y_j is DTW(X_{i-1}, Y_j), then the total cost of operations is DTW(X_{i-1}, Y_j) + $\text{dist}_{\text{DTW}}(x_i, y_j)$, where $\text{dist}_{\text{DTW}}(x_i, y_j)$ respects the real cost between the current state x_i and the duplicated state y_j.
- *Insertion*: if we can transform X_i into Y_j by adding y_j at the end of X_i, opposite to the deletion operation, the state x_i will be duplicated for alignment purposes. Suppose the real cost of the operations required to transform X_i to Y_{j-1} is DTW(X_i, Y_{j-1}), then the total cost of operations is DTW(X_i, Y_{j-1}) + $\text{dist}_{\text{DTW}}(x_i, y_j)$, where $\text{dist}_{\text{DTW}}(x_i, y_j)$ respects the real cost between duplicated state x_i and the current state y_j.

Therefore, the number of operations required to transform X_i into Y_j is obviously the minimum of the above three sub-cases: DTW(X_i, Y_j) = min(DTW(X_{i-1}, Y_{j-1}) + $\text{dist}_{\text{DTW}}(x_i, y_j)$, DTW($X_{i-1}$, Y_j) + $\text{dist}_{\text{DTW}}(x_i, y_j)$, + DTW($X_i$, Y_{j-1}) + $\text{dist}_{\text{DTW}}(x_i, y_j)$) = $\text{dist}_{\text{DTW}}(x_i, y_j)$ + min(DTW(X_{i-1}, Y_{j-1}), DTW(X_{i-1}, Y_j), DTW(X_i, Y_{j-1})).

- The non-temporal distance is not considered.
- For the temporal distance, only the temporal order is considered in terms of the Dynamic Programming.
- It is a real-penalty distance model. Therefore, it is realistic, but not sensitive to the outliers and noise.

Section 2.4.5 Time Warped Edit Distance (TWED)

In distance models explored so far, including the original ED, EDR, ERP and DTW, only the temporal order is taken into account in terms of dynamic programming. Marteau [Mar 2008] produced an elastic model named Time Warped Edit Distance (TWED), which takes into account the temporal gap difference in terms of the temporal index of states where time-series and sequences are expressed as lists (timestamps) in the form of $t_1, t_2, \ldots. t_n,$.

First, two domains S and T are defined for the binary $X_m = [x_1, \ldots, x_m] = [(s_1, t_{s_i}), \ldots, (s_m, t_{s_m})] \in S \times T$, where $S \subset R^d$ denotes the d dimensional space state vector and $T \subset R$ denotes the strictly increasing time-stamp variable. Therefore, for $x_i = (s_i, t_{s_i})$ and $x_j = (s_j, t_{s_j})$, $t_{s_i} > t_{s_j}$ whenever $i > j$. Λ denotes the null sample. For the current two states x_i and y_j, the operations of substitution, deletion and insertion can be defined as

$$\begin{cases} \Gamma(x_i \to y_j) & substitution \\ \Gamma(x_i \to \Lambda) & deletion \\ \Gamma(\Lambda \to y_j) & insertion \end{cases} \tag{2-16}$$

where Γ denotes the arbitrary cost function. The recursion is defined as

$$TWED(X_i, Y_j) = \begin{cases} 0 & if\ i = j = 0 \\ \infty & if\ i = 0\ or\ j = 0 \\ \min\{TWED(X_{i-1}, Y_{j-1}) + \Gamma(x_i \to y_j), & \\ \quad TWED(X_{i-1}, Y_j) + \Gamma(x_i \to \Lambda), & otherwise \\ \quad TWED(X_i, Y_{j-1}) + \Gamma(\Lambda \to y_j)\} & \end{cases} \tag{2-17}$$

In order to specify the cost function of the three operations (substitution, deletion and insertion), the graphical paradigm is introduced. For the convenience of illustration, the 1D time-series (as shown in y-axis) against the time-stamp (as shown in x-axis) is constructed for two state-sequences *X* and *Y*. The three operations transforming *X* into *Y* can be explained in terms of the graphical edit paradigm, as shown in the figure that

shortly follows (analogously, the current states are $x_i = (s_i, \ t_{s_i})$ and $y_j = (q_j, \ t_{q_j})$):

- *Substitution*: as shown in figure 2.1 (a), the substitution operation for the two current two states consists of adjusting x_i to y_j and adjusting x_{i-1} to y_{j-1} between two state-sequences. Suppose the matching cost of X_{i-1} and Y_{j-1} is TWED(X_{i-1},Y_{j-1}), therefore the additional cost for substitution is dist(x_i, y_j) and dist(x_{i-1}, y_{j-1}). Defining x_i as a binary in TWED, dist(x_i, y_j) sequentially consists of dist(s_i, q_j) and dist(t_{s_i}, t_{q_j}). Therefore, $\Gamma(x_i \rightarrow y_j)$ = dist(s_i, q_j) + dist(s_{i-1}, q_{j-1}) + dist(t_{s_i}, t_{q_j})+ dist($t_{s_{i-1}}$, $t_{q_{j-1}}$).
- *Deletion*: as shown in figure 2.1 (b), the deletion operation consists of adjusting x_i to x_{i-1} in the first state-sequence. No additional adjustment is required in the second state-sequence. Suppose the matching cost of X_{i-1} and Y_j is TWED(X_{i-1},Y_j), then the additional cost for insertion is dist(x_i, x_{i-1}). Defining x_i as a binary in TWED, dist(x_i, x_{i-1}) sequentially consists of dist(s_i, s_{i-1}) and dist(t_{s_i}, $t_{s_{i-1}}$). Therefore, $\Gamma(x_i \rightarrow \Lambda)$ = dist(s_i, s_{i-1}) + dist(t_{s_i}, $t_{s_{i-1}}$).
- *Insertion*: as shown in figure 2.1 (c), the insertion operation consists of adjusting x_i to x_{i-1} in the first state-sequence. No additional adjustment is required in the second state-sequence. Suppose the matching cost of X_i and Y_{j-1} is TWED(X_i, Y_{j-1}), then the additional cost for insertion is dist(y_j, y_{j-1}). Defining y_j as a binary in TWED, dist(y_j, y_{j-1}) sequentially consists of dist(q_j, q_{j-1}) and dist(t_{q_j}, $t_{q_{j-1}}$). Therefore, $\Gamma(\Lambda \rightarrow y_j)$ = dist(q_j, q_{j-1}) + dist(t_{q_j}, $t_{q_{j-1}}$).

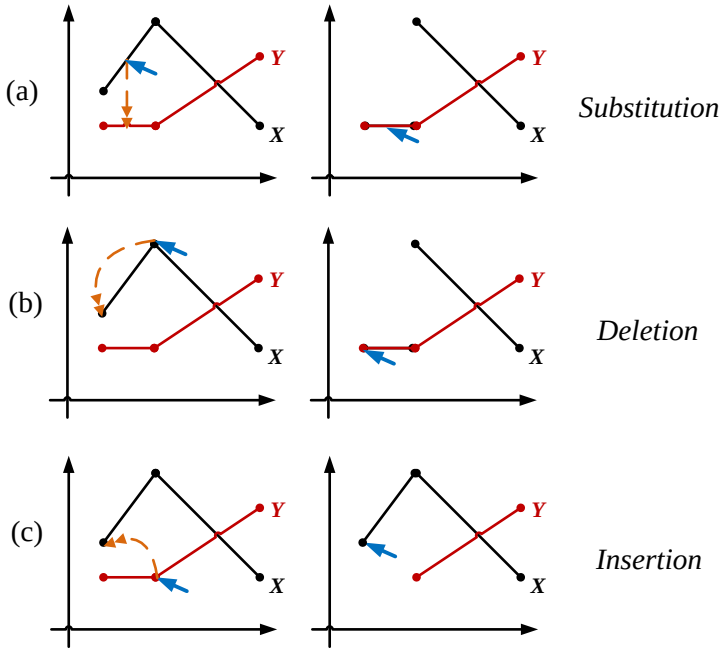


Figure 2.1 Graphical paradigm of TWED for edit cost function

This provides the basis for the TWED distance:

$$\begin{aligned}
&\Gamma(x_i \to y_j) = dist(x_i, y_j) + dist(x_{i-1}, y_{j-1}), && substitution \\
&\Gamma(x_i \to \Lambda) = dist(x_i, x_{i-1}) + \lambda, && deletion \\
&\Gamma(\Lambda \to y_j) = dist(y_j, y_{j-1}) + \lambda, && insertion
\end{aligned} \tag{2-18}$$

In summary, based on the literature review of the representation of primitive time and the conventional existing measurements for state-sequence matching, it can be noted that, firstly, the time structure in terms of both point and interval is the most reasonable to represent time-series, although it is necessary to formalize the characterization of time-series and state-sequence with respect to the rich temporal aspects including temporal order, temporal duration and temporal gap. Secondly, it is neccesary to design a new similarity measurement in order to conqure the main problems in the conventional existing measurements for state-sequence matching. Therefore, the general similarity measurement based on the formal characterization of time-series and state-sequence will be presentd in Chapter 3.

CHAPTER 3 GENERAL FRAMEWORK OF STATE-SEQUENCE MATCHING

Based on the review of the representation of time-series and state-sequences, as well as the existing similarity measurements for state-sequence matching, a general framework for state-sequence matching will be proposed. First, the formal characterization of time-series and state-sequences will be presented based on typed point-based intervals. Then, the general similarity measurement is designed to take into account both the non-temporal aspects and rich temporal aspects.

Section 3.1 Formal Characterization of Time-series and State-sequences

As mentioned in the introduction to this thesis, in most of the literature in the domain of data mining, the fundamental time theories upon which time-series and sequences are built up are not usually explicitly specified. Therefore, the formal characterizations with respect to the temporal basis were neglected. In this section, we shall present a formal characterization of time-series and state-sequences.

Section 3.1.1 Typed Point-based Time-elements and Time-series

In a system based solely on intervals as primitive, like that of Allen's interval temporal theory [All1984], or a system based on both points and intervals like that of Ma and Knight [MK1994], an "immediately before" relation can be directly expressed by the "Meets" relation.

N.B. The intuitive meaning of Meets(t_1, t_2) is that, on the one hand, t_1 and t_2 do not overlap each other (i.e., they do not have any part in common, not even a point); on the other hand, there is not any other time object standing between them.

For the sake of allowing the expression of both absolute time values and relative temporal relations, in this thesis, time-elements are defined as typed point-based intervals as shown in [MH2006]. The two different approaches to the treatment of intervals, i.e., taking intervals as primitive or as derived objects constructed out of primitive points, are actually reducible to logically equivalent expressions under some requisite interpretations. In fact, in a system based solely on points as primitives, say $(P, \leq)$, as the derived objects, an interval can be defined as a typed (left-open & right-open, left-closed & right-open, left-open & right-closed, left-closed & right-closed) subset of the set of primitive points, which must be in one of the following four forms [GS1999]:

$$(p_1, p_2) = \{p \mid p \in R \wedge p_1 < p < p_2\}$$

$$[p_1, p_2) = \{p \mid p \in R \wedge p_1 \leq p < p_2\}$$

$$(p_1, p_2] = \{p \mid p \in R \wedge p_1 < p \leq p_2\}$$

$$[p_1, p_2] = \{p \mid p \in R \wedge p_1 \leq p \leq p_2\}$$

In the above, R stands for the set of real numbers, and real numbers p and q are called the left-bound and right-bound of time-element t, respectively. The absolute values for the left and/or right bounds of some time-elements might be unknown. In this case, real number variables may be used for expressing relative relations to other time-elements (see later). If the left-bound and right-bound of time-element t are the same, t is called a time point; otherwise it is called a time interval. Without confusion, time-element $[p, p]$ is taken as identical to point p. Also, if a time-element is not specified as open or closed at its left (right) bound (that is, the left (right) type of the time-element is unknown), we shall use "<" (or ">") instead of "(" and "[" (or ")" and "]") as for its left (or right) bracket. In addition, the temporal duration of a time-element t, $T_{dur}(t)$, and the temporal gap between two adjacent elements $t_1$and t_2, T_{gap} (t_1, t_2) can be defined as below:

$$t = <p, q> \Leftrightarrow T_{dur}(t) = q - p$$

$$t_1 = <p_1, q_1> , t_2 = <p_2, q_2> \Leftrightarrow T_{gap}\ (t_1, t_2) = |p_2 - q_1|$$

Following Allen's terminology [All1984], we shall use "Meets" to denote the immediate predecessor order relation over time-elements, which can be formally defined as:

$$\text{Meets}(t_1, t_2) \Leftrightarrow \exists p_1,p,p_2 \in R(t_1 = (p_1, p) \wedge t_2 = [p, p_2)$$

$$\vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2)) \vee t_1 = (p_1, p) \wedge t_2 = [p, p_2]$$

$$\vee\, t_1 = [p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2)$$

$$\vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2) \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2]$$

$$\vee\, t_1 = [p_1, p] \wedge t_2 = (p, p_2])$$

It is easy to see that the intuitive meaning of Meets(t_1, t_2) is that, on the one hand, time-elements t_1 and t_2 do not overlap each other (i.e. they do not have any part in common, not even a point); on the other hand, there is no other time-element standing between them.

Analogous to the 13 relations introduced by Allen for intervals [All1984], there are 30 exclusive temporal order relations over time-elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations that relate points to points:

  {Equal, Before, After}

- Relations that relate points to intervals:

  {Before, After, Meets, Met_by, Starts, During. Finishes}

- Relations that relate intervals to points:

  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}

- Relations that relate intervals to intervals:

  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

We shall use a tetrad (T, R, D, G) to express the temporal reference of a given collection of temporal propositions, where:

- T = $\{t_1, \ldots, t_n\}$ is a finite set of time elements, expressing the knowledge (possibly incomplete) of what time elements are involved with respect to the given collection of propositions;

- R = $\{R^{(ij)} \mid R^{(ij)} = r^{(ij)}_1 \vee \ldots \vee r^{(ij)}_{m(ij)}, 1 \leq i, j \leq n; i \neq j\}$ is a collection of disjunctions of temporal relations over T, expressing the knowledge (possibly incomplete) as to how the time elements in T are related to each other. Here, $r^{(ij)}_k$ is one of the possible temporal relations as classified above.

- D is a collection of duration assignments (possibly incomplete) to every time element in T.

- G is the collection of temporal gap assignments to each adjacent pair of time elements in T.

The definition of these derived temporal order relations in terms of the single relation Meets is straightforward. For example:

$$\text{Before}(t_1, t_2) \Leftrightarrow \exists t \in T(\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$$

Based on such a time theory, a time-series T_n can be defined as a vector of time-elements temporally ordered one after another [MBZ2008]. Formally, a general time-series is defined in terms of the following schema:

GTS3.1) $T_n = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

GTS3.2) $R = [\text{Meets}(t_i, t_{i+1}) \vee \text{Before}(t_i, t_{i+1})]$, for all $i = 1, \ldots, n\text{-}1$

GTS3.3) $T_{dur} = [T_{dur}(t_i)] = [q_i - p_i]$, for some i where $1 \leq i \leq n$.

GTS3.4) $T_{gap} = [T_{gap}(t_i, t_{i+1})] = [p_{i+1} - q_i]$.for some i where $1 \leq i \leq n\text{-}1$.

Generally speaking, a time-series may be incomplete in various ways. For example, if the relation between t_j and t_{j+1} is "Before" rather than "Meets", it means that the knowledge about the time-element(s) between t_j and t_{j+1} is not available. In addition, if $T_{dur}(t_k)$ is missing for some k, it means that duration knowledge as for time-element t_k is unknown. Correspondingly, a complete time-series is defined in terms of the schema as below:

CTS3.1) $T = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

CTS3.2) $R = [\text{Meets}(t_i, t_{i+1})]$, for all $i = 1, \ldots, n\text{-}1]$.

CTS3.3) $T_{dur} = [T_{dur}(t_i)\ ] = [q_i - p_i]$, for all $i = 1, \ldots, n$.

CTS3.4) $T_{gap} = [T_{gap}(t_i, t_{i+1}) = 0]$, for all $i = 1, \ldots, n\text{-}1$.

Section 3.1.2 States and State-sequences

The validation of data is usually dependent on time. For instance, $1000 (account balance) can be valid before and on 1 January 2003 but become invalid afterwards. We shall use "fluents" to represent Boolean-valued, time-varying data, and denote proposition "fluent f holds true over time t" by formula Holds(f, t) :

(F1) $(f, t) \Rightarrow \forall t_1(\text{Part}(t_1, t) \Rightarrow \text{Holds}(f, t_1))$

That is, if fluent f holds true over a time-element t, then f holds true over any part of t.

(F2) $\forall t_1(\text{Part}(t_1, t) \Rightarrow \exists\, t_2\, (\text{Part}(t_2, t_1) \wedge \text{Holds}(f, t_2))) \Rightarrow \text{Holds}(f, t)$

That is, if any part of time t contains a part of itself over which fluent f holds true, then f holds true over t. Here,

$\text{Part}(t_1, t) \Leftrightarrow \text{Equal}(t_1, t) \vee \text{Starts}(t_1, t) \vee \text{During}(t_1, t) \vee \text{Finishes}(t_1, t)$

(F3) $\text{Holds}(f_1, t) \vee \text{Holds}(f_2, t) \Rightarrow \text{Holds}(f_1 \vee f_2, t)$

That is, if fluent f_1 or fluent f_2 holds true over time t, then at least one of them holds true over time t.

(F4) $\text{Holds}(\text{not}(f), t) \Leftrightarrow \forall t_1(\text{Part}(t_1, t) \Rightarrow \neg\text{Holds}(f, t_1))$

That is, the negation of fluent f holds true over time t if and only if fluent f does not hold true over any part of t.

(F5)$\text{Holds}(f, t_1) \wedge \text{Holds}(f, t_2) \wedge \text{Meets}(t_1, t_2) \Rightarrow \text{Holds}(f, t_1 \oplus t_2)$

That is, if fluent f holds true over two time-elements t_1 and t_2 that meet each other, then f holds over the ordered-union of t_1 and t_2.

A state is defined as a collection of fluents. Following the approach proposed in [MBZ2008], we shall use Belongs(f, s) to denote that fluent f that belongs to the collection of fluents representing state s. For the reason of simple expression, if $f_1, \ldots, f_m$ are all the fluents that belong to state s, we shall represent s as $<f_1, \ldots, f_m>$. Also, without confusion, we shall use formula Holds(s, t) to denote that s is the state of the world with respect to time t, provided that:

(F6) $s_1 = s_2 \Leftrightarrow \forall f\,(\text{Belongs}(f, s_1) \Leftrightarrow \text{Belongs}(f, s_2))$

That is, a state s holds true over time t if and only if every fluent in the s holds true over time t.

Consequently, a state-sequence S is defined as a list of states together with its corresponding time-series T_n. A general state-sequence is defined in terms of the schema as below:

GSS1) $S_n = [s_1, \ldots, s_n]$

GSS2) $H = [\text{Holds}(s_i, t_i)]$, for all $i = 1, \ldots, n$, where $[t_1, \ldots, t_n]$ is a time-series.

Correspondingly, a state-sequence is defined as complete if and only if the corresponding time-series is complete.

According to the basic set of axioms with respect to the point and interval based time-series theory [MH2006], for any two adjacent time elements t_1 and t_2 such that Meets(t_1, t_2), we can denote the ordered union of t_1 and t_2 as $t_1 \oplus t_2$. If Holds(s, t_1), Holds(s, t_2) , we have:

$$\text{Holds}(s, t_1 \oplus t_2)$$

$$T_{dur}(t_1 \oplus t_2) = T_{dur}(t_1) + T_{dur}(t_2)$$

That is, the "ordered union" operation over time elements is consistent with the conventional "addition" operation over the duration assignment function, i.e., 'T_{dur}'.

Section 3.2 State-based Subsequence matching

Subsequence matching is one of the most significant associations between state-sequences. First, we should note the differences between "substring" and "subsequence" which are often cited in computer science and mathematics. The notion of *string* is always regarded as a synonym for *sequence*, however, *substring* is different from *subsequence*.

- Substring: A *substring* of a string (sequence) $S = s_1 ... s_n$ can be represented as $\hat{S} = s_{1+i} ... s_{m+i}$, where $0 \le i$ and $m + i \le n$, which denotes the consecutive part of the string *S*.

- Subsequence: a *subsequence* of a sequence (string) $S = s_1 ... s_n$ can be represented as $\hat{S} = s_{i_1} ... s_{i_m}$ where $1 \le i_1 < i_2 < ... < i_m \le n$ or we can say the subsequence is exacted from a sequence along the same temporal order.

From the above definition, we can see that a substring of a string must be a subsequence of the string, rather than vice versa. For example, "ABCD" is a substring as well as a subsequence of "ABCDEFG", however, "ABD", which is a subsequence of "ABCDEFG", is not a substring of "ABCDEFG".

Section 3.2.1 Formal Characterization of State-sequence Matching

The notion of state is fundamental for many state-based applications; a state represents a static snapshot of the world of discourse, while the dynamic historical scenarios of the world can be characterized in terms of temporally ordered state-sequences. Generally speaking, a state-sequence presents a sequence of data, measured and/or spaced typically at successive times, which can be either points or intervals. State-sequence matching is a popular research topic in state-based systems and has been applied in various areas such as financial data analysis [WSZ2004], audio recognition [ZS2003], visual information retrieval [SSHZ2009], etc. Normally, state-sequence matching can be divided into two categories: whole matching [AFS1993, BKSS1990] (i.e., all state-sequences have the same length) and subsequence matching [AFS1993, MWL2001] (i.e., state-sequences have various lengths). Obviously, the whole matching problem is in fact a special case of the subsequence matching problem.

Followed by the formal tetrad characterization of state-sequence, the two state-sequences X_m and Y_n to be matched can be defined as:

GSSX1) $X_m = [x_1, \ldots, x_m]$

GSSX2) H = [Holds(x_i, t_i)], for all $i = 1, \ldots, m$,

where $[t_1, \ldots, t_m]$ is a time-series:

GTSX1) $T_m = [t_1, \ldots, t_m] = [<p_1, q_1>, \ldots, <p_m, q_m>]$

GTSX2) R= [Meets(t_i, t_{i+1})$\vee$Before(t_i, t_{i+1})], for all $i = 1, \ldots, m$-1

GTSX3) $T_{dur} = [d_i] = [T_{dur}(t_i)] = [q_i - p_i]$, for all $i = 1, \ldots, m$.

GTSX4) $T_{gap} = [g_i] = [T_{gap}(t_i, t_{i+1})] = [p_{i+1} - q_i]$ for all $i = 1, \ldots, m$-1and $g_0 = 0$.

Analogously:

GSSY1) $Y_n = [y_1, \ldots, y_n]$

GSSY2) H' = [Holds(y_j, $t_j^{'}$)], for all j = 1, …, n,

where $[t_1^{'},...,t_n^{'}]$ is a time-series:

GTSY1) $T_n^{'} = [t_1^{'},...,t_n^{'}] = [< p_1^{'}, q_1^{'} >,...,< p_n^{'}, q_n^{'} >]$

GTSY2) $R^{'} = [Meets(t_j^{'}, t_{j+1}^{'}) \vee Before(t_j^{'}, t_{j+1}^{'})]$ for all j = 1, …, n-1

GTSY3) $T_{dur}^{'} = [d_j^{'}] = [T_{dur}(t_j^{'})] = [q_j^{'} - p_j^{'}]$, for all j = 1, …, n.

GTSY4) $T_{gap}^{'} = [g_j^{'}] = [T_{gap}(t_j^{'}, t_{j+1}^{'})] = [p_{j+1}^{'} - q_j^{'}]$ for j = 1, …, n-1and $g_0^{'} = 0$.

Based on the tetrad representation of time-series and state-sequences, 3 temporal aspects should be taken into account: (i) Temporal Order (also known as temporal shifting tolerance, which has been taken into account by most ED-based similarity measurement approaches in the spirit of dynamic programming) (ii) Temporal Duration and (iii) the Temporal Gap, since they will vary the meanings of the state-sequences. For instance, the story (state-sequence) I (as S_I shown in Figure 3.1): "I ate for half an hour. After 1 hour, I walked out for 2 hours and then took a shower for half an hour".

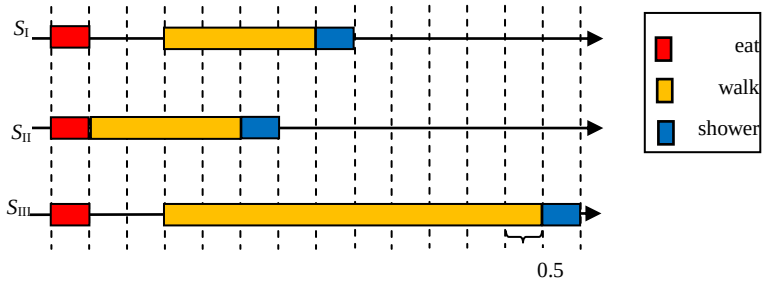


Figure 3.1 Temporal illustration of the three stories

The time-series can be described as below:

GSSI1) $S_I = [s_1, s_2, s_3]$

GSSI2) Holds(s_i, t_i), for all i = 1, 2, 3.

Where s_1, s_2, s_3 denote actions (states) "eat", "walk" and "shower" respectively and $[t_1, t_2, t_3]$ is its corresponding time-series described as a tetrad:

GTSI = (T, R, T_{dur},T_{gap}) with

GTSI1) $T = [t_1, t_2, t_3]$

GTSI2) R = [Before(t_1, t_2), Meets(t_2, t_3)]

GTSI3) T_{dur} = [$T_{dur}(t_1)$=0.5, $T_{dur}(t_2)$=2, $T_{dur}(t_3)$=0.5]

GTSI4) $T_{gap} = [T_{gap}(t_1, t_2), T_{gap}(t_2, t_3)] = [p_{21} - p_{12}, p_{31} - p_{22}] = [t_{12}, 0]$

And its corresponding complete description is (t_{12} denotes the time-element standing between t_1 and t_2) CTSI = (T, R, T_{dur},T_{gap}) with:

CTSI1) $T = [t_1, t_{12}, t_2, t_3]$

CTSI2) R = [Meets (t_1, t_{12}), Meets (t_{12}, t_2), Meets(t_2, t_3)]

CTSI3) $T_{dur} = [T_{dur}(t_1)=0.5, T_{dur}(t_{12})=1, T_{dur}(t_2)=2, T_{dur}(t_3)=0.5]$

CTSI4) $T_{gap} = [T_{gap}(t_1, t_2), T_{gap}(t_2, t_{12}), T_{gap}(t_2, t_3)] = [0, 0, 0]$

Let us think about story II as S_{II} shown in Figure 3.1: "I ate for half an hour. Then walked out for 2 hours and then took a shower for half an hour".

Obviously, the three states (events) have the same temporal order (t_1, t_2, t_3) in these two stories (state-sequences). However, the lengths of temporal gap standing between "ate" and "walked out" are different in the two stories (1 hour in story I and 0 in story II). In addition, for story III (as S_{III} shown in Figure 3.1): "I ate for half an hour. After 1 hour, I walked out for 5 hours and then took a shower for half an hour", where the lengths of the temporal gaps between each adjacent state pair are the same as those in story I. However, the duration of the state "walked out" is various. The statement "I walked out for 5 hours" in story III might be abnormal.

Section 3.2.2 General Framework for State-sequence Matching

Based on the formal characterization of time-series and state-sequences, the general similarity measurement with respect to the non-temporal information and the rich temporal information for two given state-sequences is defined as:

$$GSM(X_m, Y_n) = w_{ntem} Dis_{ntem}(X_m, Y_n) + w_{tem} Dis_{tem}(X_m, Y_n) \quad (3\text{-}1)$$

where $Dis_{ntem}(X_m, Y_n)$ and $Dis_{tem}(X_m, Y_n)$ denote the non-temporal distance and temporal distance, respectively with the corresponding weight w_{ntem} and w_{tem}.

Section 3.2.2.1 Non-temporal Matching

Non-temporal matching means common elemental state matching of the state-sequences X_m and Y_n, due to the fact that the elemental state appearing in the state-sequences are not actually ordered by their index, which in turn means the state-sequences are actually regarded as sets of states. It is a combinational problem to pair the two state-sequences in the first place. In general, for $m \geq n$, there are ${}^{m}Pr_n = m!/(m\text{-}n)!$ ways of pairing X_m and Y_n. Let Pr denote the set of all possible ordered vectors formed by selecting, in order, n random elemental states from X_m. It seems reasonable to take the pairing which gives the minimal overall distance. Hence, in this thesis, we shall define the non-temporal distance between X_m and Y_n as:

$$Dis_{ntem}(X_m, Y_n) = min_{pr \in Pr} dis_{ntem}(pr, Y_n) \quad (3\text{-}2)$$

where $dis_{ntem}(pr, Y_n) = \sqrt{\sum_{j=1}^{n} w_{jpr} dis_{jpr}(pr_j, y_j)^2} \Big/ \sqrt{\sum_{i=1}^{n} w_{ipr}}$, $pr = [pr_1, \ldots, pr_n]$ and.

Section 3.2.2.2 Temporal Matching

Based on the triad representation of state-sequences, the temporal measurement between two given state-sequences X_m with Y_n with respect to the 3 temporal aspects is

defined recursively as below:

$$Dis_{tem}(X_m, Y_n) = \min \begin{cases} Dis_{tem}(X_{m-1}, Y_n) + W_{del} \bullet Cost(x_m \to \phi) \\ Dis_{tem}(X_m, Y_{n-1}) + W_{ins} \bullet Cost(\phi \to y_n) \\ Dis_{tem}(X_{m-1}, Y_{n-1}) + W_{sub} \bullet Cost(x_m \to y_n) \end{cases} \quad (3\text{-}3)$$

where $m, n \geq 1$, $Cost(x_m \to \phi)$, $Cost(\phi \to y_n)$ and $Cost(x_m \to y_n)$ denote the cost function for edit operations *deletion*, *insertion* and *substitution*, respectively, and

$$Cost(a \to b) = \sum w_i \cdot Cost_i(a \to b), i = \{Tord, Tdur, Tgap\} \quad (3\text{-}4)$$

and $(a \to b) \in \{(x_m \to \phi), (\phi \to y_n), (x_m \to y_n)\}$.

The initialization is set as below:

$$\begin{aligned} Dis_{tem}(X_0, Y_0) &= 0 \\ Dis_{tem}(X_0, Y_j) &= \infty, \ for\ j \geq 1 \\ Dis_{tem}(X_i, Y_0) &= \infty, \ for\ i \geq 1 \end{aligned} \quad (3\text{-}5)$$

Section 3.2.3 General Definition of Cost Function

The cost function is a significant issue in similarity measurement. We have currently two categories: binary-value cost functions which are not sensitive to noise and real-penalty models which are more reasonable for real-life application but sensitive to noise since the operation cost with respect to a noise becomes much larger than normal states and will take the total cost into a much higher level. For instance, A_1^4 = [1, 2, 3, 4], B_1^4 = [1, 2, 5, 4], C_1^4 = [1, 2, 6, 4], D_1^4 = [1, 2, 1000, 4] (for the sake of convenience, we only consider the cost function of temporal order since the cost functions for temporal gap and temporal duration can be evaluated analogously). Assume that the states in any two state-sequences will be matched bi-objectively along a corresponding temporal order. Then in the binary-value models, $Cost_{Tord}(A_1^4, B_1^4)$ = $Cost_{Tord}$\{(1,1), (2,2), (3,5), (4,4)\}= $Cost_{Tord}$\{0, 0, 1, 0\}=1, $Cost_{Tord}(A_1^4, D_1^4)$ $Cost_{Tord}$\{(1,1), (2,2), (3,1000), (4,4)\} = $Cost_{Tord}$\{0, 0, 1, 0\}=1. So $Cost_{Tord}(A_1^4, B_1^4)$ = $Cost_{Tord}(A_1^4, D_1^4)$

which in turn means it is not sensitive to the noise (1000 in D_1^4) since the cost between all the unmatched state pairs (may include noise) is calculated as 1. Analogously, $Cost_{Tord}(A_1^4, B_1^4) = Cost_{Tord}(A_1^4, C_1^4) = 1$ which means it cannot distinguish the various values ($s_3 = 5$ in B_1^4 while 6 in C_1^4) in the domain of the states. In order to make up for this deficiency, the real-penalty cost function emerges, where the real distance between state pairs instead of the binary value (0/1) is accumulated. For instance, with respect to the real-penalty cost function, $Cost_{Tord}(A_1^4, B_1^4) = Cost_{Tord}\{(1,1), (2,2), (3,5), (4,4)\} = Cost_{Tord}\{0, 0, 2, 0\} = 2$ and $Cost_{Tord}(A_1^4, D_1^4)$ $Cost_{Tord}\{(1,1), (2,2), (3,6), (4,4)\} = Cost_{Tord}\{0, 0, 3, 0\} = 3$ (here the simplest one-dimension LP distance is employed for the real distance between each state pair). So $Cost_{Tord}(A_1^4, B_1^4) < Cost_{Tord}(A_1^4, C_1^4)$. Obviously, it is more reasonable than the binary-value cost function. However, $Cost_{Tord}(A_1^4, D_1^4)$ $Cost_{Tord}\{(1,1), (2,2), (3,1000), (4,4)\} = Cost_{Tord}\{0, 0, 997, 0\}=997 \gg Cost_{Tord}(A_1^4, B_1^4)$, even though they have just got one unmatched state pair, which means it is very sensitive to noise since the operation on state "1000" (with *insertion*, *deletion* or *substitution*) is much more expensive. Therefore, the problem of how to filter out the noise, or decrease its influence, should be taken into account in a real-penalty cost function. Unfortunately, none of the existing real-penalty distance models have considered it.

To filter out the noise or decrease its influence, a cost function is defined as:

$$Cost(a \to b) = \begin{cases} \sum_i w_i \cdot Cost_i(a \to b) & if\ Cost_i(a \to b) \le \delta \\ & for\ all\ i = \{Tord, Tgap, Tdur\} \\ c & else \end{cases} \quad (3\text{-}6)$$

Where $a, b \in \{x_i, y_j, \phi\}$ and c is a constant usually set to 0 (to filter out the noise) or the maximum cost that we have currently got (release the influence of the noise).

As for subsequence matching, *insertion* (or *deletion*) is required to align the two

state-sequences to be matched. It is important especially using the real-penalty cost function since the way of *insertion* (or *deletion*) will vary the cost value during matching. Reviewing the typical three real-penalty distance models ERP, DTW and TWED, the main difference is: when *insertion* (or *deletion*) is required to align state-sequence X_m and Y_n, ERP inserts a constant *g* (usually 0) into X_m while DTW duplicates the previous state in X_m and TWED duplicates the previous state in Y_n in terms of the graphical editor paradigm [12]. For instance, X_m = [1, 2], Y_n = [1, 3, 6], X_m may be aligned into [1, 2, _], [1, 2, 2] and [1, 2, 3] in ERP, DTW and TWED respectively. These different disposals will result in various costs for the *insertion*, *deletion* and *substitution* operations. We shall inherit the spirit of EDR and leave the task of how to adjust the importance of different operations to their corresponding weight W_{del}, W_{ins} and W_{sub}. Therefore, the cost functions of GSM are defined as below:

$$Cost_{Tord}(x_i \to y_j) = \begin{cases} dist_{Lp}(0, y_i) & if\ x_i = \phi \\ dist_{Lp}(x_j, 0) & if\ y_j = \phi \\ dist_{Lp}(x_i, y_j) & else \end{cases} \tag{3-7}$$

$$Cost_{Tdur}(x_i \to y_j) = \begin{cases} dist_{Lp}(0, d_j^{''}) & if\ d_i^{'} = 0 \\ dist_{Lp}(d_i^{'}, 0) & if\ d_j^{''} = 0 \\ dist_{Lp}(d_i^{'}, d_j^{''}) & else \end{cases} \tag{3-8}$$

$$Cost_{Tgap}(x_i \to y_j) = \begin{cases} dist_{Lp}(0, g_{j-1}^{'}) & if\ g_{i-1} = 0 \\ dist_{Lp}(g_{i-1}, 0) & if\ g_{j-1}^{'} = 0 \\ dist_{Lp}(g_{i-1}, g_{j-1}^{'}) & else \end{cases} \tag{3-9}$$

Where $i = 1, \ldots, m, j = 1, \ldots, n$.

In summary, the aspects considered in GSM compared with existing similarity measurements are exhibited in Table 3.1. GSM is the only similarity measurement that accounts for both the non-temporal aspects and rich temporal aspects. Meanwhile, it is also a reasonable real-penalty-style measurement and robust to noise.

Table 3.1 The aspects considered in similarity measurements

Aspects / Model	Non-temporal	Temporal Difference			Cost Function	
		Temporal Order	Temporal Duration	Temporal Gap	Anti-noise	Real Penalty
LCSS						✓
CLCS		✓			✓	
ACS		✓			✓	
T-WLCS		✓			✓	
OED		✓			✓	
EDR		✓			✓	
DTW		✓				✓
ERP		✓				✓
TWED		✓		✓		✓
GSM	✓	✓	✓	✓	✓	✓

All the non-temporal and temporal distances have been taken into account (as shown in table 3.2):

Table 3.2 General similarity measurement

Distance Aspects		Consideration
Non-temporal Aspect		Formula (3-2)
Temporal Aspect	Temporal Order	Formula (3-7)
	Temporal Gap	Formula (3-8)
	Temporal Duration	Formula (3-9)
Cost Function	Anti-noise	Formula (3-6)
	Real Penalty	Formula (3-6)

In summary, a formal characterization of time-series and state-sequence has been presented based on the typed point based interval. Benefitting from the formal consideration of temporal aspects (temporal order, temporal duration and temporal gap), a general similarity measurement named as GSM, which covers both non-temporal and all the three temporal aspects, has been designed for general state-sequence matching. In the next chapter, we shall demonstrate the generality of proposed GSM and examine the validity and effectiveness for state-sequence matching.

CHAPTER 4 GENERALIZATION AND APPLICATION OF GSM

Since the GSM proposed in chapter 3 addresses both the non-temporal aspects and all the 3 temporal aspects, it is versatile enough to subsume other existing similarity measurements in the literature of sequence matching. In fact, most of those existing measurements can be taken as special cases of GSM by means of specifying the non-temporal and temporal weights, and the cost functions, correspondingly. Meanwhile, to demonstrate the performance of the proposed GSM, experiments were conducted on 6 benchmark datasets.

Section 4.1 The Generalization of GSM

In this section, we shall analyse the powerful expressive ability of GSM by deducing the conventional existing measurements as its special cases.

Section 4.1.1 Original ED Special Case

Set the following restriction:

1) $w_{ntem} = 0, w_{tem} = 1$

2) $W_{del} = W_{ins} = W_{sub} = 1$

3) $w_{Tord} = w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = 1$, $Cost_{Tord}(x_i, y_j) = (x_i, y_j)$ with

$$Cost_{Trod}^{ED}(x_i, y_j) = \begin{cases} 0 & if\ x_i = y_j \\ 1 & else \end{cases}$$

Then we will get the recursion formulation of OED:

$$ED(X_i, Y_j) = \min \begin{cases} ED(X_{i-1}, Y_j) + 1 \\ ED(X_i, Y_{j-1}) + 1 \\ ED(X_{i-1}, Y_{j-1}) + Cost_{Tord}^{ED}(x_i, y_j) \end{cases} \quad (4\text{-}1)$$

which in turn means in OED:

1) Only the temporal order aspect has been accounted for

2) The three operations have the same status

3) No temporal gap or duration difference is taken into account

4) The cost function is binary-value

Section 4.1.2 EDR Special Case

Set the following restriction:

1) $w_{ntem} = 0, w_{tem} = 1$

2) $W_{del} = W_{ins} = W_{sub} = 1$

3) $w_{Tord} = 1, w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = 1$, $Cost_{Tord}(x_i, y_j) = Cost_{Trod}^{EDR}(x_i, y_j)$ with

$$Cost_{Trod}^{EDR}(x_i, y_j) = \begin{cases} 0 & if\ d_{LP}(x_i, y_j) \le \delta \\ 1 & else \end{cases}$$

where $d_{LP}(x_i, y_j)$ denotes the LP-Norm distance between x_i and y_j. Then we will get the formulation of EDR:

$$EDR(X_i, Y_j) = \min \begin{cases} EDR(X_{i-1}, Y_j) + 1 \\ EDR(X_i, Y_{j-1}) + 1 \\ EDR(X_{i-1}, Y_{j-1}) + Cost_{Tord}^{EDR}(x_i, y_j) \end{cases} \quad (4\text{-}2)$$

Similar to the basic ED, the cost function is binary value (0/1). In contrast, in order to be applied to real life data, EDR relaxes the matching equality by parameter δ since the strict equality in ED is limited to symbol (or string) matching.

Section 4.1.3 DTW Special Case

In the formula of *GDM*, set the following restriction:

1) $w_{ntem} = 0, w_{tem} = 1$

2) $W_{del} = W_{ins} = W_{sub} = 1$

3) $w_{Tord} = 1, w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = Cost_{Tord}(x_i, y_j) = d_{LP}(x_i, y_j)$

Then we will get the formulation of DTW:

$$DTW(X_i, Y_j) = d_{LP}(x_i, y_j) + \min \begin{cases} DTW(X_{i-1}, Y_j) \\ DTW(X_i, Y_{j-1}) \\ DTW(X_{i-1}, Y_{j-1}) \end{cases} \quad (4\text{-}3)$$

Comparing with the binary-value models like basic ED and EDR, DTW is a real-penalty model which takes real cost (computed with LP-Norm) for each operation and it duplicates the previous state when inserting or deleting.

Section 4.1.4 ERP Special Case

Set the following restriction:

1) $w_{ntem} = 0, w_{tem} = 1$

2) $W_{del} = W_{ins} = W_{sub} = 1$

3) $w_{Tord} = 1, w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = d_{LP}(x_i, g)$, $Cost_{Tord}(\phi, y_j) = d_{LP}(g, y_j)$,

$Cost_{Tord}(x_i, y_j) = d_{LP}(x_i, y_j)$

Then we will get the formulation of ERP:

$$ERP(X_i, Y_j) = \min \begin{cases} ERP(X_{i-1}, Y_j) + d_{LP}(x_i, g) \\ ERP(X_i, Y_{j-1}) + d_{LP}(g, y_j) \\ ERP(X_{i-1}, Y_{j-1}) + d_{LP}(x_i, y_j) \end{cases} \tag{4-4}$$

Distinguishing from DTW, ERP adds a constant g (usually set to 0) instead of duplicating the previous state when inserting or deleting.

Section 4.1.5 TWED Special Case

Set the following restriction:

1) $w_{ntem} = 0,\ w_{tem} = 1$

2) $W_{del} = W_{ins} = W_{sub} = 1$

3) $w_{Tord} = 1,\ w_{Tgap} = v,\quad w_{Tdur} = 0$

4) $$\begin{cases} Cost(x_i, \phi) = Cost_{Tord}^{TWED}(x_i, x_{i-1}) + v \cdot Cost_{Tgap}^{TWED}(x_i, x_{i-1}) + \lambda \\ Cost(\phi, y_j) = Cost_{Tord}^{TWED}(y_{j-1}, y_j) + v \cdot Cost_{Tgap}^{TWED}(y_{j-1}, y_j) + \lambda \\ Cost(x_i, y_j) = Cost_{Tord}^{TWED}(x_i, y_j) + Cost_{Tord}^{TWED}(x_i, y_j) + \\ \qquad v \cdot (Cost_{Tgap}^{TWED}(x_i, y_j) + Cost_{Tgap}^{TWED}(x_{i-1}, y_{j-1})) \end{cases}$$

with $$\begin{cases} Cost_{Tord}^{TWED}(a_i, b_j) = d_{LP}(a_i, b_j) \\ Cost_{Tgap}^{TWED}(a_i, b_j) = d_{LP}(i, j) \end{cases}$$

for $(a_i, b_j) \in \{(x_i, x_{i-1}), (y_{j-1}, y_j), (x_i, y_j), (x_{i-1}, y_{j-1})\}$

Then we will get the formulation of TWED

$$TWED(X_i, Y_j) = \min \begin{cases} TWED(X_{i-1}, Y_j) + Cost(x_i, \phi) \\ TWED(X_i, Y_{j-1}) + Cost(\phi, y_j) \\ TWED(X_{i-1}, Y_{j-1}) + Cost(x_i, y_j) \end{cases} \tag{4-5}$$

In TWED, the temporal gap difference is counted, but no duration difference has been taken into account. Meanwhile, based on the timestamp theory, the index value of the states are used to compute the temporal gap distance, where, for the corresponding

the tetrad (T, R, T_{dur}, T_{gap}) and triad (T', R', T'_{dur}, T'_{gap}) we have:

For time-series T_m = [t_1, …, t_m] and state-sequences X_m = [x_1, …, x_m] with H = [Holds(x_i, t_i)], for all i = 1, …, m:

1) T_m= [1, 2, …, m] = [<1, 1>, [<2, 2>, …, <m, m>]
2) R = [Before(t_i, t_{i+1})], for all i = 1, …, m-1
3) $T_{dur} = [d_i] = [q_i - p_i] = [0,...,0]$, for all i = 1, …, m.
4) $T_{gap} = [g_i] = [p_{i+1} - q_i] = [1,1,...,1]$ for all i = 1, …, m-1and $g_0 = 0$.

And for time-series $T'_n = [t'_1,...,t'_n]$ and state-sequences Y_n = [y_1, …, y_n] with H' = [Holds(y_j, t'_j)], for all j = 1, …, n:

1) T'_n = [1, 2, …, n] = [<1, 1>, [<2, 2>, …, <n, n>]
2) $R' = [Before(t'_j, t'_{j+1})]$ for all j = 1, …, n-1
3) $T'_{dur} = [d'_j] = [q'_j - p'_j] = [0,...,0]$, for all j = 1, …, n.
4) $T'_{gap} = [g'_j] = [p'_{j+1} - q'_j] = [1,1,...1]$ for j = 1, …, n-1and $g'_0 = 0$.

Section 4.1.6 LCSS Special Case

Distinguishing from other models, LCSS considers the matched states to describe the similarity (inverse to the distance used in ED based models). So the *min* is replaced by *max* in LCSS and the initialization should be changed into a minimum value 0 correspondingly. The multi-dimensional LCSS uses δ to control the matching in time that can be regarded as the temporal gap range when duration function equals to 0 and the temporal relationship between each two adjacent states is only "before". N.B. the temporal gap is just used to restrict the matching range in time. No cost on temporal gap difference is counted (set 3)).

In LCSS,

1) $w_{ntem} = 0$, $w_{tem} = 1$

2) $W_{del} = W_{ins} = 0,\ W_{sub} = 1$

3) $w_{Tord} = 1,\ w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = 0,\ Cost_{Tord}(x_i, y_j) = Cost_{Tord}^{LCSS}(x_i, y_j)$

Then we will get the formulation of LCSS:

$$LCSS(X_i, Y_j) = \max \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ LCSS(X_i, Y_{j-1}) \\ LCSS(X_{i-1}, Y_j) \\ LCSS(X_{i-1}, Y_{j-1}) + Cost_{Tord}^{LCSS}(x_i, y_j) \end{cases} \tag{4-7}$$

where:

$$Cost_{Tord}^{LCSS}(x_i, y_j) = \begin{cases} 1 & if \begin{cases} Cost_{Tord}(x_i, y_j) \le \varepsilon \\ Cost_{Tgap}(x_i, y_j) \le \delta \end{cases} \\ 0 & else \end{cases} \tag{4-8}$$

Where, ε and δ are employed to control the matching in space and time.

Section 4.1.7 CLCS Special Case

As reviewed in chapter 2, CLCS is the further disposal of LCSS; therefore it has the same setting as LCSS.

1) $w_{ntem} = 0,\ w_{tem} = 1$

2) $W_{del} = W_{ins} = 0,\ W_{sub} = 1$

3) $w_{Tord} = 1,\ w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = 0,\ Cost_{Tord}(x_i, y_j) = Cost_{Tord}^{CLCS}(x_i, y_j)$

Despite the length of the longest common subsequence, the real common subsequence is also recorded according to formula (2-2), and then the CLCS can be calculated with formulas (2-5) to (2-8).

Section 4.1.8 ACS Special Case

Similar to LCSS, we set:

1) $w_{ntem} = 0$, $w_{tem} = 1$

2) $W_{del} = W_{ins} = 0$, $W_{sub} = 1$

3) $w_{Tord} = 1$, $w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j) = 0$, $Cost_{Tord}(x_i, y_j) = Cost_{Tord}^{ACS}(x_i, y_j)$

When substituting the first 3 settings into formula (3-1) to (3-4), we can get:

$$ACS(X_i, Y_j) = ACS(X_{i-1}, Y_{j-1}) + Cost_{Tord}^{ACS}(x_i, y_j) \tag{4-9}$$

with:

$$Cost_{Tord}^{ACS}(x_i, y_j) = \begin{cases} ACS(X_{i-1}, Y_{j-1}) & if\ x_i = y_j \\ ACS(X_{i-1}, Y_j) + ACS(X_i, Y_{j-1}) & \\ \qquad - 2ACS(X_{i-1}, Y_{j-1}) & if\ x_i \neq y_j \end{cases} \tag{4-10}$$

Therefore:

$$ACS(X_i, Y_j) = \begin{cases} ACS(X_{i-1}, Y_{j-1}) \times 2 & if\ x_i = y_j \\ ACS(X_{i-1}, Y_j) + ACS(X_i, Y_{j-1}) & \\ \qquad - ACS(X_{i-1}, Y_{j-1}) & if\ x_i \neq y_j \end{cases} \tag{4-11}$$

Section 4.1.9 T-WLCS Special Case

As a LCS-based similarity measurement, the first three settings of the T-WLCS special case are the same as those in the LCSS special case but with different cost functions in the fourth setting which can be listed as:

1) $w_{ntem} = 0$, $w_{tem} = 1$

2) $W_{del} = W_{ins} = 0$, $W_{sub} = 1$

3) w_{Tord} =1, $w_{Tgap} = w_{Tdur} = 0$

4) $Cost_{Tord}(x_i, \phi) = Cost_{Tord}(\phi, y_j)$=0 , $Cost_{Tord}(x_i, y_j) = Cost_{Tord}^{T-WLCS}(x_i, y_j)$

We can define the formula of T-WLCS as:

$$T-WLCS(X_i, Y_j) = \max \begin{cases} T-WLCS(X_i, Y_{j-1}) \\ T-WLCS(X_{i-1}, Y_j) \\ T-WLCS(X_{i-1}, Y_{j-1}) + Cost_{Tord}^{T-WLCS}(x_i, y_j) \end{cases} \quad (4\text{-}12)$$

with:

$$Cost_{Tord}^{T-WLCS}(x_i, y_j) = \begin{cases} 1 & if\ x_i = y_j \\ 0 & if\ x_i \neq y_j \end{cases} \quad (4\text{-}13)$$

Therefore:

$$T-WLCS(X_i, Y_j) = \begin{cases} \max[T-WLCS(X_i, Y_{j-1}), T-WLCS(X_{i-1}, Y_j), & if\ x_i = y_j \\ \qquad T-WLCS(X_{i-1}, Y_{j-1})] + 1 & \\ \max[T-WLCS(X_i, Y_{j-1}), T-WLCS(X_{i-1}, Y_j)] & if\ x_i \neq y_j \end{cases} \quad (4\text{-}14)$$

Section 4.2 The Optimal Temporal Common Subsequence

In this section, so as to distinguish from the concept of common subsequence in conventional LCS, we define the *temporal common subsequence* of two state-sequences as the common subsequence where each state is different from its neighbour(s) (predecessor and successor):

Section 4.2.1 Definition of OTCS

Definition 4.1: Given two state-sequences, X=[x_1, x_2, …, x_m] and Y=[y_1, y_2,…, y_n], with time series TX=[tx_1, tx_2, …, tx_m] and TY=[ty_1, ty_2,…, ty_n], temporal common subsequence is defined as:

TCS(X, Y) = {[s_1, s_2,…, s_t] | s_1, s_2,…, s_t ∈{x_1, x_2, …, x_m}∩ { y_1, y_2,…, y_n } and 0 <

$ts_1 < ts_2 < \ldots < ts_t < \min(m, n)$ and $s_j \neq s_{j+1}$ for $j = 1,\ldots,t\text{-}1$.

That is to say, there are no continuous duplications of states in temporal common subsequence. Let us return to the examples in figure 1.2 and figure 1.3 in chapter 1: For instance, the longest common subsequence of B3 and C3 is '*aabbbcd*', while the temporal common subsequence of B3 and C3 is '*abcd*'. Correspondingly, the Optimal Temporal Common Subsequence (OTCS) is the one with the highest overall similarity integrated by the length of temporal common subsequence, the temporal duration difference and temporal gap difference, noted as $OTCS_L$, $OTCS_D$ and $OTCS_G$ respectively.

Section 4.2.2 The Two Properties of OTCS

The task is how to solve the OTCS problem for two arbitrary sequences, *X* and *Y*. First, let us explore the properties of the OTCS function: suppose the current state-sequences to be matched is $[x_1, \ldots, x_{i\text{-}1}, x_i]$ and $[y_1, \ldots, y_{j\text{-}1}, y_j]$

1) Matching rules: $x_i = y_j$

In this case, the current states are matched. In order to detect whether the matched states are the continuous duplicated states in the two state-sequences respectively, four situations should be considered:

i) Both of them are continuous duplicated states: $x_{i\text{-}1} = y_{j\text{-}1} = x_i = y_j$

According to the definition of OTCS, to find the temporal common subsequence, shorten each state-sequence by deleting the current state. The OTCS of the shortened state-sequences is equal to the OTCS of the current state-sequences since the continuously duplicated common state(s) will be regarded as the same temporal common state with different temporal durations in each state-sequence. This means $OTCS(X_i, Y_j) = OTCS_L(X_{i-1}, Y_{j-1})$. For example: *X* = '*aaaabb*', *Y* = '*aaeebbb*', $x_{i\text{-}1} = y_{j\text{-}1} = x_i = y_j$ = 'b',

OTCS(‘*aaaabb*’, ‘*aaeebbb*’) = OTCS(‘*aaaab*’, ‘*aaeebb*’) = ‘*ab*’.

ii) Neither of the current states is the continuously duplicated state, but have the same predecessor: $x_{i-1} = y_{j-1} \neq x_i = y_j$

In this case, the currently matched states can be regarded as the new temporal common state. So, shortening each state-sequence by deleting the current state, the OTCS of current state-sequences is equal to the OTCS of the shortened state-sequences appending the currently matched state. $OTCS(X_i, Y_j) = (OTCS_L(X_{i-1}, Y_{j-1}), x_i)$ or $(OTCS_L(X_{i-1}, Y_{j-1}), y_j)$. For example: X = ‘*aaaabbc*’, Y = ‘*aaeebbbc*’, ‘*b*’ = $x_{i-1} = y_{j-1} \neq x_i = y_j$ = ‘*c*’, OTCS(‘*aaaabbc*’, ‘*aaeebbbc*’) = (OTCS(‘*aaaabb*’, ‘*aaeebbb*’), ‘*c*’) = ‘*abc*’

iii) Either of the current states is the continuously duplicated states and obviously the two current states have different predecessors: ($x_{i-1} \neq y_{j-1}$) & (*either* x_{i-1} *or* $y_{j-1} = x_i = y_j$).

There are two sub-cases in this case: $x_{i-1} = x_i = y_j$ or $y_{j-1} = x_i = y_j$, if $x_{i-1} = x_i = y_j$, which means x_i is the continuously duplicated state, so shorten *X* by deleting x_i and the OTCS between the current *X* and *Y* is equal to the OTCS between the shortened *X* and current *Y*: $OTCS(X_i, Y_j) = OTCS(X_{i-1}, Y_j)$. For example: *X* = ‘*aaaabb*’, *Y* = ‘*aaeeb*’, OTCS(‘*aaaabb*’, ‘*aaeeb*’) = OTCS(‘*aaaab*’, ‘*aaeeb*’) = ‘*ab*’; else, $y_{j-1} = x_i = y_j$, which means y_j is the continuously duplicated state. In the same manner, shorten Y by deleting y_j and the OTCS between current *X* and *Y* is equal to the OTCS between the current *X* and shortened *Y*: $OTCS(X_i, Y_j) = OTCS(X_i, Y_{j-1})$. for example: *X* = ‘*aaaab*’, *Y* = ‘*aaeebbb*’, OTCS(‘*aaaab*’, ‘*aaeebbb*’) = OTCS(‘*aaaab*’, ‘*aaeebb*’) = OTCS(‘*aaaab*’, ‘*aaeeb*’) = ‘*ab*’. To summarize the two sub-cases, the OTCS can be calculated as: $OTCS_L(X_i, Y_j) = \max(OTCS_L(X_{i-1}, Y_j), OTCS_L(X_i, Y_{j-1}))$.

iv) Neither of the current states is the continuously duplicated state and have different predecessors: $x_{i-1} \neq y_{j-1} \neq x_i = y_j$

In this case, the currently matched states can be regarded as the new temporal common state. So, after shortening each state-sequence by deleting the current state, the OTCS of the current state-sequences is equal to the OTCS of the shortened state-sequences appending the currently matched state. $OTCS(X_i, Y_j) = (OTCS_L(X_{i-1}, Y_{j-1}), x_i)$ or $(OTCS_L(X_{i-1}, Y_{j-1}), y_j)$. For example: X = '*aaaabbc*', Y = '*aaeebbbfffc*', x_{i-1} = '*b*' ≠ y_{j-1} = '*f*' ≠ $x_i = y_j$ = '*c*', OTCS('*aaaabbc*', '*aaeebbbfffc*') = (OTCS('*aaaabb*', '*aaeebbbfff*'), '*c*') = '*abc*'.

2) Unmatching rules: $x_i \neq y_j$

This means the current states are not matched, and then the OTCS of X and Y is equal to the longer of $OTCS(X_i,Y_{j-1})$ and $LCS(X_{i-1},Y_j)$. To explain the procedure, we shall demonstrate it by dividing the situation into two cases:

i) The predecessor of the current state in the first state-sequence matches the current state in the second state-sequence: $x_{i-1} = y_j$. For example, X= '*aaaabbc*', Y= '*aaeebbb*', $x_{i-1} = y_j$ = '*b*', therefore, $OTCS(X_i, Y_j) = OTCS(X_{i-1}, Y_j)$

ii) The predecessor of the current state in the second state-sequence matches the current state in the first state-sequence: $x_i = y_{j-1}$. For example, X_i= 'aaaabb', Y_j= '*aaeebbbc*', $x_i = y_{j-1}$ = '*b*', therefore, $OTCS(X_i, Y_j) = OTCS(X_i, Y_{j-1})$

To summarize the two cases in the second property, $OTCS(X_i, Y_j)$ = $longer(OTCS(X_{i-1}, Y_j), OTCS(X_i, Y_{j-1}))$.

Section 4.2.3 The Length of The OTCS by Dynamic Programming

According to the two properties of OTCS, the algorithm calculating the length of the optimal temporal common subsequence between state-sequences X_i and Y_j for all $1 \le i \le$ m and $1 \le j \le$ n can be illustrated as algorithm 4.1, where the length of OTCS will be stored in $OTCS_L(i, j)$ and $OTCS_L(m, n)$ returns the length of OTCS of X and Y.

Algorithm 4.1: The length of the OTCS

Input: two state-sequences X_m and Y_n.
Output: the length of the longest temporal common subsequences $OTCS_L(X_m, Y_n)$.

1) **Initiation**: $x_0 = y_0 = null$
   for i = 0 : m: $OTCS_L(i, 0) = 0$
   for j = 0 : n: $OTCS_L(0, j) = 0$
2) **Recursion**:
   for i = 1: m
   for j = 1 : n
   if $x_i = y_j$ # matched
   case 1: $x_{i-1} = y_{j-1} = x_i = y_j$
   $OTCS_L(i, j) = OTCS_L(i - 1, j - 1)$
   case 2: $x_{i-1} = y_{j-1} \neq x_i = y_j$
   $OTCS_L(i, j) = OTCS_L(i - 1, j - 1) + 1$
   case 3: $(x_{i-1} \neq y_{j-1})$ & (*either* x_{i-1} *or* $y_{j-1} = x_i = y_j$)
   $OTCS_L(i, j) = \max(OTCS_L(i-1, j), OTCS_L(i, j-1))$
   case 4: $x_{i-1} \neq y_{j-1} \neq x_i = y_j$
   $OTCS_L(i, j) = OTCS_L(i - 1, j - 1) + 1$
   else $x_i \neq y_j$ # unmatched
   $OTCS_L(i, j) = \max(OTCS_L(i-1, j), OTCS_L(i, j-1))$
3) **Accomplishment**
   $OTCS_L(X_m, Y_n) = OTCS_L(m, n)$

In algorithm 4.1, the continuously duplicated states are not re-counted as new common states in any state-sequence. For example, for the same five state-sequence: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*], the OTCS length is illustrated as table 4.1. For reasons of simple illustration, the temporal duration of each state is set as 1 and the temporal gap between each pair of adjacent states is set as 0 if they are identical, or 1 if they are different.

Table 4.1 OTCS length table

$OTCS_L$	*Ø*	*a*	*a*	*e*	*b*	*b*	*f*	*c*	*c*	*g*	*d*	*d*
Ø	0	0	0	0	0	0	0	0	0	0	0	0
a	0	1	1	1	1	1	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1	1	1	1
b	0	1	1	1	2	2	2	2	2	2	2	2
b	0	1	1	1	2	2	2	2	2	2	2	2
b	0	1	1	1	2	2	2	2	2	2	2	2
c	0	1	1	1	2	2	2	3	3	3	3	3
d	0	1	1	1	2	2	2	3	3	3	4	4

From table 4.1, we can see that the duplicated continuous states are regarded as once matching which means they are not re-counted as the length of common subsequence. For instance, $OTCS_L$(‘a’,’a’) = $OTCS_L$(‘aa’,’aa’) = 1, $OTCS_L$(‘aab’,’aaeb’) = $OTCS_L$(‘aabbb’,’aaebb’) = 2. Meanwhile, it is necessary to take into account the various duplicated continuous states besides the length (number) of common subsequence. In OTCS, the various duplications will be counted with various temporal durations correspondingly.

Section 4.2.4 The Temporal Duration and Temporal Gap by Backtracking

The distinguishing character of OTCS is that besides the length of the optimal common subsequence based on the definition 4.1, the temporal duration and temporal gap are also taken into account. In order to compute the differences of temporal duration and temporal gap, a backtracking technique is developed as shown in Algorithm 4.2:

Algorithm 4.2: Track back of OTCS

```
Function backTrack(OTCS_L[0..m,0..n], X[1..m], Y[1..n], i, j)
If i=0 or j=0
  return " "
Else if X(i)=Y(j)
        if X(i-1)= Y(j-1)
            return backTrack(OTCS_L, X, Y, i-1, j-1)
        else % X(i-1)≠ Y(j-1)
            if  X(i-1)= X(i) or Y(j-1)= Y(j)%one of the predecessor is equal to current state
                if OTCS_L(i-1,j) > OTCS_L(i,j-1)
                    return backTrack(OTCS_L, X, Y, i-1, j)
                else
                    return backTrack(OTCS_L, X, Y, i, j-1)
            else %none of the predecessor is equal to current state
                return backTrack(OTCS_L, X, Y, i-1, j-1) + X[i]
    else% X(i)≠Y(j)
        if OTCS_L(i-1,j) > OTCS_L(i,j-1)
            return backTrack(OTCS_L, X, Y, i-1, j);
        else
            return backTrack(OTCS_L, X, Y, i, j-1);
```

During the procedure of backtracking, we simultaneously record $Ind_k = (f_k, l_k)$ and $Ind'_k = (f'_k, l'_k)$ as the first and the last index of the k-th common state between X_m and Y_n, where $k = 1, \ldots, \mathrm{L} = \mathrm{OTCS_L}(X_m, Y_n)$, $f_k, l_k \in [1, m]$ and $f'_k, l'_k \in [1, n]$. According to the typed point-based intervals, the temporal duration difference $OTCS_D(X_m, Y_n)$ and temporal gap difference $OTCS_G(X_m, Y_n)$ are calculated as below:

$$OTCS_D(X_m, Y_n) = \sum_{k=1}^{L} \left| (q_{l_k} - p_{f_k}) - (q'_{l'_k} - p'_{f'_k}) \right| \tag{4-9}$$

$$OTCS_G(X_m, Y_n) = \begin{cases} 0 & if\ k = 1 \\ \sum_{k=2}^{L} \left| (p_{f_k} - q_{l_{k-1}}) - (p'_{f'_k} - q'_{l'_{k-1}}) \right| & else \end{cases} \tag{4-10}$$

Finally, the overall similarity with respect to the temporal order, temporal duration

and temporal gap is defined as:

$$\begin{aligned} OTCS(X_m, Y_n) = w_{Tord} \bullet OTCS_L(X_m, Y_n) \\ - w_{Tdur} \bullet OTCS_D(X_m, Y_n) - w_{Tgap} \bullet OTCS_G(X_m, Y_n) \end{aligned} \tag{4-11}$$

Example evolution:

First, let us show an example of OTCS. Figure 4.1 presents the OTCS table for state-sequences *X*=[*aabbccdddd*] and *Y*=[*bbaaeebbbfccccedd*], where the elements in the table denote the length of the OTCS obtained by the algorithm and the first and the last indices of the temporal common states are circled in red and green respectively. For instance, for the first common state '*a*', $Ind_k = (1,2)$, $Ind_k^{'} = (3,4)$, which means it starts from the first state and ends at the second state in the first state-sequence *X*, whilst it starts from the third state and ends by the forth state in the second state-sequence *Y*.

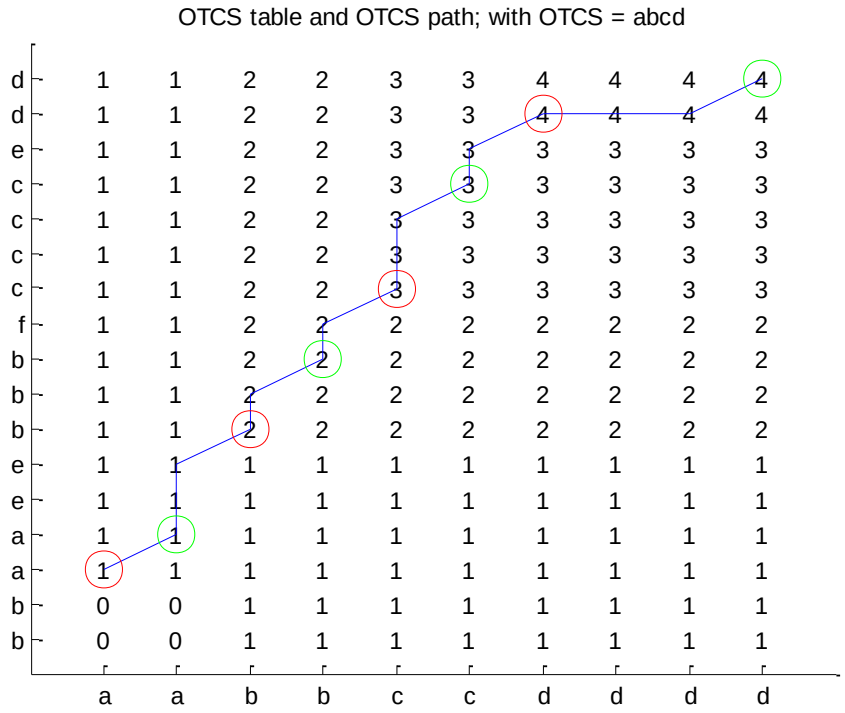


Figure 4.1 OTCS table and OTCS path with OTCS =abcd

For the same five state-sequences: S^1 = [*abcd*], S^2 = [*aaaaabc*], S^3 = [*aabbccdd*], S^4 = [*aaebbfccgdd*] and S^5 = [*aaaabbb*]. The examples of OTCS calculation are evaluated in the following five tables in table 4.2. In order to clearly see the difference from the point of view of temporal order, temporal duration and temporal gap individually, the result of OTCS is shown by a triad that denotes the $OTCS_L$, $OTCS_D$ and $OTCS_G$ respectively. From which we can see that we can distinguish the common subsequence with the same length by further comparison of the differences of temporal duration and temporal gap.

Table 4.2 Example evolution of OTCS

(a) OTCS(S^1, S^1) table

OTCS(S^1, S^1)	Ø	*a*	*b*	*c*	*d*
Ø	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
a	[0,0,0]	[1,0,0]	[1,0,0]	[1,0,0]	[1,0,0]
b	[0,0,0]	[1,0,0]	[2,0,0]	[2,0,0]	[2,0,0]
c	[0,0,0]	[1,0,0]	[2,0,0]	[3,0,0]	[3,0,0]
d	[0,0,0]	[1,0,0]	[2,0,0]	[3,0,0]	[4,0,0]

(b) OTCS(S^1, S^2) table

OTCS(S^1, S^2)	Ø	*a*	*a*	*a*	*a*	*a*	*b*	*c*
Ø	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
a	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[1,4,0]	[1,4,0]	[1,4,0]
b	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[1,4,0]	[2,4,0]	[2,4,0]
c	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[1,4,0]	[2,4,0]	[3,4,0]
d	[0,0,0]	[1,0,0]	[2,1,0]	[2,2,0]	[2,3,0]	[2,4,0]	[2,4,0]	[3, 4, 0]

(c) OTCS(S^1, S^3) table

OTCS(S^1, S^3)	Ø	*a*	*a*	*b*	*b*	*c*	*c*	*d*	*d*
Ø	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
a	[0,0,0]	**[1**,0,0]	[1,1,0]	[1,1,0]	[1,1,0]	[1,1,0]	[1,1,0]	[1,1,0]	[1,1,0]
b	[0,0,0]	[1,0,0]	[1,1,0]	**[2**,1,0]	[2,2,0]	[2,2,0]	[2,3,0]	[2,3,0]	[2,3,0]
c	[0,0,0]	[1,0,0]	[1,1,0]	[2,1,0]	[2,2,0]	**[3**,2,0]	[3,3,0]	[3,3,0]	[3,3,0]
d	[0,0,0]	[1,0,0]	[1,1,0]	[2,1,0]	[2,2,0]	[3,2,0]	[3,3,0]	**[4**,3,0]	**[4**,4,0]

(d) OTCS(S^1, S^4) table

OTCS(S^1,S^4)	Ø	*a*	*a*	*e*	*b*	*b*	*f*	*c*	*c*	*g*	*d*	*d*
Ø	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]
a	[0, 0, 0]	[1, 0, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]	[1, 1, 0]
b	[0, 0, 0]	[1, 0, 0]	[1, 1, 0]	[1, 1, 0]	[2, 1, 2]	[2, 2, 2]	[2, 2, 2]	[2, 2, 2]	[2, 3, 2]	[2, 3, 2]	[2, 3, 2]	[2, 3, 2]
c	[0, 0, 0]	[1, 0, 0]	[1, 1, 0]	[1, 1, 0]	[2, 1, 2]	[2, 2, 2]	[2, 2, 2]	[3, 2, 4]	[3, 3, 4]	[3, 3, 4]	[3, 3, 4]	[3, 3, 4]
d	[0, 0, 0]	[1, 0, 0]	[1, 1, 0]	[1, 1, 0]	[2, 1, 2]	[2, 2, 2]	[2, 2, 2]	[3, 2, 4]	[3, 3, 4]	[3, 3, 4]	[4, 3, 6]	[4, 4, 6]

(e) OTCS(S^1, S^5) table

OTCS(S^1, S^5)	Ø	*a*	*a*	*a*	*a*	*b*	*b*	*b*
Ø	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
a	[0,0,0]	**[1**,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[1,3,0]	[1,3,0]	[1,3,0]
b	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	**[1**,3,0]	[2,3,0]	[2,4,0]	[2,5,0]
c	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[2,3,0]	[2,4,0]	**[2**,5,0]
d	[0,0,0]	[1,0,0]	[1,1,0]	[1,2,0]	[1,3,0]	[2,3,0]	[2,4,0]	[2,5,0]

Table 4.3 shows the matching results between each pair of state-sequences of the given five state-sequences. For instance, the length of the optimal common subsequence is identical between (S^1, S^1), (S^1, S^3) and (S^1, S^4) with $OTCS_L(S^1, S^1) = OTCS_L(S^1, S^3) = OTCS_L(S^1, S^4) = 4$. However, S^1 will be taken as the most similar state-sequence to S^1 itself since $OTCS_D(S^1, S^1) = 0 < OTCS_D(S^1, S^3)$ or $OTCS_D(S^1, S^4)$ and $OTCS_G(S^1, S^1) = 0 < OTCS_G(S^1, S^3)$ or $OTCS_G(S^1, S^4)$ which means S^1 has less temporal duration difference and temporal gap difference to S^1 itself than to S^3 or S^4. Furthermore, S^3 seems closer to S^1 than S^4 with less difference in temporal gap but the same length of optimal common subsequence and the same difference in temporal

duration. Thus, the similarity between S^1 and S^1 to S^5 can be ordered as: OTCS(S^1, S^1) > OTCS(S^1, S^3) > OTCS(S^1, S^4) > OTCS(S^1, S^2) > OTCS(S^1, S^5), which is reasonable.

Table 4.3 OTCS table between S^1 to S^5

Similarity		S^1	S^2	S^3	S^4	S^5
OTCS	S^1	[4, 0, 0]	[3, 4, 0]	[4, 4, 0]	[4, 4, 6]	[2, 5, 0]
	S^2	[3, 4, 0]	[3, 0, 0]	[3, 5, 0]	[3, 5, 4]	[2, 3, 0]
	S^3	[4, 4, 0]	[3, 5, 0]	[4, 0, 0]	[4, 0, 6]	[2, 3, 0]
	S^4	[4, 4, 6]	[3, 5, 4]	[4, 0, 6]	[7, 0, 0]	[2, 3, 2]
	S^5	[2, 5, 0]	[2, 3, 0]	[2, 3, 0]	[2, 3, 2]	[2, 0, 0]

Section 4.3 Experimental Results of Application of GSM

Section 4.3.1 Experiment Databases

To demonstrate the performance of the proposed GSM as well as OTCS, experiments were conducted on 6 benchmark datasets as elaborated in Table 4.4.

Table 4.4 Description of 6 benchmark datasets**.**

Dataset	**Sample**	**Dimension**	**Class**
AT&T face[1]	400	1024	40
USPS[2]	9298	256	10
MNIST[3]	1000	784	10
COIL20[4]	1440	1024	20
Isolet1[5]	1560	617	26
BinAlpha[6]	1014	320	26

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[2] http://www.gaussianprocess.org/gpml/data/

[3] http://yann.lecun.com/exdb/mnist/

[4] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[5] http://archive.ics.uci.edu/ml/datasets/ISOLET

[6] http://yann.lecun.com/exdb/mnist/

- **AT&T Faces** Dataset contains 400 different images of 40 distinct subjects with 10 images per subject. For some subjects, the images were taken at different times, varying the lighting, facial expression and facial details (glasses/no glasses). All images were taken against a dark homogeneous background with the subjects in an upright, frontal, position. We reshape each image into one vector.

- **USPS** Dataset is a handwritten digit database, 500 images (50 images for every digit) were selected for the reported experiments.

- **MNIST** Dataset is a handwritten digit database. Each image is centered (according to the center of mass of the pixel intensities) on a 28×28 grid. In our experiments, we randomly chose 1000 images (i.e. each digit has 100 images). We reshaped each image into one vector.

- **COIL20** Dataset contains 20 objects. Each image of the same object is taken at 5 degrees intervals as the object is rotated on a turntable, consequently each object has 72 images associated with it. The size of each image is 32×32 pixels, with 256 grey levels per pixel. Each image is represented by a 1024 dimensional vector.

- **Isolet1** Spoken Letter Recognition Dataset generated by 150 subjects announcing the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1, isolet2, isolet3, isolet4, and isolet5. The features include spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. In our experiment, we utilized subset isolet1 only.

- **BinAlpha** Dataset containing 26 hand-written alphabets. We selected 30 images for every alphabet. We reshaped each image into one vector.

Section 4.3.2 Construction of Temporal Duration and Temporal Gap

In order to demonstrate the effectiveness of our measurement, and to avoid destroying the well organised structure of the original data sets, we construct 10 different distributions for temporal duration and temporal gap. For each class of the 6

benchmark datasets, the distributions of the temporal duration and temporal gap were selected randomly from the following 10 distributions. Figure 4.2 shows one example for each of the 10 distributions of duration.

1) Normal distribution with mean 0.5 and standard deviation 1;

2) Quadratic distribution: $y = x(2+i/10)$;

3) Constant distribution: $y = i/100$;

4) Negative quadratic distribution: $y = (1-x)(2+i/10)$;

5) Circle distribution: $y = \sqrt{1-x^{(2+i/10)}}$;

6) Power distribution: $y = x^{1/(2+i/10)}$;

7) Cosine distribution: $y=-(\frac{1}{2}+\frac{i}{400})\sin(2\pi x)+\frac{1}{2}$;

8) Sine distribution: $y=(\frac{1}{2}+\frac{i}{400})\sin(2\pi x)+\frac{1}{2}$;

9) Step function: $y = \begin{cases} 1-i/400 & i < 50 \\ i/400 & else \end{cases}$;

10) Quadratic distribution: $y=(4-\frac{i}{50})\bullet(x-\frac{1}{2})^2+\frac{i}{200}$;

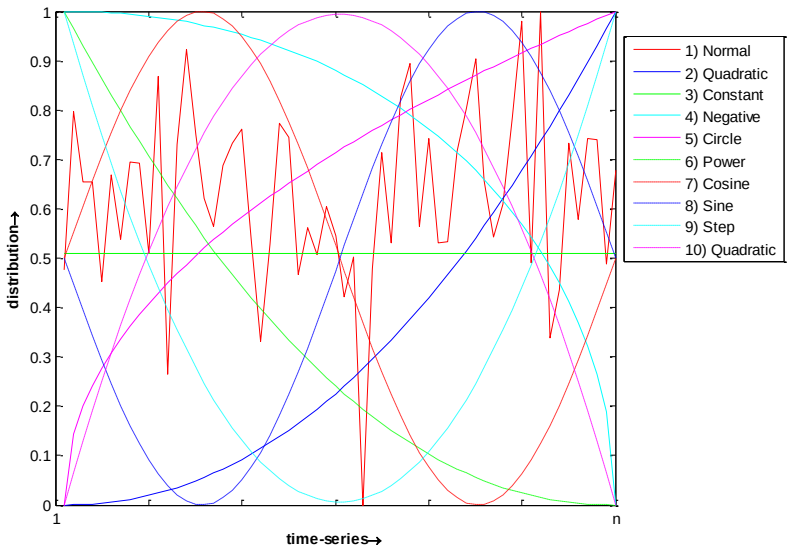


Figure 4.2 Distribution examples of temporal duration and temporal gap

Section 4.3.3 Contribution of Temporal Aspects in GSM

A K-means a clustering experiment was conducted to explore the weight contribution of temporal order, temporal duration and temporal gap. In order to highlight the contribution of temporal aspects, we first set $w_{ntem}=0$, $w_{tem}=1$. The clustering accuracies against temporal duration and temporal gap on 6 datasets are reported in figure 4.3 and figure 4.4. We set the weight of temporal order $w_{Tord}=1$, while the temporal duration and temporal gap w_{Tdur} and w_{Tgap} were varied as {1/256, 1/64, 1/16, 1/4, 1, 4, 16, 64}. Generally speaking, the temporal order contributes more significance than temporal duration and temporal gap. The temporal duration plays a slightly more significant role than temporal gap. The first 3 optimal weights for temporal duration and temporal gap are selected to construct the optimal combination of the temporal duration and temporal gap, and the clustering accuracies, are shown in figure 4.5 where the red circles denote the highest clustering accuracies and the corresponding weight combination is set as the final weight for temporal duration and temporal gap of the GSM on each dataset.

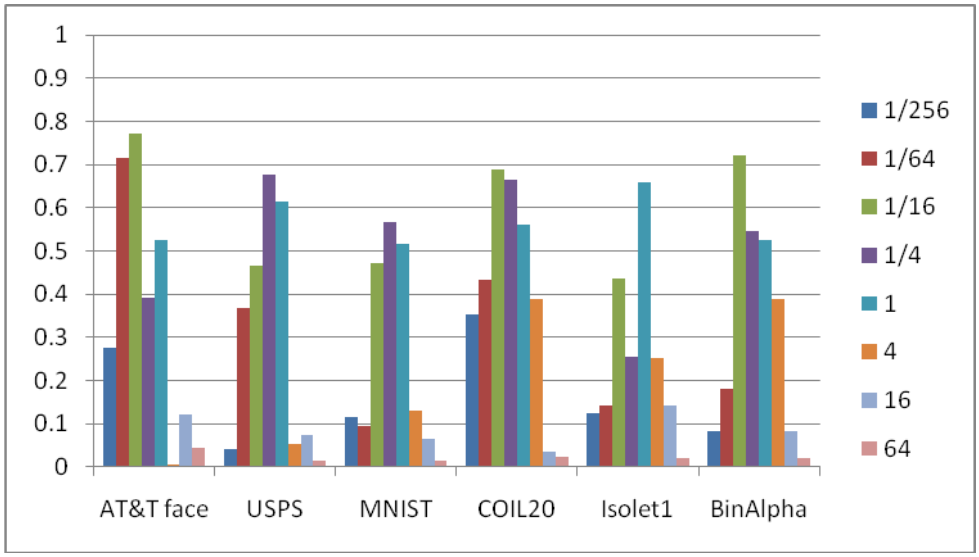


Figure 4.3 Weights contribution of temporal duration

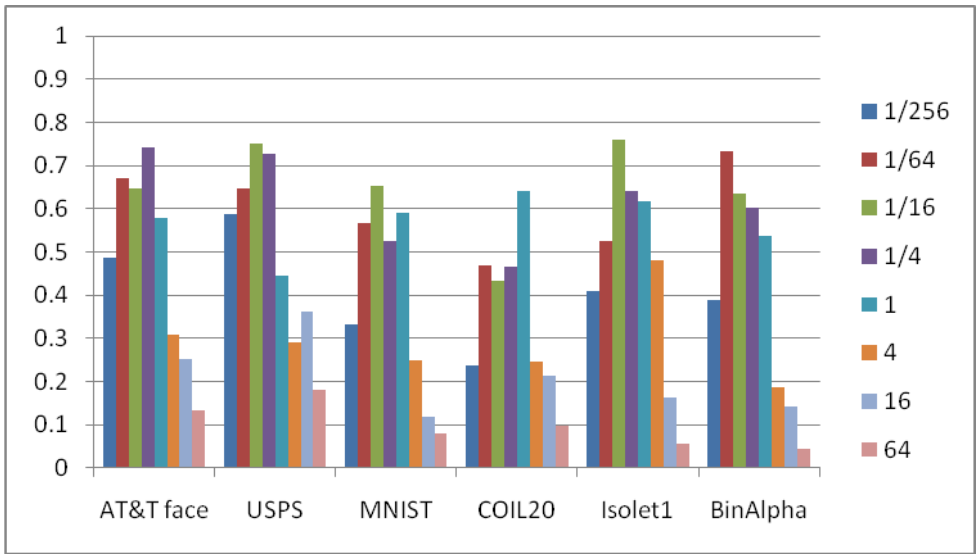


Figure 4.4 Weights contribution of temporal gap

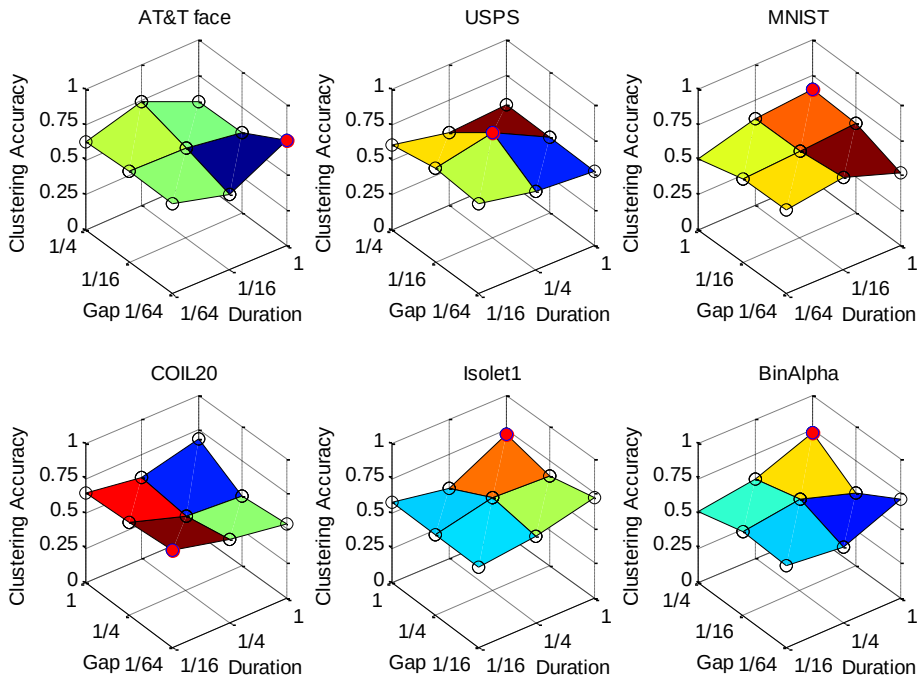
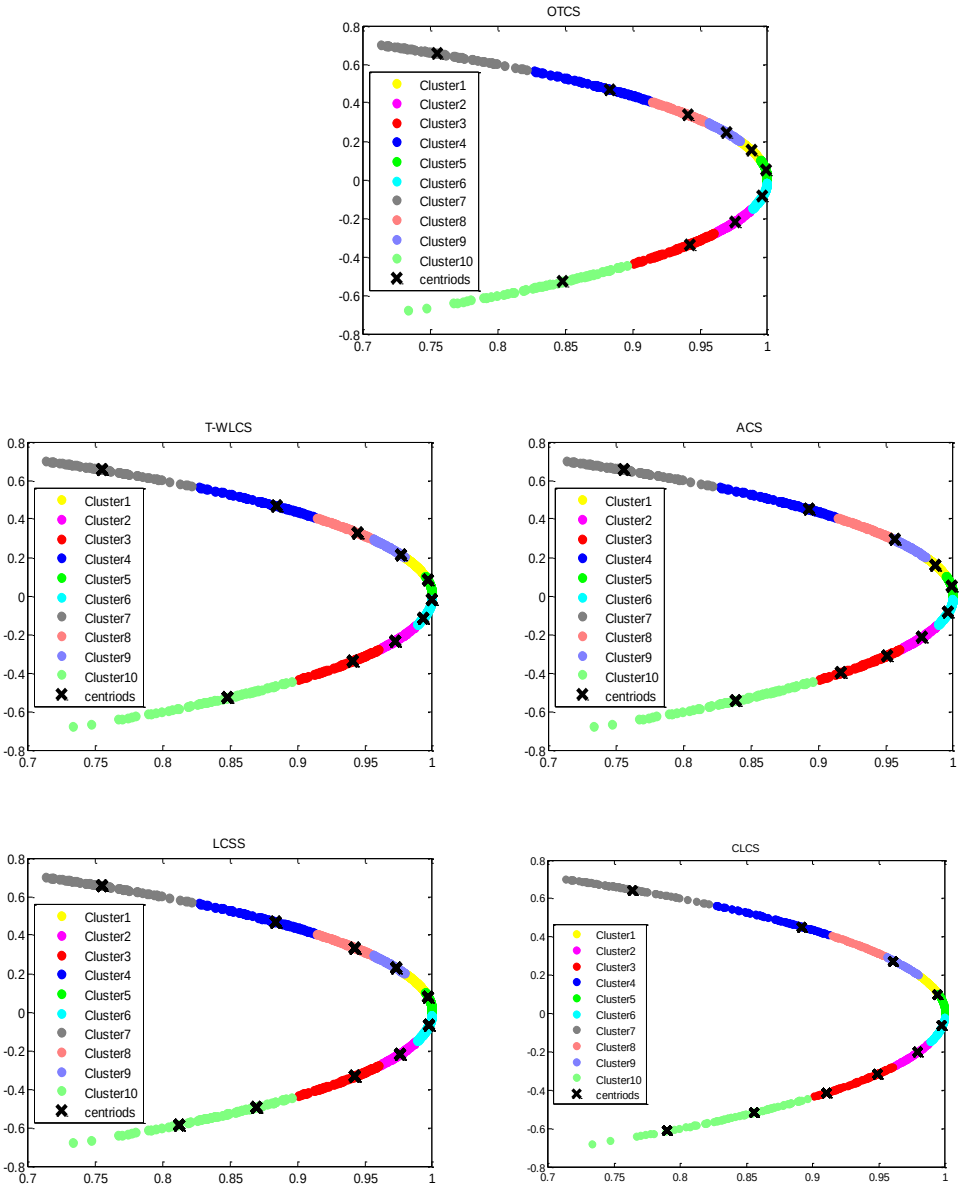


Figure 4.5 Optimal combination of temporal duration and temporal gap

Section 4.3.4 Comparison of GSM with Binary-value Measurements

In order to compare the performance of GSM with binary-value measurements OED, EDR, LCSS, CLCS, T-WLCS and ACS, the GSM was refined as OTCS with w_{ntem}

varying from $\{10^{-4}, 10^{-3}, \ldots 10^{4}\}$ and the optimal w_{ntem} that led to the best performance, while temporal duration and temporal gap were set as the optimal weight combination as shown in figure 4.5 ($w_{Tord} = 1$). Figure 4.6 shows an example of the clustering results on the MNIST dataset with OTCS compared to other binary-value measurements. The dimension was reduced to 2-dimension by PCA dimensionality reduction in order to plot the clustering results. From this we can see that OTCS has the best clustering results since the centroids are the most consistent to the data distribution.



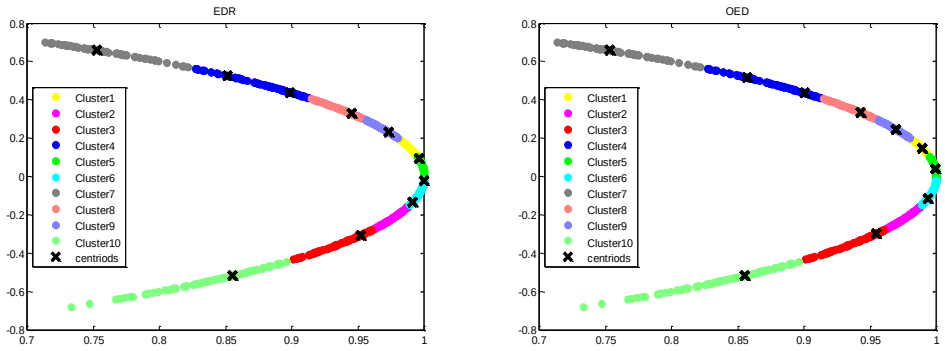


Figure 4.6 An example of clustering results on 2-d MNIST dataset

Table 4.5 shows the clustering accuracy of each dataset. Generally speaking, compared with the other reputable binary-value measurements, OTCS outperforms all of them with highest clustering accuracy, especially on the BinAlpha dataset.

Table 4.5 Clustering accuracy comparison of Binary-value measurements

Dataset / Measurement	AT&T face	USPS	MNIST	COIL20	Isolet1	BinAlpha
OED	65.39	60.50	54.95	59.84	65.85	68.96
EDR	76.92	66.87	66.31	61.28	70.49	71.32
LCSS	74.57	66.25	52.96	53.74	60.37	56.44
CLCS	60.23	57.64	50.35	51.87	55.24	53.49
ACS	75.84	73.85	55.66	60.55	64.85	60.55
T-WLCS	72.59	70.17	58.23	66.62	66.36	61.21
OTCS	**78.36**	**76.41**	**66.35**	**69.20**	**75.58**	**72.66**

Section 4.3.5 Comparison of GSM with Real-penalty Measurements

In comparison to real-penalty measurements such as ERP, DTW and TWED, the main advantage of GSM is that it is not sensitive to noise. In order to demonstrate the soundness of GSM, the noised datasets have been reconstructed by meanings of adding Gaussian noise with different means ([0, 0.2,…, 2]) and variances ([0.1, 0.2,…, 1]) to each dataset. Table 4.6 below shows the average mean and standard deviation (STD) of the retrieval precision on each noised dataset, which statistically demonstrates the

soundness of GSM with higher precision (mean) and smaller fluctuation (STD). Figure 4.7 illustrates the retrieval precision on the MNIST dataset in detail with respect to various mean and variance of Gaussian noise, which verifies the effectiveness of GSM visually.

Table 4.6 Statistic of the retrieval precision of noised dataset

Statistic \ Dataset		AT&T face	USPS	MNIST	COIL20	Isolet1	BinAlpha
ERP	***Mean***	63.71	65.60	59.48	61.53	74.66	71.25
	STD	0.1249	0.1391	0.1742	0.2519	0.1285	0.1595
DTW	***Mean***	73.37	72.29	65.79	73.11	78.51	74.29
	STD	0.1932	0.1128	0.1890	0.1438	0.0891	0.1032
TWED	***Mean***	79.95	75.30	68.80	72.96	79.38	76.90
	STD	0.0993	0.1025	0.1359	0.1235	0.0940	0.0895
GSM	***Mean***	**85.65**	**80.54**	**74.82**	**78.44**	**84.19**	**82.84**
	STD	**0.0632**	**0.0738**	**0.1022**	**0.0983**	**0.0593**	**0.738**

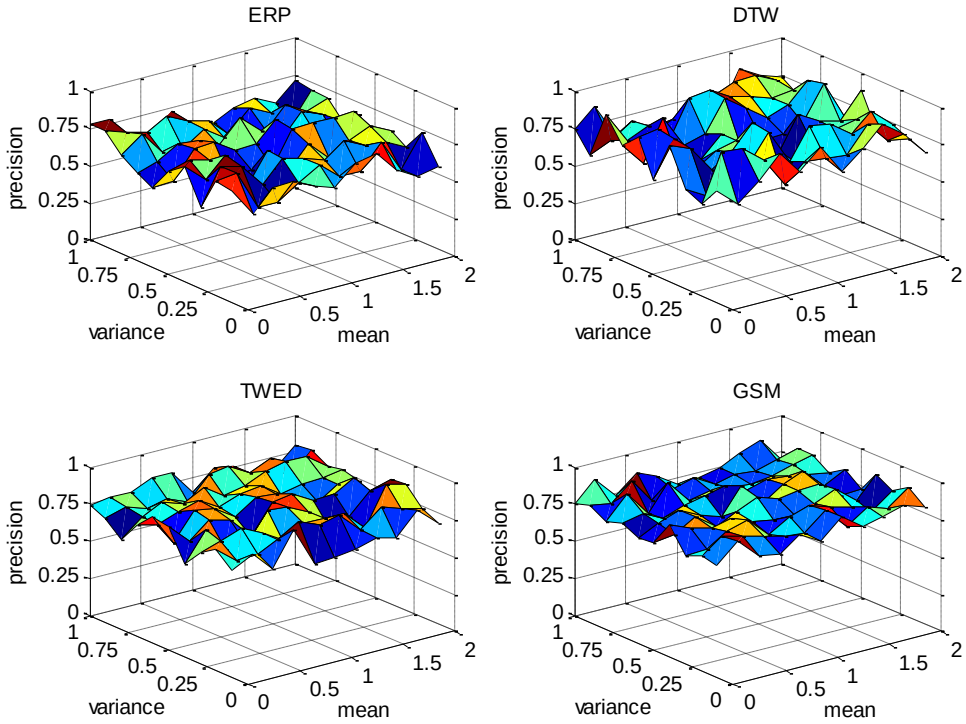


Figure 4.7 Retrieval precision of GSM on MNIST against Gaussian noise

Section 4.3.6 Capability to Handle Rich Temporal Aspects

In order to demonstrate the capability of GSM to handle rich temporal aspects, a classification experiment was conducted on each dataset where a leave-one-out mechanism was employed. Half of each dataset was chosen as the training data while the rest was taken as the test data. Table 4.7 shows the classification precision with different combinations of temporal aspects. From this we can see that the GSM can address most matching tasks involved in time-series and state-sequence data, especially with different temporal matching requirements.

Table 4.7 Classification precision with combinations of distance aspects

Dataset / Aspects	AT&T face	USPS	MNIST	COIL20	Isolet1	BinAlpha
T_{ord}	87.50	90.69	85.40	87.08	89.23	86.00
T_{dur}	91.00	86.56	82.20	88.75	90.13	87.18
T_{gap}	88.50	87.12	83.80	88.47	89.87	87.77
T_{ord}+T_{dur}	89.50	89.61	86.80	89.86	92.69	90.73
T_{ord} +T_{gap}	90.50	91.44	89.20	89.72	93.21	89.15
T_{dur} + T_{gap}	87.50	90.77	86.60	89.86	92.82	90.34
T_{ord}+T_{gap}+T_{dur}	**94.00**	**93.53**	**89.80**	**91.81**	**94.23**	**92.90**

In summary, the generalization of the proposed GSM has been explored first, which demonstrates that the conventional existing measurements can be regarded as special cases of our GSM. Particularly, the new LCS-based measurement named OTCS has been proposed, followed by its detail algorithms and the example evolution. The experimental results of the proposed GSM and the particular OTCS on 6 benchmark datasets have verified the performance for state-sequence matching. State-sequence matching is quite ubiquitous in real-life. So the next chapter will present two interesting investigations/case studies.

CHAPTER 5 CASE STUDY OF BASKETBALL ZONE-DEFENCE DETECTION

State-based temporal pattern recognition, the procedure for matching temporal pattern of time-series and state-sequences (also known as state-sequence matching), is a popular activity in real-life such as financial data analysis, audio recognition, visual information retrieval, etc. It has played a very important role in data mining, particularly with respect to temporal information. In the following two chapters, we shall investigate two video-based cases for temporal pattern recognition: basketball zone-defence detection in chapter 5 and video copy detection in chapter 6. The model of each case will be designed, and then novel strategies will be proposed to address the typical problems in each case.

Section 5.1 Formal Characterization and Basketball Zone-defence Detection

Based on the formal characterization of time-series and state-sequence, the formal characterization of our particular case, basketball zone-defence detection, will be presented in this section.

Section 5.1.1 Formal Characterization of Video Database

With the development and progress in information age, multimedia information, especially video information, is becoming an active and topical research object, which includes video retrieval, video structural representation, video annotation and so on. Videos can be organized at different levels for various research purposes. In this thesis, videos are organised in terms of clips as shown in figure 5.1. Each video, which

presents an entire story that to be analyzed/ studied, can be firstly divided into sequential video clips (each of which is actually constructed by sequential frames). Then, the sequential key-frames are obtained by specific key-frame extraction algorithm to represent the corresponding video clips. Therefore, the task of video analysis can actually be transformed into the problem of exploring the knowledge between key-frame sequences, where a feature vector is extracted from each corresponding key-frame.

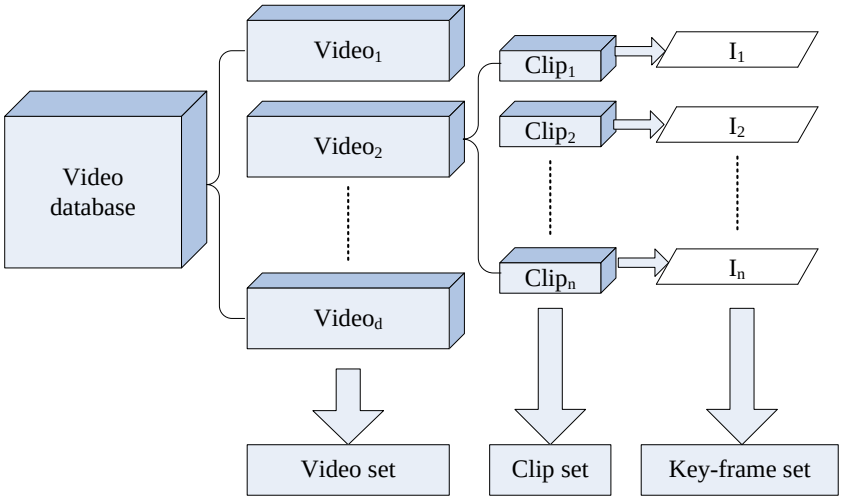


Figure 5.1 Video database organization

Therefore, the formal temporal characterization of video database based on the tetrad time-series and state-sequence can be described as follows:

GSSI1) $I = [I_1, \ldots, I_n]$

GSSI2) H = [Holds(I_i, t_i)], for all $i = 1, \ldots, n$,

where $[t_1, \ldots, t_n]$ is a time-series:

GTSI1) $T_n = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

GTSI2) $R = [\text{Meets}(t_i, t_{i+1}) \vee \text{Before}(t_i, t_{i+1})]$, for all $i = 1, \ldots, n\text{-}1$

GTSI3) $T_{dur} = [d_i] = [T_{dur}(t_i)] = [q_i - p_i]$, for all $i = 1, \ldots, n$.

GTSI4) $T_{gap} = [g_i] = [T_{gap}(t_i, t_{i+1})] = [p_{i+1} - q_i]$ for all $i = 1, \ldots, m\text{-}1$and $g_0 = 0$.

Specifically:

- T_n= [t_1, …, t_n] = [<p_1, q_1>, …, <p_n, q_n>] expresses the knowledge time elements involved with respect to the given collection of video clip. <p_i, q_i> denotes the start time and end time of the *i*th video clip.

- R= [Meets(t_i, t_{i+1})∨Before(t_i, t_{i+1})] is the collection of disjunctions of temporal relations over T_n, expressing the possible temporal relationship between each pair of adjacent key-frames (also the corresponding states) where "Meets(t_i, t_{i+1})" for complete time-series and "Before(t_i, t_{i+1})" for incomplete time-series.

- $T_{dur} = [d_i] = [T_{dur}(t_i)] = [q_i - p_i]$ is the collection of temporal duration assignments (possibly incomplete) to every time element in T_n, which is actually the duration of the *i*th video clip.

- $T_{gap} = [g_i] = [T_{gap}(t_i, t_{i+1})] = [p_{i+1} - q_i]$ is the collection of temporal gap assignments to each adjacent pair in time element T_n, which is actually the possible interval between each pair of adjacent key-frames.

From the tetrad characterization of the video database, we can see that the video pattern recognition follows the GSM (or the proposed OTCS), which is flexible enough to handle the situations with various temporal aspects. In this chapter, the video of basketball zone-defence will be studied and the zone-defence detection system with particular structure relationship will be explored.

Section 5.1.2 Basketball Zone-defence

As a case study of state-sequence matching, zone-defence detection in basketball videos is investigated in this chapter. Different from images, videos contain rich temporal information. Therefore, we focus our case study on video patterns. Broadly speaking, video pattern recognition aims to search out similar video(s) to match a query video. Video clip detection is an important task that has been widely researched [BABST2007, HR2007 and MBG2008].

Zone-defence detection is important in basketball games. On one hand, a coach needs to lay out the zone-defence strategy and check whether the team is playing in the right strategy or not all the time; at the same time, the coach also needs to know which zone-defence strategy the defenders are adopting.

Basketball zone-defence is a defensive strategy whereby each “zone defender” is responsible for guarding an area on the court (or "zone"), and any offensive player that comes into that area. Figure 5.2 shows the ordinary positions of 5 defenders in 1-3-1 zone defence. Zone defenders move their positions on the court according to where the ball moves. Zone-defence can disrupt the opponent’s offensive plan by means of protecting the paint area and forcing the opponent to shoot from outside. In addition, changing defences from man-to-man to various zones can make the offense off-balanced and confused.

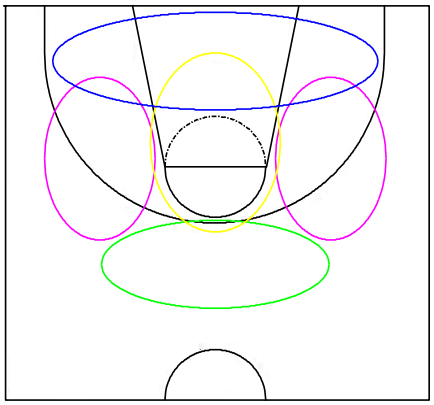


Figure 5.2 Defenders’ positions in 1-3-1 zone press

For instance, a typical round attacking in a 1-3-1 zone-defence clip can be represented by the frames showing in figure 5.3 where the yellow circles, the blue squares and the red dot denote the defenders, the offenders and the ball respectively. The arrows with solid lines show how the defenders generally move in the zone, while the arrows with dotted lines denote the direction of passing the ball, and the arrows with the curved lines denote the direction of dribbling the ball.

Diagram A shows the basic formation of the setup. *Diagram B* shows player movements as the ball crosses the half-court. If the ball is passed to the corner, the formation changes into *diagram C*. Similarly, the following *diagrams* (*D*-*J*) show the way on which the formation adjusts when the ball is moved.

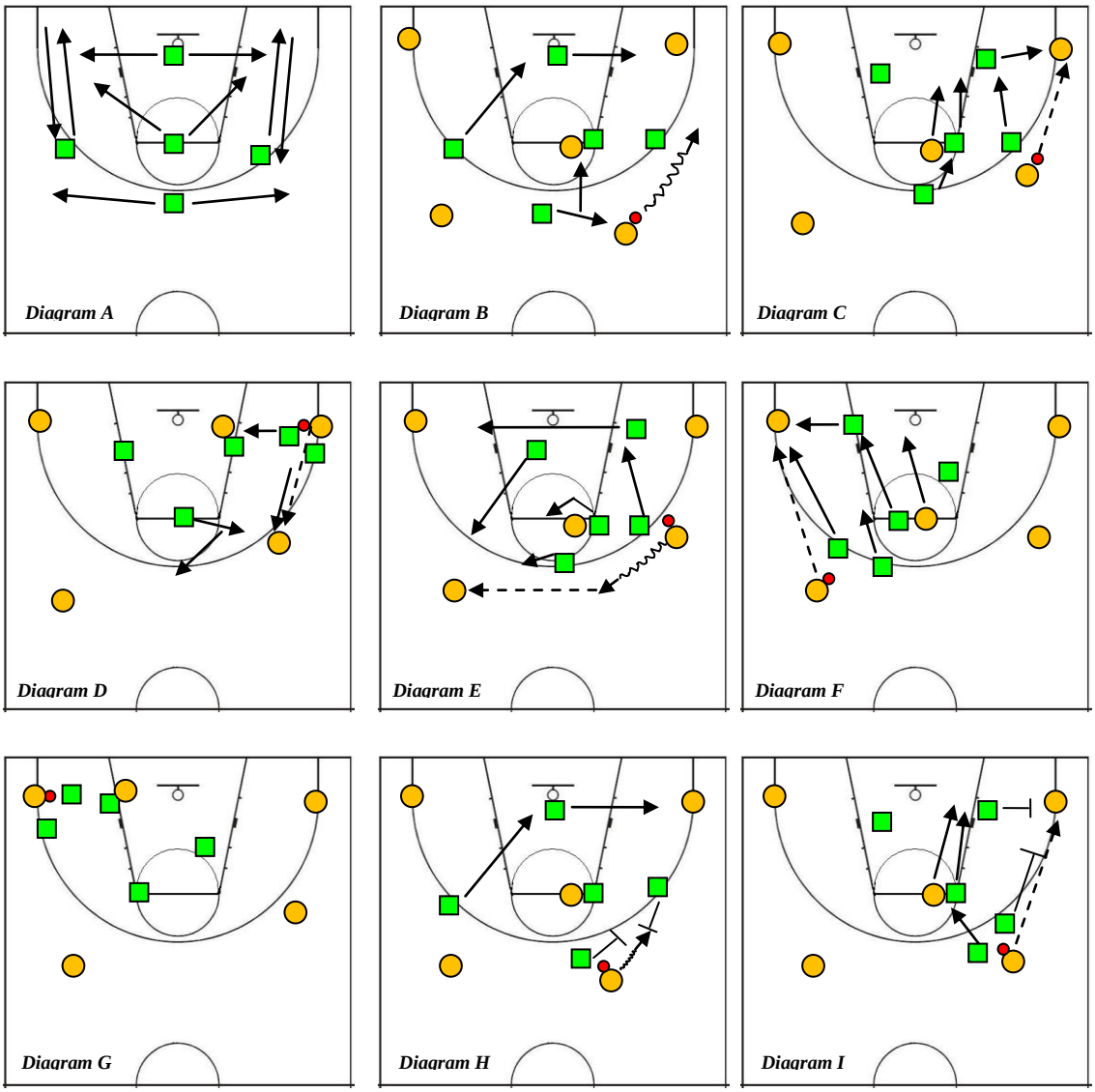


Figure 5.3 A typical round attacking in 1-3-1 zone-defence clip

Section 5.1.3 Graphic Representation of Basketball Zone-defence

In basketball zone-defence video, each clip represents a certain round of offense (or defence) and is denoted as a list of images, or the so-called key-frames sequence: $I = [I_1, \ldots, I_n]$, which consists of the key-frames extracted one per 2 seconds from the

clip. We premise that:

(1) The defenders have adjusted to their best defensive positions at the moment when the ball is about to be passed or dribbled;

(2) Since the zone-defence strategy is to defend against the offensive opponent attacking into the interior playfield, we only consider the key-frames when the ball is in the midfield, the wing and the corner as key-frames.

According to these two premises, a basketball zone-defence video clip is structured by zone-defence states or so-called state-sequence: $SS = [S_1, \ldots, S_n]$, and Holds(S_i, t_i) for $i = 1, \ldots, n$, where $[t_1, \ldots, t_n]$ is a time-series of the moments referred to the premise (1).

Each key-frame I_i ($i = 1, \ldots, n$) can be described by its corresponding six-note graph G_i structured by the 5 defenders' positions (horizontal and vertical coordinates) plus the ball's position. Following the conventional notations in graph theory, we represent a zone-defence graph as $G = <V, E>$, where V and E denote the set of notes (defenders' positions) and the set of edges respectively, and $E \subseteq V \times V$. In particular, here $|V| = 6$. The position of each note is denoted by the horizontal and vertical coordinates of the corresponding vertex. Assuming $V = \{V_b, V_1, V_2, V_3, V_4, V_5\}$, it is presented in ascending ordered by Euclidean distance to the ball (V_b).

Obviously, each state S_i has its corresponding graph G_i, where $i = 1, \ldots, n$. In addition, we shall use the following vector $[ball_1, \ldots, ball_n]$ to record the ball's position of each state, where $ball_i \in \{midfield, wing, conner\}$ for $i = 1, \ldots, n$.

Zone-defence can be divided into various kinds of zone-defence strategies, including 2-3, 1-3-1, 1-2-2, 3-2, 2-2-1, 2-1-2 and 1-2-1-1 strategies. The first three strategies, which have been noted as the most typical ones employed in actual basketball games, are focused upon in this thesis.

Section 5.1.4 System of Basketball Zone-defence Detection

Figure 5.4 shows the flow chart of the basketball zone-defence detection system. Each test zone-defence video clip is decomposed into a sequence of key-frames. Each key-frame is represented by a zone-defence graph as mentioned above and matched with the graphs in the standard zone graph database. The global distance from each standard zone is then obtained according to the graph-sequence that is the most similar (has the smallest distance) to the test graph-sequence, which in turn, provides matching results to confirm which zone-defence strategy the test key-frame sequence belongs to.

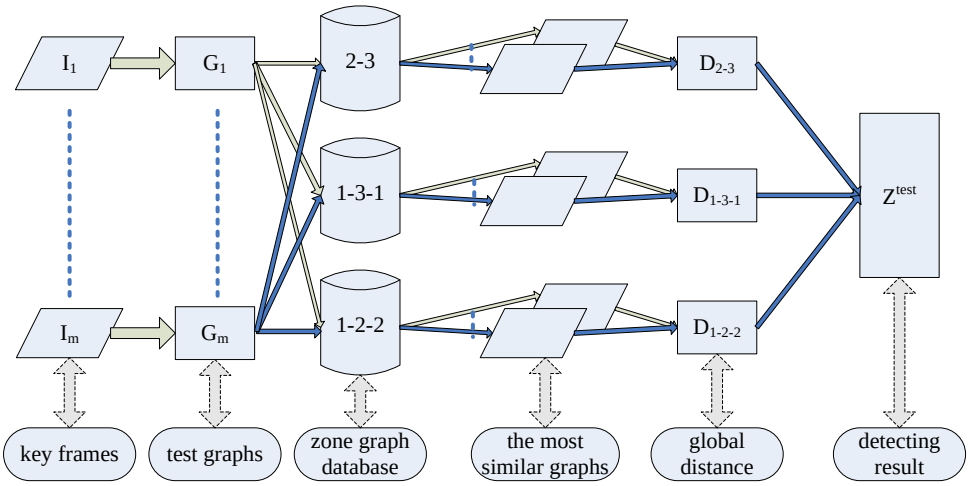


Figure 5.4 The flow chart of basketball zone-defence detection system

The detail procedure of basketball zone-defence detection is shown as follows:

Firstly, compute the distance between test clip and standard 2-3 zone-defence strategy.

Step 1: For each key frame I_i, $i = 1, 2, \ldots, m$, compute the distances between its corresponding zone-defence graph G_i and graphs with the same ball position as G_i in the standard 2-3 zone graph database:

$$D(G_i, G_{z_j}^{23}) = [d_{ij}^{23}] \tag{5-1}$$

where $ball_i^{test} = ball_{z_j}^{23}$, $z_j \in \{1,2,...,n_{23}\}$, $j = 1,2,...,n_p < n_{23}$, and n_p is the number of the graphs with the same ball position as G_i^{test} in 2-3 zone graph database..

Step 2: Determine the distance between G_i ($1 \le i \le m$) and 2-3 zone-defence strategy. The one with the smallest distance is the most similar graph to G_i:

$$D_i^{23} = \arg\min_j([d_{ij}^{23}]) \tag{5-2}$$

Step 3: Compute the global distance between the test clip and the 2-3 zone-defence strategy example. The sum of the smallest distances to each key-frame in the test zone-defence clip is calculated as the global distance:

$$GD_{test}^{23} = \sum D_i^{23} \tag{5-3}$$

Secondly, using the above three steps, we can define the global distance between the test clip and the 1-3-1 zone-defence strategy examples as:

$$GD_{test}^{131} = \sum D_i^{131} \tag{5-4}$$

Thirdly, we can define the global distance between the test clip and the 1-2-2 zone-defence strategy example in the same manner as:

$$GD_{test}^{122} = \sum D_i^{122} \tag{5-5}$$

Finally, the zone-defence strategy with the smallest global distance is regarded as the strategy that the test clip belongs to. The zone-defence strategy pattern of the test zone-defence video clip is calculated as:

$$Z^{test} = \arg\min_{z=\{23,131,122\}} (GD_{test}^{23}, GD_{test}^{131}, GD_{test}^{122}) \tag{5-6}$$

From the flowchart, we can see the step 1 is one of the most important techniques in the basketball zone-defence detection. How can we measure the similarity between the zone-defence graphs? The graph matching approach is the natural solution. Therefore, the Laplacian Matrix based graph matching algorithm is introduced for the basketball zone-defence detection in the next section.

Section 5.2 LM-based state matching algorithm

As mentioned above, each zone state has its corresponding zone graph. Therefore, state matching can be transformed into the corresponding graph matching. In this section, we shall extend the Laplacian matrix-based algorithm proposed in [LTWB2005] for matching zone graphs. The original algorithm proposed in [LTWB2005] is demonstrated to be precise in matching image pairs; however, on one hand, it is invariant with respect to zoom, and on the other hand, it is very sensitive to the translation of single vertex. The main process of the Laplacian matrix-based algorithm proposed in [LTWB2005] is expounded as follows:

Algorithm 5.1: Laplacian matrix-based graph matching

1) Formulate the Laplacian distance Matrices for zone graph G and H:

$$L(G)=[l_{ij}]=\begin{cases}-\left\|V_{G_i}-V_{G_j}\right\|^2 \Big/ M^2 & (i\neq j)\\ -\sum_{k\neq i} l_{ik} & (i=j,k\in\{1,...,n\}),i,j=1,...,5\end{cases} \tag{5-7}$$

$$L(H)=[l_{ij}]=\begin{cases}-\left\|V_{H_i}-V_{H_j}\right\|^2 \Big/ M^2 & (i\neq j)\\ -\sum_{k\neq i} l_{ik} & (i=j,k\in\{1,...,n\}),i,j=1,...,5\end{cases} \tag{5-8}$$

Here, we take M as the diagonal line length of the half-court playfield. Obviously, Laplacian Matrix $L(G)$ and $L(H)$ have following properties: positive,

semi-definite, the multiplicity of eigenvalue 0 is 1, and the corresponding eigen vector is with all 1 elements.

2) Compute the Singular Value Decomposition (SVD) for each Laplacian Matrix respectively:

$$L(G) = U \bullet diag\{\lambda_1, ..., \lambda_5\} \bullet U^T \tag{5-9}$$

$$L(H) = V \bullet diag\{\gamma_1, ..., \gamma_5\} \bullet V^T \tag{5-10}$$

where $\lambda_1 \geq ... \geq \lambda_{n-1} \geq \lambda_n = 0$ and $\gamma_1 \geq ... \geq \gamma_{n-1} \geq \gamma_n = 0$ denote the singular values of $L(G)$ and $L(H)$, $U = \{U_1, U_2, ..., U_n\}$ and $V = \{V_1, V_2, ..., V_n\}$ are $n \times n$ orthogonal matrices, $U_i (i = 1, 2, ..., n)$ and $V_i (i = 1, 2, ..., n)$ are column vectors of U and V.

3) Sign adjusting [LTWB2005] V and into $\tilde{V}$.

The decomposition of $L(H)$ is slight different from that of $L(G)$. The smaller the distance between V_i and U_i, the better. The detail measurement is: fixing the U_i, the V_i is adjusted and marked as $\tilde{V}_i$.

$$\tilde{V}_i = \begin{cases} V_i & if \ \|V_i - U_i\| < \|-V_i - U_i\| \\ -V_i & else \quad i = 1, 2, \cdots, n \end{cases} \tag{5-11}$$

Where the i th row vectors of U and $\tilde{V}$ reflect the features of i th vertices (characteristic points) of G and H respectively, marking as U^i and $\tilde{V}^i$.

4) Construct the matching distance. Thinking that:

$$D_{ij} = \left\| U^i - \tilde{V}^j \right\| = (U^i - \tilde{V}^j)(U^i - \tilde{V}^j)^T = 2[1 - U^i (\tilde{V}^j)^T] \tag{5-12}$$

So, the bigger $(U^i - \tilde{V}^j)^T$ is, the smaller the distance between U^i and $\tilde{V}^i$ is, which means the higher the possibility of matching the *i*th vertex of *G* and the *j*th vertex of *H*.

5) Define the matching relationship matrix:

$$C = U\tilde{V}^T = [U^i(\tilde{V})^T] = [C_{ij}] \tag{5-13}$$

C_{ij} reflects the matching relationship of vertices (characteristic points) between graph *G* and *H*. The *i*th vertex of *G* matches the *j*th vertex of *H* if C_{ij} is the biggest element in both its columns and rows.

6) Compute the matching distance of each vertex in *G*, with respect to its relationships to the vertices in *H*: $\forall i, j, k, t \in (1, 2, ..., n)$:

$$MD_i = \begin{cases} D_{ij}, & if \ \ C_{ij} = \max(C_{it}) \wedge C_{ij} = \max(C_{kj}) \\ \max D_{kt}, & else \end{cases} \tag{5-14}$$

7) Compute the compound matching distance between graph *G* and *H*:

$$Dis(G, H) = \sum_{i=1}^{n} MD_i \tag{5-15}$$

Obviously, $n = 5$ in basketball zone-defence graphs.

Note that in basketball zone-defence, in addition to the Spatial Distance (SD) relationships as characterized by formula (5.7) and (5.8), the Spatial Direction (SD') relationships between defenders also plays an indispensable role. Hence, additional direction Laplacian Matrices with respect to the direction relationships are formulated as:

$$L'(G)=[l'_{ij}]=\begin{cases}-R(V_{G_i}-V_{G_j})^2\big/\pi^2 & (i\neq j)\\ -\sum_{k\neq i} l'_{ik} & (i=j,k\in\{1,...,n\}),i,j=1,...,5\end{cases} \tag{5-16}$$

$$L'(H)=[l'_{ij}]=\begin{cases}-R(V_{H_i}-V_{H_j})^2\big/\pi^2 & (i\neq j)\\ -\sum_{k\neq i} l'_{ik} & (i=j,k\in\{1,...,n\}),i,j=1,...,5\end{cases} \tag{5-17}$$

where $R(V_{G_i}-V_{G_j})$ and $R(V_{H_i}-V_{H_j})$ denotes the direction relationships between vertex pairs (V_{G_i},V_{G_j}) and (V_{H_i},V_{H_j}) respectively:

$$R(V_{G_i},V_{G_j})=\underset{[0,\pi]}{\arg\cos}\left(\frac{x_{V_{G_j}}-x_{V_{G_i}}}{\left\|V_{G_j}-V_{G_i}\right\|}\right) \tag{5-18}$$

$$R(V_{H_i},V_{H_j})=\underset{[0,\pi]}{\arg\cos}\left(\frac{x_{V_{H_j}}-x_{V_{H_i}}}{\left\|V_{H_j}-V_{H_i}\right\|}\right) \tag{5-19}$$

N.B.: Single vertex translation has less effect on the direction Laplacian Matrices (as formulated in Eq.(5.16) and Eq.(5.17)) than the distance Laplacian Matrices.

With the same procedure as step 2) to step 6) as illustrated in the above, we can obtain the spatial direction distance between graph *G* and *H*: $Dis'(G,H)=\sum_{i=1}^{n} MD'_i$

Finally, the global matching distance between graph *G* and *H* is defined by:

$$D(G,H)=\mu Dis(G,H)+\mu' Dis'(G,H) \tag{5-20}$$

Where μ denotes the weight of the spatial distance in the global distance. The experimental results of the extended Laplacian Matrix-based graph matching algorithm

taking into account both spatial distance and spatial direction will be tested and demonstrated on basketball zone-defence detection system in section 5.1.5.

Section 5.3 Structure-based Feature Extraction

Graphical representation has been investigated for zone-defence detection. Graph matching (GM) algorithms and their improved variants have been well applied to match graph patterns [ZMLP2009 and MZH2007]. However, the efficiency and accuracy of most graph matching algorithms depends very much on the tested graphs constructed according to the expectation or artificial criteria, rather than real-life applications [ZMLP2009], which in turn means most graph matching algorithms are sensitive to outliers or local bias such as the translation of subprime notes in the graph. [CHTH2005] proposed a Spatial-Relationship (SR) based approach to describe the position relationship between defenders. However, this relies on the accuracy of identification of each defender, which is hardly achievable.

As we know, the defence-lines and the structure relationship between defence-lines play a crucial role in team sports such as basketball, football, volleyball and so on. The analysis of the structure relationship between defence-lines is very necessary and significant in basketball zone-defence. Therefore, in this thesis, a structure-based feature is proposed to describe the structure relationship between defence-lines.

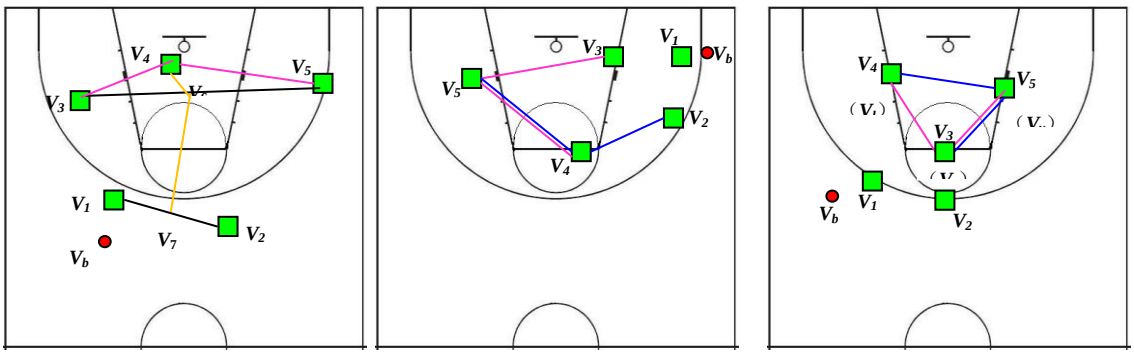
Different zone-defence strategies have a different number and type of defence-lines in basketball, For instance, there are two defence-lines in the 2-3 zone-defence strategy. Generally, the 2 defenders in the front line construct the first defence-line and the remaining 3 defenders are viewed as the second defence-line. Different zone-defence strategies have their own typical defence-lines. For instance, the typical defence-line of the 2-3 zone defence strategy is the second defence-line. We shall define the structure-based features to describe the structure relationship between defence-lines. The angle formed by the typical defence-line in each zone-defence

strategy is named the Character-Angle (CA), the definition of which is crucial to the extraction of the other structure features. Therefore, the 10 dimensional feature vector will be defined to describe the basketball zone-defence graphs in the following three subsections.

Section 5.3.1 Structure-based Features in 2-3 Zone-defence

In the standard 2-3 zone-defence strategy, normally we define the 2 defenders closest to the ball as the first defence-line and the remaining 3 defenders as the second defence-line, which is defined as the 2-3 character line. The angle formed from the 2-3 character line is defined as the "2-3 character-angle" and denoted in shorthand by writing CA_{23}: the angle constructed by the pink lines as shown in figure 5.5. One of the two supplementary angles formed by the character-lines that face the ball is regarded as the character-angle, similarly hereinafter for the 1-3-1 and 1-2-2 zone-defences.

There are two folds regarding the definition of CA_{23}:



(a) *a general example* (b) *counter example 1* (c) *counter example 2*

Figure 5.5 Zone graph examples in 2-3 zone-defence

(1) Which 3 notes construct CA_{23}?

Normally, CA_{23} is composed of the 3 defenders furthest from the ball. However, in some zone graphs, CA_{23} may not be exactly constructed by the 3 defenders furthest from the ball by common sense from human understanding of zone-defence strategies.

For instance, in Figure 5.5 (b), assume that $V = \{V_b, V_1, V_2, V_3, V_4, V_5\}$ has been ascendingly ordered by the distance to the ball (V_b) and V_3 and V_2 have the approximately same distance to the ball. For the reason of neatness, the layout of the 5 offenders is ignored. Obviously, the CA_{23} should be constructed by V_2, V_4 and V_5 (marked as the angle constructed by the blue line), which is more reasonable according to common sense than that constructed by the furthest 3 notes (V_3, V_4 and V_5).

In other word, if the difference between the distances from the third and forth furthest notes to the ball is smaller than a given threshold, then the one forming a larger angle with the segment constructed by the farthest two notes will be taken to form the character line. The algorithm is described as follows:

Algorithm 5.2: Notes determination to construct CA_{23}

If $(\left|\overline{V_2V_b} - \overline{V_3V_b}\right| < \delta) \,\&\, (\angle(V_2, \overline{V_4V_5}) > \angle(V_3, \overline{V_4V_5}))$

$CN_{23} = \{V_2, V_4, V_5\}$

Else $CN_{23} = \{V_3, V_4, V_5\}$

where δ=0.05 (the distance of diagonal of half-court is normalized to 1), CN_{23} denotes the set of notes constructing CA_{23} and $\angle(X, \overline{YZ})$ represents the angle between note X and segment YZ which is defined as:

$$\angle(X, \overline{YZ}) = \begin{cases} \angle XYZ & |XY| > |XZ| \\ \angle XZY & \text{else} \end{cases} \tag{5-21}$$

(2) Which one is the vertex of CA_{23}?

For the reason of simple description, without losing generality, we assume CN_{23} = $\{V_3, V_4, V_5\}$, as shown in Figure 5.5(c), and arrange $\{V_3, V_4, V_5\}$ into $\{V_l, V_v, V_r\}$ in clockwise order with respect to the ball, where $l, v, r \in \{3, 4, 5\}$. In general, node V_v is

then taken as the vertex of CA_{23} while V_l, V_r, which denote the left node and the right node respectively, are the end-points of CA_{23}. However, if Angle < V_v, V_b, V_l > (or Angle <V_v, V_b, V_r>) is smaller than a given threshold, and $|V_lV_b| < |V_vV_b|$ (or $|V_rV_b| < |V_vV_b|$) then V_l (or V_r) will be re-taken as the vertex of CA_{23}. For instance, in Figure 5.5(c), $CN_{23} = \{V_3, V_4, V_5\}$. Assume that V_4, V_5 and V_3 are in the clockwise order with respect to the ball. V_3 should be defined to be the vertex of CA_{23}, which is more reasonable than regarding V_5 as the vertex of CA_{23}. The algorithm is described as follows:

Algorithm 5.3: Character angle detection of 2-3 zone-defence

If $(\angle V_lV_bV_v < \theta) \& (|V_lV_b| < |V_vV_b|)$

$CA_{23} = \angle V_vV_lV_r$

Else if $(\angle V_rV_bV_v < \theta) \& (|V_rV_b| < |V_vV_b|)$

$CA_{23} = \angle V_vV_rV_l$

Else $CA_{23} = \angle V_lV_vV_r$

where $\theta = \pi/12$ and we appoint CA_{23} as the obtuse angle if its vertex is biased towards the ball compared with its two end points.

The first 4 structure features with respect to CA_{23} are correspondingly defined as below (As for the general example illustrated in figure 5.5(a), $\overline{V_1V_2}$ is the first defence-line and V_3, V_4, V_5 is the second defence-line, and V_6, V_7 are the midpoints of $\overline{V_3V_5}$, $\overline{V_1V_2}$ respectively):

I. $CA_{23} = \angle V_3V_4V_5$: Character-Angle of 2-3 zone-defence.

As explained earlier, this angle characterises the defenders' positions on the character line of 2-3 zone-defence.

II. $FSA_{23} = \angle(\overline{V_7V_6}, \overline{V_3V_5})$: Angle formed by the first and the second defence-lines.

where $\angle(\overline{XY}, \overline{ZW})$ denotes the acute angle formed by segment $\overline{XY}$ and segment $\overline{ZW}$ and similarly hereinafter. It characterises the structural relationship between the first and the second defence-lines.

III. $BCA_{23} = \angle(\overline{V_4V_6}, \overline{V_3V_5})$: the bias of the CA_{23}.

which is an angle that presents the bias of the vertex on second defence-lines of 2-3 zone-defence.

IV. $RFSA_{23} = (|V_1V_2|/|V_3V_5|) \angle (\overline{V_1V_2}, \overline{V_3V_5})$: restricted FSA_{23}.

which denotes the restricted angle of the first and the second defence-lines of the 2-3 zone-defence. The shorter $\overline{V_1V_2}$ is in comparison to $\overline{V_3V_5}$, the less effect the angle of segment $\overline{V_1V_2}$ and segment $\overline{V_3V_5}$ has to zone graphs. So, it is reasonable to take into account a coefficient to the angle.

Section 5.3.2 Structure-based Features in 1-3-1 Zone-defence

In 1-3-1 zone-defence, the nearest defender to the ball represents the first defence-line. The second defence-line is constructed by 3 defenders, presenting the basic character of the 1-3-1 zone-defence, which is defined as the 1-3-1 character line. The angle formed from the 1-3-1 character line is defined as the "1-3-1 character-angle" and denoted as CA_{131}. The key point here is to define the vertex and two end points of CA_{131}.

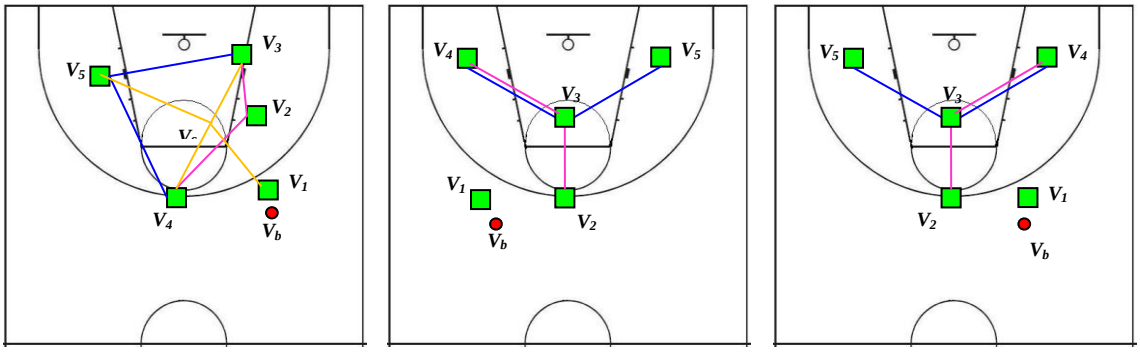


Figure 5.6 Zone graph examples in 1-3-1 zone-defence

Based on CA_{23}, as defined above, there are two cases to define CA_{131}: (Here, we also use V_1, V_2, V_3, V_4 and V_5 to denote the 5 defenders, and assume V_1 is the nearest defender to the ball, $CA_{23} = \angle V_3V_5V_4$ in Figure 5.6(a) and $CA_{23} = \angle V_4V_3V_5$ in Figure 5.6(b) and (c) marked as the blue lines). If the corresponding CA_{23} is smaller than π (as shown in Figure 5.6(a)), then CA_{131} has the same two end-points (V_3 and V_4) as that of CA_{23}, and the vertex of CA_{131} is the node (V_2) from the remaining 3 that is neither the closest to the ball nor the vertex of CA_{23}. Otherwise (as shown in Figure 5.6(b) and (c)), CA_{131} will have the same vertex as that of CA_{23} (V_3), and the node (V_2) which is neither on the 2-3 character line nor the closest to the ball will be taken as one of the two end-points of CA_{131}, and then the other end-point is one of the two end-points of CA_{23} (V_4) which will ensure that CA_{131} divides the remaining two nodes into each side of the 1-3-1 character line respectively. The detection algorithm is expounded below:

Algorithm 5.4: Character angle detection of 1-3-1 zone-defence

If $CA_{23} = \angle V_3V_5V_4 < \pi$

$$CA_{131} = \angle V_3V_2V_4$$

Else $CA_{23} = \angle V_4V_3V_5 \geq \pi$

$$CA_{131} = \angle V_2V_3V_4$$

In continuation from the first 4 features with respect to CA_{23}, the next 3 features with respect to CA_{131} are defined below (as for the general example illustrated in Figure 5.6(a), and assumes V_6 is the midpoint of segment $\overline{V_3V_4}$):

V. $CA_{131} = \angle V_3V_2V_4$: Character-Angle of 1-3-1 zone-defence, which characterises the defenders' positions on the character line of the 1-3-1 zone-defence analogously.

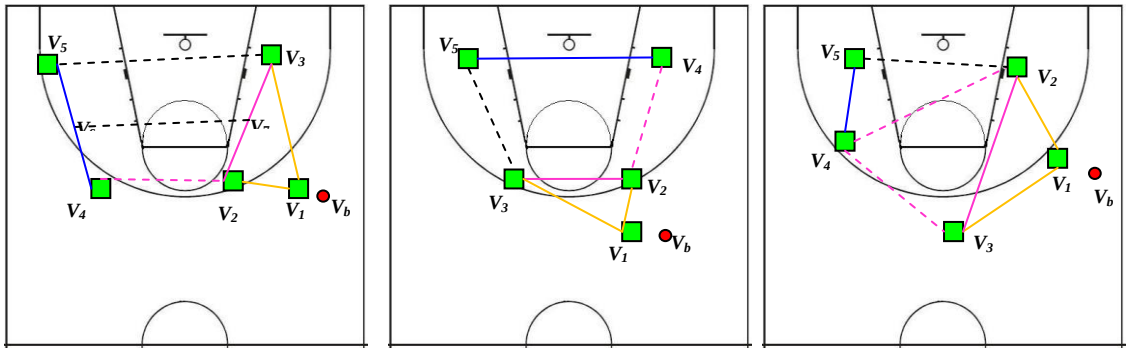
VI. $FSA_{131} = \angle(\overline{V_1V_6}, \overline{V_3V_4})$: Acute angle formed by the first and the second defence-lines, which characterises the structure relationship between the first and the second defence-lines of the 1-3-1 zone-defence.

VII. $STA_{131} = \angle(\overline{V_5V_6}, \overline{V_3V_4})$: Acute angle formed by the second and the third defence-lines, which characterises the structure relationship between the second and the third defence-lines of 1-3-1 zone-defence.

Section 5.3.3 Structure-based Features in 1-2-2 Zone-defence

In the 1-2-2 zone-defence, the defender closest to the ball forms the first defence-line. As per the examples shown in figure 5.7, assume that V_1 is the closest defender; the CA_{131} is $\angle V_4V_2V_3$ in figure 5.7(a) and (b) marked as the pink dotted line and the pink solid line, while $\angle V_2V_4V_3$ in figure 5.7(c) marked as pink dotted lines. If $CA_{131} \geq \pi$ (Figure 5.7(a) and (b)), the vertex of CA_{131} (V_2) and the nearer one (V_3) to the first defence-line (V_1) of the two end-points of CA_{131} construct the second defence-line ($\overline{V_2V_3}$ marked as the pink solid line); the remaining two defenders define the third defence-line ($\overline{V_4V_5}$ marked as the blue line). Otherwise (Figure 5.7(c)), the two end-points of CA_{131} define the second defence-line ($\overline{V_2V_3}$ marked as the pink solid line) and the rest two defenders define the third defence-line ($\overline{V_4V_5}$ marked as the blue line).

The first and the second defence-lines present the basic character of 1-2-2 zone-defence. The angle formed from the 1-2-2 character line is defined as "1-2-2 character-angle" ($\angle V_2V_1V_3$ marked as the yellow lines) and denoted as CA_{122}.



(a) $CA_{131} \geq \pi$ case 1 (b) $CA_{131} \geq \pi$ case 2 (c) $CA_{131} < \pi$

Figure 5.7 Zone graph examples in 1-2-2 zone-defence

The algorithm is described as follows (CA_{122}, SDL_{122} and TDL_{122} denote the Character Angle, the second defence-line and the third defence-line of 1-2-2 zone-defence, respectively):

Algorithm 5.5: Character angle detection of 1-2-2 zone-defence

If $C_{131} = \angle V_4V_2V_3 \geq \pi$

$|V_1V_3| < |V_1V_4|$

$CA_{122} = \angle V_2V_1V_3$, $SDL_{122} = \overline{V_2V_3}$, $TDL_{122} = \overline{V_4V_5}$

Else

$CA_{122} = \angle V_3V_1V_4$, $SDL_{122} = \overline{V_3V_4}$, $TDL_{122} = \overline{V_2V_5}$

Following the first 4 features with respect to CA_{23} and the 3 features with respect to CA_{131}, the last 3 features with respect to CA_{122} are defined as below (assume that $CA_{122} = \angle V_2V_1V_3$, $SDL_{122} = \overline{V_2V_3}$ and $TDL_{122} = \overline{V_4V_5}$, V_6 and V_7 are the midpoints of segment $\overline{V_4V_5}$ and segment $\overline{V_2V_3}$ respectively as shown in Figure 5. 6(a)):

VIII. $RCA_{122} = (\min(|V_1V_2|, |V_1V_3|)/\max(|V_1V_2|, |V_1V_3|)) \angle V_2V_1V_3$

Here, we add a coefficient to take into account the effect from the movement of node V_1 along the circle formed from V_1, V_2 and V_3.

IX. $RSTA_{122} = (|V_2V_3|/|V_4V_5|) \angle (\overline{V_2V_3}, \overline{V_4V_5})$

$RSTA_{122}$ is with respect to the restricted angle of segment $\overline{V_2V_3}$ and segment $\overline{V_4V_5}$ and reflects the structure relationship between the second and the third defence-lines of 1-2-2 zone-defence.

X. $BST_{122} = \angle (\overline{V_6V_7}, \overline{V_2V_3})$

which reflects the bias between the second and the third defence-lines of 1-2-2 zone-defence.

The feature vector is constructed by the above 10 features with respect to the three typical zone-defence strategies:

$f = \{CA_{23}, FSA_{23}, BCA_{23}, RFSA_{23}, CA_{131}, FSA_{131}, STA_{131}, RCA_{122}, RSTA_{122}, BST_{122}\}$

The feature vector is not only listed by the 10 components one by one, but also has internal relationships. The features of one typical zone-defence also reflect the structure relationship of the other typical zone-defences.

According to the structure-based features extracted above, the test basketball zone-defence video clip with n key-frames (or zone-defence graphs) can be represented by a $n \times 10$ feature matrix $F_{clip} = \{f_1, f_2, ..., f_n\}'$ and a ball's position vector $ball_{clip} = \{ball_1, ball_2, ..., ball_n\}$, where $f_i = \{f_{i1}, f_{i2}, ..., f_{i10}\}$ and $ball_i$ denotes the feature vector and the ball's position of the ith key-frame of the detected clip

respectively. Analogously, the 3 standard zone-defence databases are represented by 3 corresponding feature matrices with their respective ball position vectors. For instance, the standard 2-3 zone-defence database is represented by $F_{23} = \{f_1^{23}, f_2^{23}, ..., f_{14}^{23}\}'$ and $ball_{23} = \{ball_1^{23}, ball_2^{23}, ..., ball_{14}^{23}\}$. The distance between two zone-defence graphs can be expressed by:

$$D(G, H) = ED(f_i, f_{z_j}^{23}) = [d_{ij}^{23}] \tag{5-22}$$

Section 5.4 Experimental Results

Section 5.4.1 Experimental Setup

A standard zone-defence graph database of the 3 typical zone-defence strategies (2-3, 1-3-1 and 1-2-2 zone-defence) was constructed and populated with graph data corresponding to some of the pictures illustrated on two basketball coaching web sides[8].

Table 5.1 below shows the detailed number of zone-defence graphs collected as standard zone-defence graphs for each strategy in different ball position. Analogously, only the key-frames when the ball is in the midfield, the wing and the corner are considered.

Table 5.1 The number of standard zone-defence graphs

Zone-defence / Ball's position	2-3	1-3-1	1-2-2
Midfield	4	3	2
Wing	4	12	7
Corner	6	6	2
Totally	14	21	11

[8] http://www.coachesclipboard.net; http://www.guidetocoachingbasketball.com

The metric position detection of defenders and the ball is implemented similarly as in [ABCB2003]: The ball's position, which is either in the midfield, in the wing, or in the corner, is obtained from its motion described in terms of camera motion, which in turn, is captured by image motion estimation algorithm [BCB1999]. As for the defenders' positions, in the first place, the defending and offensive sides are distinguished by the colour difference of sportswear; template matching and projective transformation are then implemented to determine the metric position of defenders [ABCB2003].

The system has been tested using both simulated and real basketball zone-defence video clips. We formulated 40 clips (key-frame sequences) provided by the professional coaches and collected about 1 hour of real basketball zone-defence video, including 112 clips containing 3 to 8 key-frames each as listed in Table 5.2.

Table 5.2 The number structure of test data

	Zone-defence strategy		**Total clips**	**Total key-frames**
Simulated	2-3	20	60	145
	1-3-1	20		161
	1-2-2	20		128
Real-life	2-3	52	112	286
	1-3-1	31		221
	1-2-2	29		169

Section 5.4.2 LM-based Basketball Zone-defence Detection

First, we give an example of the matching (global) distances between a given test state-sequence and 3 standard zones as shown in figure 5.8, where: the second row is the corresponding graphs of the test state-sequence with 3 states as shown in the first row; the remaining rows are the most similar graph compared with each test graph in 2-3, 1-3-1, and 1-2-2 zone-defence strategies, as appearing in the row order. From figure 5.8, it can clearly be seen that the most similar zone-defence formation in

comparison to the test state-sequence is the 2-3 zone-defence pattern, which agrees with the matching result from our algorithm.

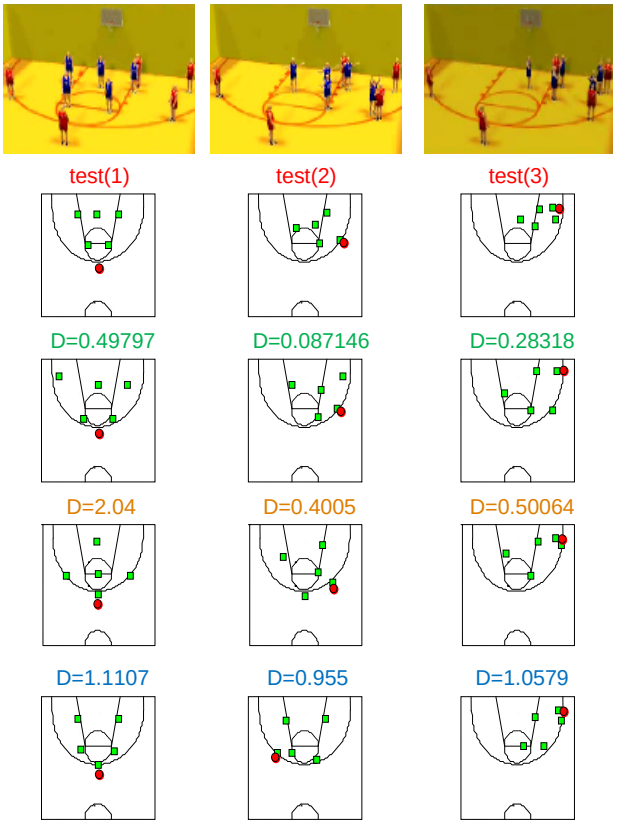


Figure 5.8 An example of basketball zone-defence video clip recognition

Table 5.3 below shows the matching precision for each zone-defence pattern. It indicates that the matching algorithm (SDD') proposed here, which takes into account both spatial distance and spatial direction relationships, outperforms SD or SD', whichh only address spatial distance or spatial direction relationships, respectively. In particular, the weights of SD is 0.75 (μ=0.75), which means the weight of SD' in SDD' is 0.25 in Eq. (5.20), leading to the optimal results.

From the table 5.3 we can see:

1) The LM-based graph matching algorithm is effective for zone-defence graphs which can lead the average precision from 68.8% to 91.6%;

2) In LM-based graph matching schema, the Spatial Distance is more significant than the Spatial Direction. In fact, the precision of SD (78.8%) is higher than that of SD' (75.6%), where the setting of the weights of SD (0.75) and SD' (0.25) leads to the optimal results for SDD'.

3) Both the Spatial Distance and the Spatial Direction should be taken into account. In fact, the average precision of SDD' (85.2%) is higher than either the precision with SD (78.8%) or the precision with SD' (75.6%).

Table 5.3 Matching precise for each zone-defence pattern

Test data	**Zone**	**Precision (%)**			**Average precision**
		SD	SD'	SDD'	
Real	2-3	74.6	69.8	82.7	75.7
	1-3-1	65.9	63.1	77.4	68.8
	1-2-2	80.3	70.7	86.2	79.0
Simulated	2-3	82	80	85	82.3
	1-3-1	91	89	95	91.6
	1-2-2	79	81	85	81.6
Average precision:		78.8	75.6	85.2	

Section 5.4.3 CA-based Basketball Zone-defence Detection

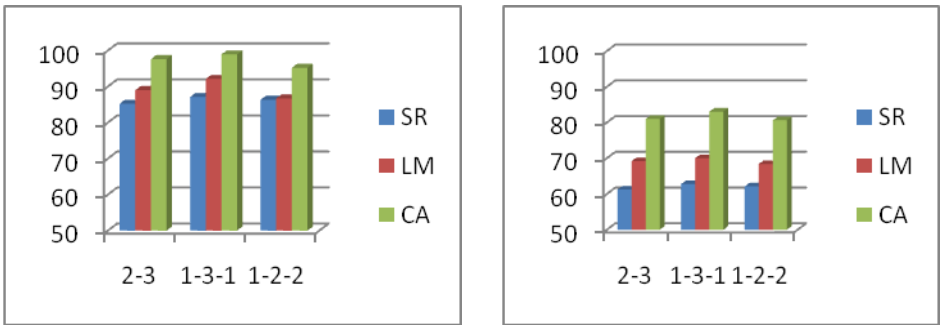
The CA-based algorithm is the first work that focuses on feature description of basketball zone-defence graphs. There are few systems focused on basketball zone-defence detection. Here, we compare the proposed CA-based algorithm with the LM-based algorithm in section 5.2 and SR-based algorithm [CHTH2005].

Table 5.4 reports the detection result of each algorithm based on both simulated and real-life data. Here detection results of "Correct MPD (Metric Position Detection)" are the results detected on the test clips with correct MPD. It's clear that the CA-based algorithm has the highest efficiency, especially with regard to correct MPD.

Table 5.4 Detection result of 3 algorithms based on different data

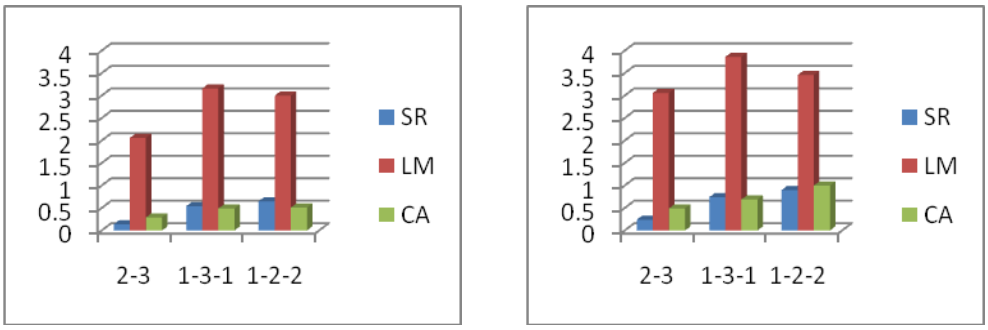
Database		**Video clips**		**Correct MPD**	
	Results	Test	Detected	Test	Detected
Simulated data	SR	40	35	38	34
	LM		36		35
	CA		**37**		**37**
Real-life data	SR	112	70	91	69
	LM		78		74
	CA		**91**		**85**

Figure 5.9 and figure 5.10 show the detecting precision and detecting complexity of the proposed CA-based algorithm compared with the other two algorithms in both simulated data and real-life data on each zone-defence. With respect to the computational complexity of the flow chart of system shown in figure 5.4, the overall time complexity $T_{all} = T_{in} + T_f + T_m + T_{out}$ where T_{in}, T_f, T_m, T_{out} denote the time for input, feature extracting, zone matching and output respectively. In order to emphasize the effectiveness of different matching algorithms, the input and output time, which are the same in the system with different matching approaches, were ignored. This means only $T_f + T_m$ were reported in figure 5.10. It is clear that the CA-based detecting method has higher detecting precision than the SR-based and LM-based algorithms in both simulated and real-life data. In comparison with LM-based graph matching algorithm, benefiting from use of the simple similarity strategy (Euclidian Distance), both SR-based and CA-based approaches have less computational complexity



(a) Simulated data (b) Real-life data

Figure 5.9 Detecting precision comparison with different methods



(a) Simulated data (b) Real-life data

Figure 5.10 Detecting complexities comparison with different methods

It is frequent for defenders to have some translational motion compared with the standard position in standard zone graphs. So the translational motion of the farthest defence-line from the ball in each zone-defence graph, which is regarded to have least influence to the global strategy, is added to the test video clip as a disturbance to test the robust of proposed approach. For each note V on the farthest defence-line in each zone-defence, we add the disturbance α as:

$$V' = V \pm d(\cos\gamma + \sin\gamma) \tag{5-23}$$

where d denotes the movement distance of note V to V' and γ denotes the angle between d and the x-axis (the mid-field line) as shown in Figure 5.11. Figure 5.12 shows the efficiency in each zone-defence with different disturbance.

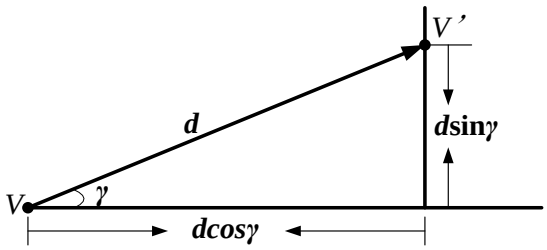
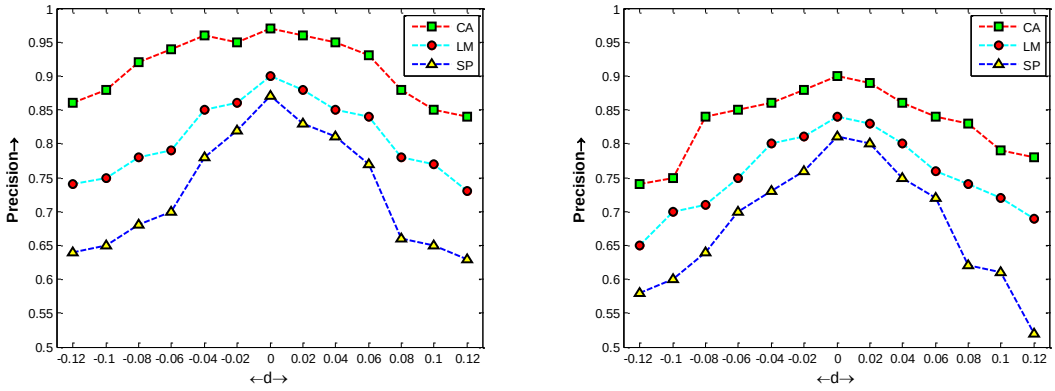


Figure 5.11 Disturbance of the nodes on the farthest defence-line



(a) Simulated data (b) Real-life data

Figure 5.12 Precision influence with disturbance in each method

The precision comes down with growing disturbance in every method. But in the CA-based method, it drops much slower than the other two and still has a tolerable performance even with a high disturbance, which demonstrates that the CA-based method is robust for the detecting system.

CHAPTER 6 CASE STUDY OF VIDEO COPY DETECTION

A video clip is constructed by a well-ordered sequence of frames (images). Due to the rapid increase of multimedia and the shortage of storage, in many real applications, the video databases are represented in terms of the sequence of high-dimensional feature vectors, which consists of the popular low-level features including color distribution, texture structure, shape configure, spectral character and so on. Therefore, the video clip matching problem can be transferred into a feature sequence matching problem.

Section 6.1 Problem Definition of Video Copy Detection

Video copy detection, also named video subsequence identification or video subsequence matching, is very significant for copyright authorization in commercial society where we would like to identify whether the current video clip is simply transformed from another video clip. Especially in TV commerce, it is essential to clarify the original TV shows from varies TV channels. Generally speaking, there are two categories of video copy detection: video watermarking and content-based video copy detection. First of all, it is essential to distinguish their conceptions.

Video watermarking: Video watermarking can be understood as the technique that permanently "embeds" the identifiable signal(s) or pattern(s) into the host video, to protect the copyright of digital video products. The main difference of watermarking is that we cannot detect the originality of a product if it has not been “watermarked” or “embedded”.

Video copy detection: Video copy detection can be considered as the procedure to detect whether a query video clip has been re-edited (such as crossover, deleting,

inserting) or visual transformed (such as reformatted, resolved, brightened, ect.) compared to some original one. It's a typical subsequence matching problem.

Previously, various similarity models based on Euclidean distance [AFS1993] have been proposed for subsequence matching, one efficient category being the sliding window based algorithms [FRM1994, MWL2001, MWH2002]. However, most models are very brittle; even a slight misalignment in time axis and the time-consuming problem would limit their application to large databases. Subsequently, many successful measurements such as Longest Common Sequence (LCS) [VGK2002], Edit distance [ALK1999], Dynamic time warping (DTW) [SC1978] and their variants emerged as required. LCS is directed at finding the longest common sequence in all the sequences (two in our case) along the same temporal order. It can skip some states that include noise but ignores how many and which kind of states it skips. ED calculates the similarity between two state-sequences by the number of operations such as insertion, deletion and substitution required to transform one to the other. However, reordering operations such as crossover and backward, which are very common in time-series data, are not allowed. DTW is robust to time warping such as stretching and shrinking (which means with different durations of each state), followed by variants such as PDTW [KP2000], SPRING method [SFY2007], EDTW [APPK2008]. However, they are very sensitive to noise, since each state will be matched including the noise.

Therefore, the objective of this chapter is to present an efficient framework for subsequence matching based on a bipartite graph representation and to propose a hybrid similarity model, while taking into account both spatial and temporal similarity with high tolerance in inversion, crossover and noise (noises).

Based on the above explanation, the formal definition of video copy detection can be defined as follows:

Definition 6.1: Video copy detection. Let $Q = [q_1, q_2, \ldots, q_m]$ be the query video state-sequence and $SS = [s_1, s_2, \ldots, s_n]$ be the video state-sequence in the video database, where m, n denote the length of video sequence Q and SS respectively and $q_i = (q_{i1},\ldots, q_{id})$ and $s_j = (s_{j1},\ldots, s_{jd})$ denote the d-dimensional feature vectors for the corresponding frames. The task of video copy detection is to detect a subsequence $S = [s_{k1}, s_{k2}, \ldots, s_{kt}]$ in SS, where $1 \le k1 < k2 < \ldots. < kt \le n$, which is most similar to the query video sequence Q.

(a) Key frame sequence one

(b) Key frame sequence two

Figure 6.1 Key frame sequences from the same video scenario with difference temporal order

Generally, the typical subsequence matching technique is directed at detecting similar sequences along the strictly same temporal order. In fact, the restrictive temporal order always ruins the task of video copy detection since the copy video may derive from an existing video by different ordering. For instance, Figure 6.1 illustrates such a scenario from “Fox Business”.

If we compare the two sequences in figure 6.1 according to the strict temporal order, there will be low similarity between the corresponding frame pairs. However, the two sequences contain the same content since they are from the same video scenario. Therefore, for video copy detection, the similarity between sequences with different temporal edition (eg. reorder) needs to be considered.

Section 6.2 Bipartite Graphical Representation

While the video clip is organized as a key-frame sequence, the video copy detection problem can actually be transformed into the bipartite graph matching problem with particular temporal measurement. We shall systematically introduce the procedure of transforming subsequence matching into the bipartite graph matching problem in this section.

Definition 6.2, bipartite graph: In graph theory, a bipartite graph <*X*, *Y*, *E*> is a graph where the vertices can be classified into two disjoint sets *X* and *Y*. The pair of vertices connected by each edge are in *X* and *Y* separately. Figure 6.2 shows an example of a bipartite graph.

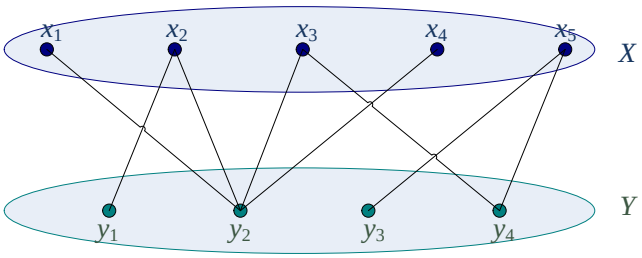


Figure 6.2 An example of bipartite graph

Table 6.1 Notations used in this section

Notation	Definition
$Q = [q_1, q_2, \ldots, q_m]$	Query state-sequence
$SS = [s_1, s_2, \ldots, s_n]$	A state-sequence in database
$D = [SS_1, \ldots, SS_L]$	The database with L state-sequences
$NN(q_i, SS, d_{max})$	Set of nearest neighbours of q_i in SS
$NN(Q, SS, d_{max})$	Set of nearest neighbours of all q_i in Q in SS
$BG = <Q, SS, E>$	Bipartite graph between Q and SS
$MSM(Q, SS)$	The set of MSM between Q and SS
$MSM(Q, D)$	The set of MSM between Q and all SS_j in D
M	A normal matching in $MSM(Q, D)$
$\overleftarrow{M}$	An inverse-ordered matching of M

The list of notations that will be used in this section is given in Table 6.1. The procedure can be briefly described as following:

Section 6. 2.1 Searching the similar pairs by thNN

For the query video clip Q and one of the video clips in the database, as shown in figure 6.2, the first task is searching the similar key-frame pairs between two key-frame sequences. Due to the repeating or re-referring phenomenon of video clips, for each key-frame in a query video clip, there may be several similar key-frames in the database. Therefore, a *k*NN (*k* Nearest Neighbours) approach is adopted. Given a query key-frame, the idea of *k*NN is to search out the *k* nearest key-frames in the video sequence to be matched. Considering that some key-frames may have few similar key-frames in the video clip to be matched, it is redundant to search out the *k* nearest key-frames for every key-frame in query video clip. For instance, for a noise key-frame, there may be no similar key-frames in the video clip to be matched. Therefore, different from the original *k*NN searching technique, a distance threshold d_{max} is defined for *k*NN to search for each state q_i in SS within the given maximum distance d_{max}. We name it threshold Nearest Neighbours (*th*NN), whose procedure can be illustrated as algorithm 6.1.

Algorithm 6.1: The threshold Nearest Neighbours

Input:
q_i, SS
Output:
$NN(Q, SS, d_{max})$: set of nearest neighbors of q_i in SS within distance d_{max}
Initialization:
$NN(q_i, SS, d_{max}) = NN(Q, SS, d_{max}) = null$
Updating:
For i = 1 to m and j = 1 to n
If distance(q_i, s_j) < d_{max}
$NN(q_i, SS, d_{max}) = NN(q_i, SS, d_{max}) \cup s_j$

$NN(Q, SS, d_{max}) = \{NN(Q, SS, d_{max}) , NN(q_i, SS, d_{max})\}$

Section 6.2.2 Constructing un-weighted bipartite graph

Based on the *thNN* approach, we can define the bipartite graph $BG = <Q, SS, E>$ for $NN(Q, SS, d_{max})$, where the key-frames in Q and SS are constructed as the nodes allocated on each side of the bipartite graph respectively. For each key-frame state pair q_i and s_j, the edge exists if and only if distance(q_i, s_j) < d_{max}. In other words, each key-frame q_i is only linked to its threshold neighbours. The edge set, $E \subseteq Q \times SS$, actually denote *thNN* mapping between Q and SS, as shown in Figure 6.3:

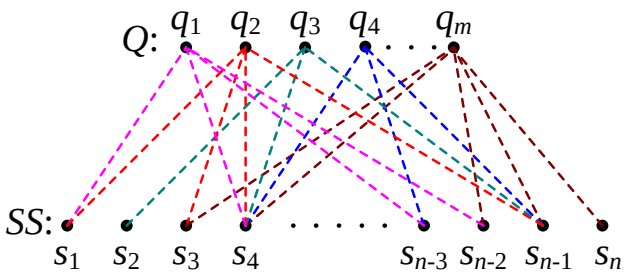


Figure 6.3 Bipartite graph representation

Obviously, the number of edges related to q_i is $|NN(q_i, SS, d_{max})| = b_i$ and the total edges in the bipartite graph is $\sum_{i=1}^{m} |NN(q_i, ss, d_{\max})|$. The set of mappings between Q

and *SS* is {<q_1, *NN*(q_1, *SS*, d_{max})>, < q_1, *NN*(q_1, *SS*, d_{max})>, …, < q_1, *NN*(q_1, *SS*, d_{max})>}. It is a 1: M mapping bipartite graph since the number of *NN*(q_i, *SS*, d_{max}) is not unique (larger than 1), which means each key-frame in the query video clip has several neighbours. Therefore, a set of bipartite graphs can be constructed according to the mapping set. According to the concept of combination, the number of 1:1 mapping bipartite graphs is $b_1 \times b_2 \times \ldots \times b_m$. Below are several 1:1 mapping bipartite graphs constructed based on the *thNN* searching.

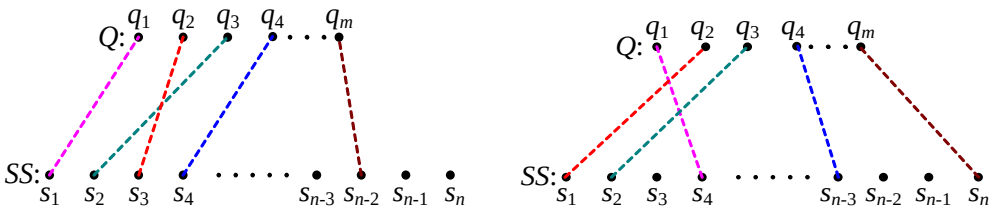


Figure 6.4 1:1 mapping bipartite graphs

Section 6.2.3 Maximum Size Matching (MSM) algorithm

A 1:1 mapping, however, is not enough. The size of the 1:1 mapping (the number of edges in the corresponding bipartite graph) is attractive to us. For instance, the one edge mapping bipartite graph <q_1, s_1 (q_1, s_1)> is obviously not the satisfied mapping we would like to obtain in video clip detection. In order to obtain the maximum size mapping in the mapping set we have already obtained, the Maximum Size Matching (*MSM*) algorithm [Shi2004] is employed to produce a set of 1-1 mappings between *Q* and *SS* with the maximum size for the corresponding *BG*. Note that the output of *MSM* in general is not unique. For instance, the two 1:1 mappings in figure 6.4 are both the maximum mappings from figure 6.3. The typical Hungarian Algorithm [Kuh1955, mun1957, AMO1993] is conducted for Maximum Size Matching. Firstly, several terms related to the bipartite graph matching should be noted:

Definition 6.3 Matching: Given a bipartite graph *G*, a *matching* is a subgraph of *G*, where any pair of edges has no common vertex. Or we can say the degree of any vertex is no larger than 1.

Definition 6.4 Maximum Matching: is a matching that contains the largest number of edges.

Definition 6.5 Alternating Path: is a path where the matched edges and unmatched edges exist alternatively.

Definition 6.6: Augmenting Path: an augmenting path is a particular alternating path that both starts from and ends at the unmatched vertices.

Algorithm 6.2: Hungarian Algorithm for Maximum Size Matching

Input:
bipartite graph $BG = <Q, SS, E>$
Output:
1:1 Maximum Size Matching $MSM(Q, SS)$
Initialization:
$MSM(Q, SS) = null$
Updating:
For $i = 1$ to m do
Start from q_i, searching for the augment path AP.
$MSM(Q, SS) = MSM(Q, SS) \cup AP$

Based on the above definitions, the Hungarian algorithm can be defined as in algorithm 6.2:

For each given state-sequence *SS*, algorithm 6.2 produces a corresponding set of 1-1 matching $MSM(Q, SS)$ between Q and SS with the maximum size. Therefore, if we denote the set of such matching between Q and all SS_j in D as $MSM(Q, D)$, we have:

$$MSM(Q,D) = \bigcup_{j=1}^{L} MSM(Q, SS_j) \tag{6-1}$$

The remaining main problem is then to develop an appropriate similarity measurement for searching the corresponding optimal matching.

Section 6.3 Hybrid Similarity Model

As mentioned earlier, for a given matching $M \in MSM(Q, D)$, both temporal similarity and non-temporal similarity should be taken into account. On one hand, the non-temporal similarity is defined according to the Euclidean distance between each mapping:

Section 6.3.1 Non-temporal similarity

The non-temporal similarity is measured by the total similarity which is in inverse proportion to the Euclidean distance between each matched state pair.

$$Sim_{NT} = \sum (1 - dis(q_i, s_j)) \Big/ (\sqrt{d}\,|Q|) \tag{6-2}$$

where $dis(q_i, s_j)$ denotes the Euclidean distance between each matched state pair q_i and s_j (which has achieved during *kNN* search) in the matching M and d denotes the feature dimension of each state. Obviously, the similarity value falls into [0, 1].

On the other hand, as the distinctive feature of time-series data, temporal similarity needs special treatments with respect to the three measurements described in the following three sections.

Section 6.3.2 Temporal order similarity

There may be some pairs of state-sequences with the same non-temporal similarity but with different temporal order. Here, we shall use the idea of LCSS [VGK2002] to measure temporal order similarity. However, in existing normal LCSS based formalisms, the typical reordering situations inversion in time-series data has been

neglected. In order to catch such types of reordering, we define the temporal order similarity as below:

$$Sim_{TO} = \max(LCS(M), LCS(\overleftarrow{M})) / |Q| \tag{6-3}$$

which takes into account both normal order and inverse order.

Section 6.3.3 Temporal alignment similarity:

In normal LCSS formalisms, in subsequence matching, unmatched states are simply skipped regardless how many of them there are. ED [ALK1999] is an alternative measurement that distinguishes the number of unmatched states that are skipped. However crossover, which should be compatible since it is ubiquitous, is not allowed in ED since it only matches in the single forward direction. Following the approach proposed in [SSHZ2009], we define the following temporal alignment similarity:

$$Sim_{TA} = 2|M| / (|Q| + |SS|) \tag{6-4}$$

which takes into account the number of unmatched states and accepts crossover.

Section 6.3.4 Temporal concentration similarity:

It is easy to see that the distribution of matched (or unmatched) states and the internal temporal distance (or similarity) is ignored in Sim_{TA}. For instance, by Eq.(6-4), sequence [1, 2, 3, 4, 5] will be taken as having the same similarity with [1, a, a, 2, 3, 4, a, 5], [1, 2, a, 3, a, 4, a, 5] and [1, b, c, 2, 3, 4, d, 5]. In addition, the duration of various times, over which the corresponding states are associated with, is not addressed in (6-27). Here, we introduce a similarity measurement to govern such temporal concentration. In what follows in this paper, we use *CD* and *DD* to denote the Concentration similarity Degree and the Discrete similarity Degree:

$$CD = Dur(CMS_1) + \sum_{i=2} (Dur(CMS_i) \Big/ \sum_{t=1}^{i} Dur(CMS_t)) \tag{6-5}$$

$$DD = \sum_{i} (\lambda_i Dur(CUS_i) \Big/ \sum Dur(CUS)) \tag{6-6}$$

where CMS_i and CUS_i are defined as "Continuous Matched Subsequences" and "Continuous Unmatched Subsequence", respectively, in descending ordered with respect to the length of these subsequences; and $Dur(CMS)$ and $Dur(CUS)$ denote the list of the duration of each continuous subsequence in *CMS* and *CUS*, respectively. λ represents the internal temporal distance with respect to each adjacent continuous matched and unmatched subsequences. In fact, if $CUS_i = [s_t, \ldots, s_p]$

$$\lambda = \begin{cases} \sum_{i=t}^{p} dis(s_{p+1}, s_i) \Big/ |CUS_i| & if \ \ t = 1 \\ \sum_{i=t}^{p} dis(s_{t-1}, s_i) \Big/ |CUS_i| & if \ \ p = length(SS) \\ \sum_{i=t}^{p} (dis(s_{t-1}, s_i) + dis(s_{p+1}, s_i)) \Big/ 2|CUS_i| & else \end{cases} \tag{6-7}$$

In order to reduce the computing complexity, we replace s_{t-1} and s_{p+1} by their corresponding query states in *Q* since the Euclidean distance in Eq.(6-2) between each state in *Q* and a state in *SS* has been achieved in the *kNN* search stage.

The temporal concentration similarity can be defined as:

$$Sim_{TC} = (CD - DD)/|Q| \tag{6-8}$$

Section 6.3.5 Hybrid Similarity Model

Normally, the overall similarity can be simply defined as the average of individual similarities. However, as we have argued earlier, the individual similarity measurements introduced in this paper have various features. In fact, while the non-temporal similarity and the temporal similarity may be treated in parallel, the three temporal similarities are progressive one after the other. Therefore, it is not appropriate

to simply accumulate all of them together. In what follows, we use a hybrid approach to combine the four similarity measurements.

Step 1: reorder $MSM(Q, D)$ as $MSM(Q, D)$' by Sim_{TO}, Sim_{TA}, and Sim_{TC}:

Firstly, reorder it by the Sim_{TO}; then for the matchings with the same Sim_{TO}, reorder them by Sim_{TA}; analogously, reorder by Sim_{TC} if there some matchings with the same Sim_{TA} exist.

Step 2: Integrate temporal similarity; get the integrated temporal similarity $Sim_{TS} = Adjust(Sim_{TO})$. For those $\mu = j\text{-}i\text{+}1$ matchings $[M'_i, \ldots, M'_j]$ with the same Sim_{TO}, evenly stretch their similarities into $[Sim_{TO} + \sigma/2, Sim_{TO} - \sigma/2]$ where σ denotes the adjust operator defined as below:

$$\sigma = \begin{cases} (Sim_{TO_{i-1}} - Sim_{TO_{j+1}})/3\mu & if \ \ i \neq 1, j \neq x \\ Sim_{TO_{j+1}}/2\mu & if \ \ i = 1 \\ Sim_{TO_{i-1}}/2\mu & if \ \ j = x \end{cases} \tag{6-9}$$

Step 3: Overall similarity; reorder $MSM(Q, D)$' as $MSM(Q, D)$'' in terms of overall similarity *Sim*, which is defined as the average of the non-temporal similarity and integrated temporal similarity:

$$Sim = (Sim_{NT} + Sim_{TS})/2 \tag{6-10}$$

Section 6.4 Experimental Results

The proposed method was evaluated using a real-life video database that consisted of 6 classes of video clip including news, basketball sports, education, scene, animation and MTV, each of which is in MPEG-1format with frame rate of 30 fps and with average duration of 2.9 minutes. For each key-frame, the 64-dimensional color histogram is extracted as the corresponding feature vector which has been normalized into [0, 1] afterwards. The detailed information on the video clip database is reported in table 6.2.

Table 6.2 Video clip database structure

database	Duration(hours)	Num of clips	Num of key-frames
news	5.5	90	4560
basketball	4.4	120	2359
education	3.9	80	3096
scene	3.2	100	2547
animation	7.6	120	5213
MTV	7.5	150	6864
TOTAL	**32.1**	**660**	**24639**

Section 6.4.1 Set up

The database consists of 660 video clip state-sequences with average length of 37.3hrs for each, including 6 different classes (100 examples each): In order to avoid the influence of segmenting error to the proposed similarity model, we shall use the original database in the form of individual 660 key-frame sequences as the training data. Several query sets are reconstructed as following:

Original Query Set (*OQS*): which consists of 180 state-sequences (the first 30 state-sequences from each class);

Reordered Query Set (*RQS*): each state-sequence of this set is in α percent reordered (in inverse order while α=1) from the corresponding state-sequence in *OQS*;

Shortened Query Set (*SQS*): each state-sequence is with length of (1- β) × (number of key-frames), by deleting β × (number of key-frames) states evenly, from the beginning and from the end of the corresponding state-sequence in *OQS*;

Noised Query Set (*NQS*): each state-sequence of this set is obtained by means of adding a Gaussian noise to each state-sequence in *OQS.*

For each query state-sequence, by means of following the procedure presented in section 6.2, we obtain a set of optional matching in the training database, and according to the hybrid similarity model proposed in section 6.3, we then calculate the overall

similarity respectively. The precision is defined as the ratio of the number of state-sequences with the same class as the query state-sequence out of the first 100 optimal matching in *MSM*(*Q, D*)''. We focus on the performance of our similarity model compared with that of [SSHZ2009] (named as "Shen"), which is just simply defined by the average of its individual similarity measurements. Meanwhile, another two models which employ ED and LCSS as temporal similarity have been tested respectively.

Section 6.4.2 Effectiveness of d_{max}

Figure 6.5 shows the precision using the *OQS* dataset with different d_{max} in *thNN* search. We can see that there is no distinct influence of d_{max} within [0, 0.3]. In order to reduce the complexity of our matching system, we default d_{max} = 0.3 if not specified.

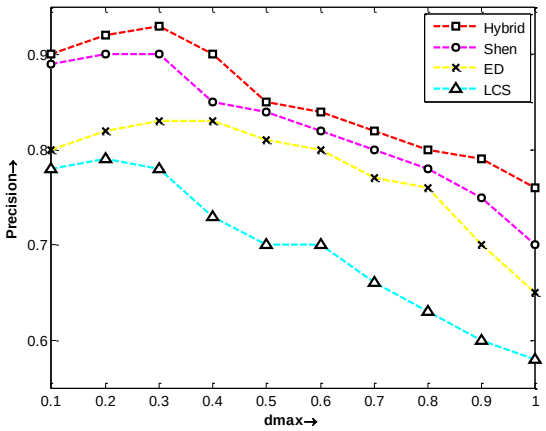


Figure 6.5 Precision of *OQS* against d_{max}

Section 6.4.3 Effectiveness of α

Figure 6.6 shows the precision on the *RQS* dataset against α. In order to reveal the performance of the progressive temporal similarity measurement we proposed in this paper, we omit the non-temporal similarity in each method. From the figure we can see that, in our method, the precision has an approximate quadratic distribution with respect to α, which means it can better detect the reordered state-sequences compared to the other approaches.

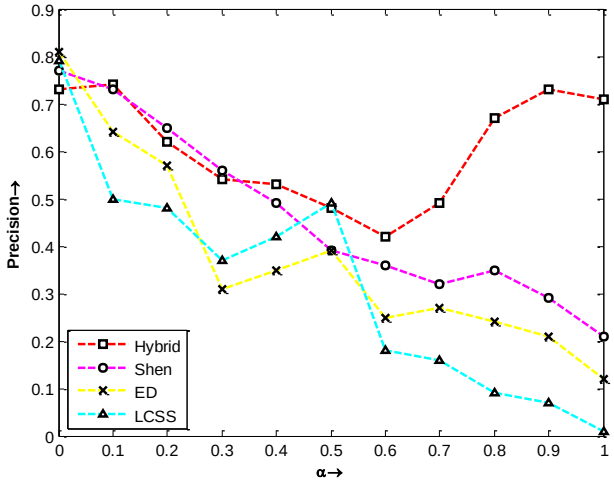
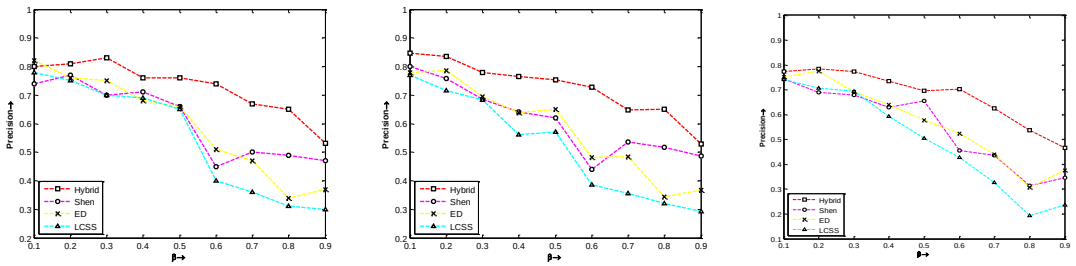


Figure 6.6 Precision of *RQS* against α

Section 6.4.4 Effectiveness of β

To evaluate the effect of β, we formed the *SQS* dataset by deleting β*60 states in different positions: evenly, from the beginning and the end. Figure 6.7 shows the matching results against different β. Generally speaking, our method is more robust than others no matter whether the state-sequences are shortened evenly, from the beginning or from the end. The precision drops much more slowly in our method especially for $\beta \in [0.1, 0.5]$. In addition, according to our statistic, the query set shortened from the beginning has a slightly higher precision than the other two sets shortened evenly and at the end in our similarity model. Generally speaking, the position (where being shortened) does not affect the precision very much in any similarity model.



(a) Evenly (b) From the beginning (c) From the end

Figure 6.7 Precision of *SQS* against β

Section 6.4.5 Robustness

Figure 6.8 shows the results of data seeded with Gaussian noise with different means ([0, 2]) and variances ([0.1, 1]). Visually, our method has higher precision and smaller fluctuation. Table 6.7 below shows the average mean and standard deviation (STD) of each subfigure in Figure 6.8.

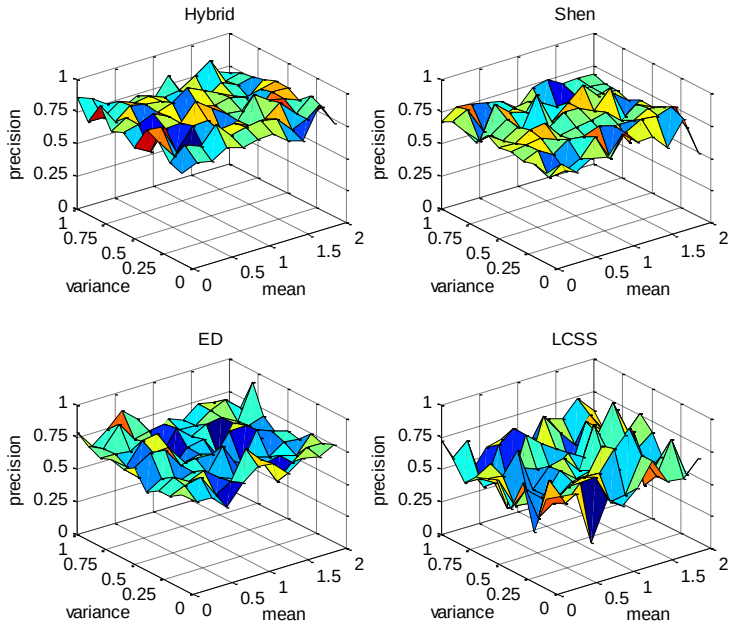


Figure 6.8 Precision with Gaussian noise against mean and variance

Table 6.3 Statistic of the precision of noised query set

	Hybrid	**Shen**	**ED**	**LCSS**
Mean (%)	77.69	70.25	66.72	58.28
STD	0.0624	0.0735	0.0809	0.1228

In summary, the hybrid similarity model has a satisfactory performance on video copy detection. Furthermore, it can handle the reorder edition in video clips and is robust to the noise. Since the similarity factors have just been combined linearly, it would be worthwhile developing a non-linear combination in the future.

CHAPTER 7 CONCLUSION AND FUTURE WORK

Section 7.1 Conclusion

To sum up, this thesis has designed a general framework for state-sequence matching particularly with the formal characterization of time-series and state-sequence and a general similarity measurement. In addition, two cases of state-based temporal pattern recognition have been investigated and explored.

The evolution of the representation of time-series and conventional similarity measurements have been reviewed in detail. The relevant problems have been pointed out as motivation of this thesis: the general framework with the formal characterization of time-series and state-sequence as well as the general similarity measurement.

The main findings with respect to the research issues listed in section 1.2 are summarized as following:

1). A formal characterization of time-series and state-sequences has been presented for both complete and incomplete situations, where the time-series is formalized as a tetrad (T, R, T_{dur}, T_{gap}) that denotes the temporal order of time-elements, the temporal relationship between time-elements, the temporal duration of each time-element and the temporal gap between adjacent time-elements respectively. It is powerful enough to describe the state-sequences with both non-temporal information and rich temporal information.

2). The General Similarity Measurement (GSM) has been designed for state-sequence matching. It takes into account both non-temporal and rich temporal aspects, including temporal order, as well as temporal duration and

temporal gap. The versatile property of the proposed GSM has been verified by the means of deducing the conventional similarity measurements as its special cases. Experimental results on 6 benchmark datasets have demonstrated that it can address the most general problems in matching time-series data with rich temporal information. When specified as a real-penalty similarity measurement, GSM can distinguish the distance caused by various states in the same operation and filter out noise that may push the distance at an abnormal level if specified as a binary-value similarity measurement. In particular, a new LCS-based similarity measurement named Optimal Temporal Common Subsequence (OTCS) has been proposed as the special case of GSM. In OTCS, the continuous duplicated states are counted as the same state with different temporal duration. The advantage of OTCS has been verified by both the sample evolution and the experiments on the 6 benchmark datasets.

3). The basketball zone-defence detection system has been investigated as a case study of state-based temporal pattern recognition. On one hand, we have extended the Laplacian Matrix-based algorithm to take account of the effects from zoom and single defender's translation in zone-defence graph matching. A set of character-angle based features was proposed to describe the zone-defence graph. It can describe the structure relationship between defender-lines for basketball zone-defence, and has a robust performance in both simulation and real-life applications especially when disturbance exists.

4). The video copy detection system has been investigated as another case study of state-based temporal pattern recognition. The state-sequence matching problem has been represented by bipartite graph matching problem. A hybrid similarity model addressing both non-temporal and temporal relationship between state-sequences has been proposed, where the non-temporal similarity has been defined in form of Euclidean distance, whilst the temporal similarity has been constructed with temporal order similarity, temporal alignment

similarity and temporal concentration similarity. The experimental results on the real-life video database have demonstrated that the proposed model is robust to states alignment with various numbers and different values, as well as various reordering including inversion and crossover.

Section 7.2 Future Work Discussion

In the General Similarity Measurement (GSM) as well as the special Optimal Temporal Common Subsequence (OTCS) case, the parameter (the values of weights) selection is a vital and arduous task. How to automatically select the optimal values for the weights remains one a research task for the future. Furthermore, more intelligent computation of the temporal durance difference and temporal gap difference also presents interesting future work. In addition, in order to be applied to large scale databases, it is very important to adopt proper pruning strategies to improve efficiency, which will also be part of our future work.

In basketball zone-defence detection, the extended Laplacian Matrix-based algorithm only takes account of the effects from zoom and single defender's translation in zone-defence graph matching. However, the effect from rotation is ubiquitous in zone-defence graph matching. As an area of future work, it would be worthwhile to take account of the effects from rotation in basketball zone-defence detection. Furthermore, the basketball database is still small in our experiments. It would be necessary to expand the size of the dataset to further explore both the non-temporal and temporal relationships between state-sequences of basketball zone-defence, and therefore to obtain the best defence and attacking strategy. In addition, both the extended Laplacian Matrix-based algorithm and the Character-Angle based feature have been tested on basketball zone-defence videos. Therefore, as future work, they may be tested on other team sports games such as football, volleyball, and so on.

In the hybrid similarity model, the non-temporal similarity and temporal similarity including temporal order similarity, temporal alignment similarity and temporal

concentration similarity have been combined by means of linear accumulation, which is a simple but inappropriate method of aspect combination. With respect to future work, it would be interesting to explore more appropriate combination strategies. Meanwhile, it is hoped that this model can provide a steady usage with regards to larger time-series databases and real-life applications such as Content-based Video Retrieval (CBVR), which may also be an area for future work.

REFERENCE:

[All1984] J. Allen: “Towards a General Theory of Action and Time”. Artificial Intelligence, 23, 1984, pp: 123-154.

[ALK1999] D. Adjeroh, M. Lee and I. King: “A distance measure for video sequences”. Computer Vision and Image Understanding, 75(1-2), 1999, pp: 25-45.

[ABCB2003] J. Assfalg, M. Bertini, C. Colombo, A. Bimbo and W. Nunziati: “Semantic annotation of soccer videos: automatic highlights identification”. Computer Vision and Image Understanding, 92(2-3), 2003, pp: 285-305.

[AFS1993] R. Agrawal, C. Faloutsos and A. Swami: “Efficient similarity search in sequence databases”. In Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO’93), Springer Press, Chicago, Illinios, USA, Oct. 13-15, 1993, pp: 69-84.

[AH1989] J. F. Allen and P. J. Hayes: “Moments and Points in an Interval-based Temporal-based Logic”. Computational Intelligence, 5(4), 1989, pp: 225-238.

[ALK1999] D. A. Adjeroh, M. C. Lee, and I. King: “A distance measure for video sequences”. Computer Vision and Image Understanding, 75 (1-2), 1999, pp: 25–45.

[AMO1993] R. Ahuja, T. Magnanti and J. Orlin: “Network Flows”. Prentice Hall, 1993

[APPK2008] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos: “Approximate Embedding-Based Subsequence Matching of time-series”. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD’08), Vancouver, BC, Canada, Jun. 9–12, 2008, pp: 365-378.

[AS1995] R. Agrawal and R. Srikant: “Mining Sequential Patterns”. In Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan, March 6-10, 1995, pp: 3-14.

[Bel1957] R. Bellman: “Dynamic Programming”. Princeton University Press, Princeton, NJ, 1957.

[Bru1972] B. C. Bruce: "A Model for Temporal References and Application in a Question Answering Program". Artificial Intelligence, 3, 1972, pp: 1-25.

[Ben1983] J. V. Benthem: “The Logic of Time”. Kluwer Academic, Dordrech, 1983.

[Bee1992] P. V. Beek: “Reasoning About Qualitative Temporal Information”. Artificial Intelligence, 58, 1992, pp: 297-326.

[BABST2007] A. Bagdanov, M. Bertini, A. Bimbo, G. Serra and C. Torniai: “Semantic annotation and retrieval of video events using multimedia ontologies”. In Proceedings of 1st IEEE International Conference on Semantic Computing (ICSC’07), Irvine, California, Sep. 17-19 2007, pp: 713-720.

[BC1996] D. J. Berndt and J. Clifford: “Finding patterns in time series: a dynamic programming approach”. Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Menlo Park, CA. 1996, pp:

229-248.

[BCB1999] G. Baldi, C. Colombo, A. Bimbo: "A compact and retrieval-oriented video representation using mosaics". In Proceedings of 3rd International Conference on Visual Information Systems (VISual'99), Springer Lecture Notes on Computer Science, Amsterdam, The Netherlands, June 2-4, 1999, pp: 171–178.

[BHR2000] L. Bergroth, H. Hakonen and T. Raita: "A Survey of Longest Common Subsequence Algorithms". In Proceedings of 7th IEEE International Symposium on String Processing and Information Retrieval (SPIRE'00), A Coruña, Spain, Sep. 27-29, 1999, pp: 39–48.

[BKK1999] N. Babaguchi, Y. Kawai and T. Kitahashi: "Event Based Video Indexing by Intermodal Collaboration". In Proceedings of 1st International Workshop on Multimedia Intelligent Storage and Retrieval Management (MISRM'99) in conjunction with ACM Multimedia Conference 1999, Orlando, Oct. 30th, 1999, pp: 1-9.

[BKSS1990] N. Beckmann, H. Kriegel, R. Schneider and B. Seeger: "The r*-tree: An efficient and robust access method for points and rectangles". In Proceedings of the International Conference on Management of Data (SIGMOD'99), ACM Press, Atlantic City, NJ, May 23-25, 1990, pp: 322-331.

[CHTH2005] S. Chin, C. Huang, C. Tang, C. Hung: "An Application Based on Spatial-Relationship to Basketball Defensive Strategies". Embedded and Ubiquitous Computing (EUC) Workshops, Nagasaki, Japan, Dec 8-9, 2005, pp: 180–188.

[CN2004] L. Chen and R. Ng: "On the Marriage of LP-Norm and Edit Distance".

In Proceedings of the International Conference on Very Large Data Bases, Toronto, Canada, Aug. 29 – Sep.3, 2004, pp: 792-801.

[COO2005] L. Chen, M.T. Ozsu and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories". In Proceedings of the 24th International Conference on Management of Data (SIGMOD'05), Baltimore, Maryland, USA, Jun. 13-16, 2005, pp: 491-502.

[CZKA1996] Y. L. Chang, W. Zeng, I. Kamel and R. Alonso: "Integrated Image and Speech Analysis for Content-based Video Indexing". In Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan, Jun. 17-23, 1996, pp: 03-06.

[DGM1997] G. Das, D. Gunopulos and H. Mannila: "Finding similar time series". In Proceedings of 1st the European Symposium on Principles of Data Mining and Knowledge Discovery from Databases (ECML-PKDD), Trondheim, Norway, Jun 24-27, 1997, pp: 88-100.

[EBMM2003] A. A. Efros, A. C. Berg, G. Mori and J. Malik: "Recognizing action at a distance", In Proceedings of 9th IEEE International Conference on Computer Vision (ICCV'03), Nice, France, Oct. 13-16, 2003, vol. 2, pp: 726–733.

[FM2008] M. M. Fuad and P. F. Marteau: "The Multi-resolution Extended Edit Distance Metric". In Proceedings of 3rd International ICST Conference on Scalable Information Systems, Vico Equense, Italy, Jun. 4–6, 2008, pp: 1-6.

[FRM1994] C. Faloutsos, M. Ranganathan & Y. Manolopoulos: "Fast subsequence matching in time-series databases". In Proceedings of

the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minneapolis, Minnesota, USA, May 25-27, 1994, pp: 419-429.

[Gal1990] A. Galton: "A Critical Examination of Allen's Theory of Action and Time". Artificial Intelligence, 42, 1990, pp: 159-188.

[Gro1999] R. Groth: "Data Mining: Building Competitive Advantage". Prentice-Hall Inc., 1999.

[GS1999] V. Guralnik and J. Srivastava: "Event detection from time series data". In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99), San Diego, CA, Aug 15-18, 1999, pp: 33-42.

[GS2004] A. Gao, H. Siegelmann: "Time-Warped Longest Common Subsequence Algorithm for Music Retrieval". In Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04), Barcelona, Spain, Oct. 10-14, 2004.

[GSC1995] Y. H. Gong, L. T. Sin, C. H. Chuan et al.: "Automatic Parsing of TV Soccer Programs". In Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS'95), 1995, pp: 167-174.

[HR2007] Q. Hua, Y. Rui: "Optimizing Multi-Graph Learning Towards A Unified Video Annotation Scheme". In Proceedings of the ACM International Conference on Multimedia (ACM MM'07), Augsburg, Germany, Sep. 24 -29, 2007, pp: 17-26.

[JAS2002] M. D. Jaere, A. Aamodt and P. Skalle: "Representing Temporal Knowledge for Case-Based Prediction". In Proceedings of the 6th

European Conference on Advances in Case-Based Reasoning (ECCBR'02), Aberdeen, Scotland, UK, Sep 4-7, Vol. 2416, 2002, pp: 174 - 188.

[KC2005] Y. Kim and T. Chua: "Retrieval of News Video Using Video Sequence Matching". In Proceedings of the 11th International Conference on Multimedia Modelling (MMM'05), IEEE Press, Melbourne, Australia, Jan. 12-14, 2005, pp: 68 – 75.

[KP1998] E. Keogh and M. Pazzani: "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback". In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, USA, Aug 27-31, 1998, pp: 239-241.

[KP2000] E. Keogh and M. Pazzani: "Scaling up dynamic time warping for data mining applications". In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, Aug. 20-23, 2000, pp: 285-289.

[Kuh1955] H. W. Kuhn: "The Hungarian Method for the assignment problem". Naval Research Logistics Quarterly, 1955, Vol.2, pp: 83-97.

[Lev1965] V. I. Levenshtein: "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals". Soviet Physics Doklady, 1965, Vol.10, No.8, pp: 845-848.

[Lad1987] P. Ladkin: "Models of axioms for time intervals". In Proceedings of the 6th National Conference on Artificial Intelligence (AAAI'87), Seattle, Washington, USA, Jul. 13–17, 1987, pp: 234-239.

[LTWB2005] D. Liang, Q. Tong, N. Wang, W. Bao and L. Qu: "A Laplacian Matrix

Based Algorithm for Image Matching". Computer Engineering and Applications, 2005, Vol. 41(36), pp: 31-38.

[LWH2003] Y. Luo, T. D. Wu, J. N. Hwang: "Object-based analysis and interpretation of human motion in sports video sequences by dynamic Bayesian networks". Computer Vision and Image Understanding - Special isssue on video retrieval and summarization, 2003, Vol. 92 (2-3), pp: 196–216.

[LXYC2006] S. Liu, M. Xu, H. Yi, L. Chia and D. Rajan: "Multimodal Semantic Analysis and Annotation for Basketball Video" EURASIP Journal on Applied Signal Processing, 2006, pp: 1-13.

[Mar2008] P. F. Marteau: "Time Warp Edit Distances with Stiffness Adjustment for Time Series Matching". IEEE Transaction on Pattern Analysis and Machine Intelligence, 2009, Vol.31 (2), pp: 306-318.

[MBG2008] B. Marco, A. Bimbo and S. Giuseppe: "Video Event Annotation using Ontologies with Temporal Reasoning". In Proceedings of 4th Italian Research Conference on Digital Library Systems (IRCDL), Padova, Italy, Jan. 24-25, 2008, pp: 24-25.

[MBZ2008] J. Ma, R. Bie, G. Zhao: "An ontological Characterization of Time-series and State-sequences for Data Mining". In Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Jinan, China, Oct. 18-20, 2008, pp: 325-329.

[MH2006] J. Ma and P. Hayes, "Primitive Intervals Vs Point-Based Intervals: Rivals or Allies?" The Computer Journal, 2006, Vol.49 (1), pp: 32-41.

[MK1994] J. Ma and B. Knight: "A General Temporal Theory". The Computer Journal, 1994, Vol.37 (2), pp: 114-123.

[MM2008a] M. M. Muhammad and P. F. Marteau: "The Extended Edit Distance Metric". The 6th International Workshop on Content-Based Multimedia Indexing (CBMI), London, UK, Jun. 18-20, 2008, pp 242-248.

[MM2008b] F. M. Muhammad and P. F. Marteau: "The Multi-resolution Extended Edit Distance Metric". The 3rd International ICST Conference on Scalable Information Systems, Vico Equense, Italy, Jun. 4-6, 2008, pp: 1-6.

[Mun1957] J. Munkres: "Algorithms for the Assignment and Transportation Problems". Journal of the Society of Industrial and Applied Mathematics, 1957, Vol.5 (1), pp: 32-38.

[MWH2002] Y. -S. Moon, K. -Y. Whang, and W. S. Han: "General Match A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows". Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD'02), Madison, Wisconsin, USA, Jun. 4-6, 2002, pp: 382-393.

[MWL2001] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh. "Duality-based subsequence matching in time-series databases". In Proceedings of the 17th International Conference on Data Engineering (ICDE'01), Apr. 2-6, 2001, Heidelberg, Germany, pp: 263-272.

[MZH2007] J. Ma, G. Zhao, E. Hancock: "A Navigation-based Algorithm for Matching Scenario Patterns". In Proceedings of International Conference on Artificial Intelligence and Pattern Recognition (AIPR'07), Orlando, Florida, USA, Jul.9-12, 2007, pp: 151-157.

[NSG2001] S. Nepal, U. Srinivasan and G. Reynolds: "Automatic detection of

goal segments in basketball videos". In Proceedings of the 9th ACM International Conference on Multimedia (MULTIMEDIA'01), Ottawa, Canada, Sep. 30 – Oct. 5, 2001, pp: 261-269.

[PKKV2009] M. Perse, M. Kristan, S. Kovacic, G.Vuckovic and J. Pers: "A trajectory-based analysis of coordinated team activity in a basketball game". Computer Vision and Image Understanding, 2009, Vol.113 (5), pp: 612-621.

[PVS2001] H. Pan, B. P. Van and M. I. Sezan: "Detection of slowmotion replay segments in sports video for highlights generation". In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2001, Vol.3, pp: 1649-1652.

[RGA2000] Y. Rui, A. Gupta and A. Acero: "Automatically Extracting Highlights for TV Baseball Programs". In Proceedings of the 8th ACM International Conference on Multimedia, Los Angeles, CA, USA, Oct. 30 - Nov. 3, 2000, pp: 105-115.

[SC1978] H. Sakoe, and S. Chiba: "Dynamic programming algorithm optimization for spoken word recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, 26 (1), 1978, pp. 43- 49.

[Shi2004] D. Shier: "Matchings and assignments". In Handbook of Graph Theory, J. L. Gross and J. Yellen, Eds. CRC Press, 2004, pp: 1103–1116.

[SFY2007] Y. Sakurai, C, Faloutsos and M. Yamamuro: "Stream monitoring under the time warping distance". IEEE 23rd International Conference on Knowledge and Data Engineering, Istanbul, Turkey, Apr.17-20, 2007, pp: 1046-1055.

[SHSZ2008] J. Shao, Z. Huang, H. Shen, X. Zhou, E. Lim and Y. Li: "Batch nearest neighbour search for video retrieval". IEEE Transactions on Multimedia, 2008, Vol.10 (3), pp: 409-420.

[SSHZ2009] H. Shen, J. Shao, Z, Huang and X. Zhou: "Effective and Efficient Query Processing for Video Subsequence Identification". IEEE Transactions on Knowledge and Data Engineering, 2009, Vol.21 (3), pp: 321-334.

[TSKR2000] Y. P. Tan, D. D. Saur, S. R. Kulkarni and P. J. Ramadge: "Rapid estimation of camera motion from compressed video with application to video annotation". IEEE Transactions on Circuits and Systems for Video Technology (CSVT), 2000, Vol. CSVT-10, pp: 133-146.

[UFF2005] R. Urtasun, D. J. Fleet and P. Fua: "Monocular 3d tracking of the golf swing". IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, Jun. 20-26 2005, vol. 2, pp: 932–938.

[Vil1994] L. Vila: "A survey on temporal Reasoning in Artificial Intelligence", AI Communication, 1994, Vol.7, pp: 4-28.

[VGK2002] M. Vlachos, D. Gunopulos and G. Kollios: "Discovering similar multidimensional trajectories". In Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, CA, USA, Feb 26-Mar 1, 2002, pp: 673–684.

[VHGK2003] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E.J. Keogh, "Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures". In Proceedings of ACM Special Interest Group Knowledge Discovery and Data Mining (SIGKDD'03), Washington,

DC, USA, Aug. 24 – 27, 2003, pp: 216-225.

[Wan2007] H. Wang, "All Common Subsequences". In Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Hyderabad, India, Jan. 6-12, 2007, pp: 635-640.

[WSZ2004] H. Wu, B. Salzberg, and D. Zhang: "Online Event-Driven Subsequence Matching over Financial Data Streams". In Proceedings of International Conference Management of Data (SIGMOD'04), ACM Press, Paris, France, Jun. 13-18, 2004, pp: 23-34.

[Xu2001] P. Xu: "Algorithms and systems for segmentation and structure analysis in soccer video". In Proceedings of IEEE International Conference on Multimedia and Expo (ICME'01), Tokyo, Japan, Aug. 22-25, 2001, pp: 184–187.

[XC2004] H. Xu and T. Chua: "The fusion of audio-visual features and external knowledge for event detection in team sports video". In Proceedings of the 6th ACM SIGMM International workshop on Multimedia Information Retrieval (MIR'04), New York, NY, USA, Oct.15-16, 2004, pp: 127-134.

[XDC2004] M. Xu, L. Duan, J. Cai, L.Chia, C. Xu and Q. Tian: "HMM-Based Audio Keyword Generation". In Proceedings of the 5th Pacific Rim Conference on Multimedia (PCM'04), Tokyo, Japan, Nov.30-Dec.3, 2004, pp: 566-574.

[YJF1998] B. Yi, H. Jagadish and C. Faloutsos: "Efficient retrieval of similar time sequences under time warping". In Proceedings of the 14th International Conference on Data Engineering (ICDE'98), Orlando, FL, USA, Feb. 23-27, 1998, pp: 201-208.

[YX2003] H. J. Ye, G. Y. Xu: “Fast search in large database using vector quantization”. In Proceedings of International Conference on Image and Video Retrieval (CIVR’03), Berlin: Springer-Verlag, Urbana-Champaign, IL, USA, Jul. 24-25, 2003, pp: 477-487.

[ZMLP2009] A. Zheng, J. Ma, B. Luo and M. Petridis: “Temporal Pattern Recognition in Basketball Video Clips”. In Proceedings of 5th International Conference on Computer and Information Science (ICIS’09), Shanghai, China, Jun. 1-3, 2009, pp: 416"-421.

[ZMPT2009] A. Zheng, J. Ma, M. Petridis, J. Tang and B. Luo: “A Robust Approach to Subsequence Matching”. Studies in Computational Intelligence (SCI), 2009, Vol.253, pp: 39-49.

[ZS2003] Y. Zhu and D. Shasha: “Warping indexes with envelope transforms for query by humming”. In Proceedings of International Conference on Management of Data (SIGMOD’03), ACM Press, San Diego, California, USA, Jun.9-12, 2003, pp: 181–192.

Temporal Pattern Recognition in Video Clips Detection

Aihua Zheng[1,2] Jixin Ma[2] Bin Luo[1] Sulan Zhai[1] Jin Tang[1]
[1]*Anhui University, People's Republic of China*
[2]*The University of Greenwich, United Kingdom*
{a.zheng, j.ma}@gre.ac.uk luobin@ahu.edu.cn

Abstract

Temporal representation and reasoning plays an important role in Data Mining and Knowledge Discovery, particularly, in mining and recognizing patterns with rich temporal information. Based on a formal characterization of time-series and state-sequences, this paper presents the computational technique and algorithm for matching state-based temporal patterns. As a case study of real-life applications, zone-defence pattern recognition in basketball games is specially examined as an illustrating example. Experimental results demonstrate that it provides a formal and comprehensive temporal ontology for research and applications in video events detection.

Key words: algorithm, temporal pattern recognition, basketball zone-defence.

1. Introduction

Data mining is the process of finding trends and patterns in data [4]. Generally speaking, data mining requires some historical knowledge as for the internal temporal relationships of certain patterns. Therefore, temporal representation and reasoning is essential and ubiquitous for data mining and knowledge discovery. In fact, recognizing temporal patterns actually plays an important role in many applications such as prediction, forecast, explanation, diagnosis, history reconstruction, decision making, and so on, where the history of situations in terms of time-series of states is more vital than distinct states/processes or actions/events. For instance, in the area of medical information systems, a patient's medical history is obviously very important: to prescribe the right treatment, the doctor needs to analysis not only the patient's current state, but also his/her previous health situations, including: How long has the patient been ill? Did the patient have the same problem or relevant disease previously? Has the patient had some treatment already before seeing the doctor? Has the patient been allergic to any drugs in the past? Also, in weather forecast, to provide correct and accurate prediction, weather experts need to know not only the current weather parameters summarized as temperature, air pressure, precipitation amount, wind speed and residual snow/ice amount, but also the weather histories in terms of time-series of weather parameters over some certain prior periods, such as: How long did the heat wave last? Was there lightning before or during the rain? Did snow melt then refreeze? And so on. Similarly, in basketball games, to find correct zone-defence strategy detection, we need to know not only the current positions of each defender, but also their previous positions and movements, etc.

It has been noted that, time-series and sequences are important patterns in data mining and have attracted a lot of researchers' interests [3, 8, 9, 11, 13]. However, in most of those proposed formalisms, the fundamental time theories based on which time-series and sequences are formed up are usually not explicitly specified, where time-series and sequences are simply expressed as lists in the form of well-ordered indexes or as sequences of collection of observations, and so on. The formal characterizations with respect to the temporal basis are neglected, leaving some critical issues unaddressed.

In what follows in this paper, the formalism for formalizing time-series and state-sequences is briefly introduced in section 2. Based on this formalism, section 3 presents the computational technique and algorithm for matching state-based temporal patterns, illustrated by a real-life case study. Experimental results are provided, analyzed and evaluated in section 4, demonstrating the efficiency of the proposed technique and algorithm. Finally, section 5 provides a brief summary and concludes the paper.

2. The formalism

For general treatment, in this paper, we shall adopt the general time theory proposed in [10] as the temporal basis. The time theory takes a nonempty set of primitive time elements, with an *immediate predecessor* relation, Meets,

This research is supported in part by National Nature Science Foundation of China (No. 60772122)

over time elements, and a *duration assignment* function, Dur, from time elements to non-negative real numbers. If Dur(t) = 0, then t is called a point; otherwise, that is Dur(t) > 0, t is called an interval (detailed characterization of such a time theory is given in [10]).

Analogous to the 13 relations introduced by Allen for intervals [1,2], there are 30 exclusive temporal order relations over time elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations which relate points to points: {Equal, Before, After}
- Relations which relate points to intervals: {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations which relate intervals to points: {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations which relate intervals to intervals: {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

Based the above time theory, a time-series ts is defined as a vector of time-elements temporally ordered one after another [9]. Formally, a general time-series is characterized in terms of the following schema:

GTS1) $ts = [t_1, \ldots, t_n]$;

GTS2) $Meets(t_j, t_{j+1}) \vee Before(t_j, t_{j+1})$, for all $j = 1, \ldots, n-1$;

GTS3) $Dur(t_k) = d_k$, for some k where $1 \leq k \leq n$ and d_i is a non-negative real number.

N.B. : $Before(t_1, t_2) \Leftrightarrow \exists t(Meets(t_1, t) \wedge Meets(t, t_2))$

Generally speaking, a time-series may be incomplete in various ways. For example, if the relation between t_j and t_{j+1} is "Before" rather than "Meets", it means that the knowledge about the time-element(s) between t_j and t_{j+1} is not available. In addition, if $Dur(t_k) = d_k$ is missing for some k, it means that duration knowledge as for time-element t_k is unknown. Correspondingly, a complete time-series is defined in terms of the schema as below:

CTS1) $ts = [t_1, \ldots, t_n]$;

CTS2) $Meets(t_j, t_{j+1})$, for all $j = 1, \ldots, n-1$;

CTS3) $Duration(t_i) = d_i$, for all $i = 1, \ldots, n$, where d_i is a non-negative real number.

The validation of data is usually dependent on time. For instance, $1000 (Account Balance) can be valid before and on 1 January 2003 but become invalid afterwards. We shall use fluents to represent Boolean-valued, time-varying data, and denote statement "fluent f holds true over time t" by formula Holds(f, t):

(F1) $(f, t) \Rightarrow \forall t_1(Part(t_1, t) \Rightarrow Holds(f, t_1))$

That is, if fluent f holds true over a time element t, then f holds true over any part of t.

(F2) $\forall t_1(Part(t_1, t) \Rightarrow \exists t_2(Part(t_2, t_1) \wedge Holds(f, t_2))) \Rightarrow Holds(f, t)$

That is, if any part of time t contains a part of itself over which fluent f holds true, then f holds true over t.

Here, $Part(t_1, t_2)$ is the shorthand writing of $Equal(t_1, t) \vee Starts(t_1, t) \vee During(t_1, t) \vee Finishes(t_1, t)$.

(F3) $Holds(f_1, t) \vee Holds(f_2, t) \Rightarrow Holds(f_1 \vee f_2, t)$

That is, if fluent f_1 holds true over time t or fluent f_2 holds true over time t, then at least one of them holds true over time t.

(F4) $Holds(not(f), t) \Leftrightarrow \forall t_1(Part(t_1, t) \Rightarrow \neg Holds(f, t_1))$

That is, the negation of fluent f holds true over time t if and only if fluent f does not hold true over any part of t.

(F5) $Holds(f, t_1) \wedge Holds(f, t_2) \wedge Meets(t_1, t_2) \Rightarrow Holds(f, t_1 \oplus t_2)$

That is, if fluent f holds true over two time elements t_1 and t_2 that meets each other, then f holds over the ordered-union [10] of t_1 and t_2.

A state is defined as a collection of fluents. Following the approach proposed in [12], we shall use Belongs(f, s) to denote that fluent f belongs to the collection of fluent representing state s.

For the reason of simple expression, if $f_1, \ldots, f_m$ are all the fluents that belong to state s, we shall represent s as $<f_1, \ldots, f_m>$. Also, without confusion, we shall use formula Holds(s, t) to denote that s is the state of the world with respect to time t, provided that:

(F6) $s = <f_1, \ldots, f_m> \Rightarrow Holds(s, t) \Leftrightarrow Holds(f_1, t) \wedge \ldots \wedge Holds(f_m, t)$

That is, a state s holds true over time t if and only if every fluent in the s holds true over time t.

A state-sequence ss is defined as a list of states together with its corresponding time-series ts [9]. A general state-sequence is defined in terms of the schema as below:

GSS1) $ss = [s_1, \ldots, s_n]$;

GSS2) $Holds(s_i, t_i)$, for all $i = 1, \ldots, n$, where $[t_1, \ldots, t_n]$ is a time-series.

Finally, a state-sequence is defined as complete if and only if the corresponding time-series is complete.

3. States-based basketball zone-defence pattern recognition

As a popular worldwide sport game, basketball has led to various research interests, including basketball video retrieval, shot segmentation, event or highlight detection, semantic annotation, etc. In what follows in this paper, we shall focus on the so-called zone-defence pattern matching (or zone-defence strategy detection) as a real-life case study of states-based temporal pattern recognition.

3.1. Zone-defence state and graph

Zone-defence is a very common defence strategy in basketball. In particular, zone-defence uses basic principles to force opponents either in full-court, three-quarter-court, or half-court areas in order to upset their offense [6]. Comparing with man-to-man defence, zone pressure defence requires each defender guard his zone consistently. Fig.1 shows the ordinary positions of 5 defenders in 1-3-1 zone-defence:

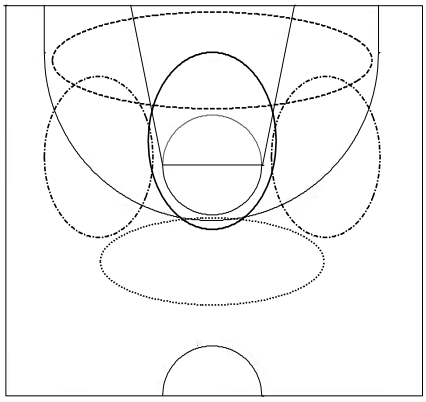


Fig.1. 5 defenders' positions in 1-3-1 zone-defence

Firstly, we premise that: 1) The defenders have adjusted to their best defensive positions at the moment when the ball is just to be passed or dribbled; 2) As the zone-defence strategy is to defence the offensive opponent to attack into interior playfield, we only consider the states when the ball is in the midfield, the wing and the corner.

According to these two premises, a basketball zone-defence video clip is structured by zone-defence states or so-called state-sequence: $SS = [S_1, \ldots, S_n]$, and $Holds(S_i,t_i)$ for $i = 1, \ldots, n$, where $[t_1, \ldots, t_n]$ is a time-series of the moments referred in premise 1).

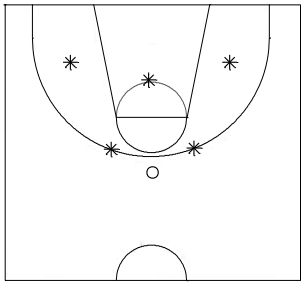
Following the conventional notations in graph theory, we represent a zone-defence graph as $G = <V_G, E_G>$, where V_G and E_G denote the set of the vertices (defenders' position) and the set of edges respectively, and $E_G \subseteq V_G \times V_G$. In particular, here, $|V_G| = 5$. The position of each defender is denoted by the horizontal and vertical coordinates of the corresponding vertex.

Obviously, each state S_i has its corresponding graph G_i, where $i = 1, \ldots, n$. In addition, we shall use the following vector $[ball_1, \ldots, ball_n]$ to record the ball's position of each state, where $ball_i \in \{midfield, wing, conner\}$ for $i = 1, \ldots, n$.

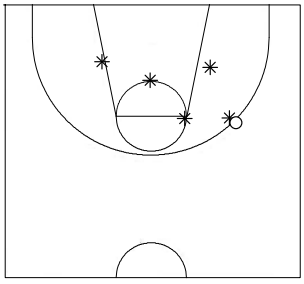
3.2. Standard zone-defence graph database

Zone-defence can be divided into various formations, including 2-3, 1-3-1, 1-2-2, 3-2, 2-2-1, 2-1-2 and 1-2-1-1 zone-defence strategies, where the first three have been noted as the most common ones employed in actual basketball games. In this paper, we shall focus on the first three formations.

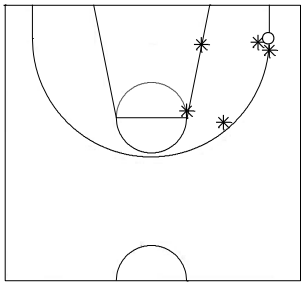
In the first place, we shall formulate the standard zone-defence graph database according to two famous basketball coaching web sides [5, 6]. For instance, a typical 2-3 zone-defence clip (state-sequence) for the ball from the state of setting-up in the midfield to the state of passing or dribbling to the wing and then to the corner can be presented in terms of the following 3 graphs as shown in Fig.2 (the star marks denote the 5 defenders and the circle denotes the ball):



(a) setting-up in the midfield



(b) passing or dribbling to the wing



(c) passing or dribbling to the corner

Fig.2. A sample clip (3 states) of 2-3 zone**-defence**

Table 1 below shows the number of our standard zone-defence graph database of different zone-defence strategies obtained from the two web sites [5, 6].

Table 1. The number of standard zone graphs

zone-defence	2-3	1-3-1	1-2-2

ball's position			
midfield	4	3	2
wing	4	12	7
corner	6	6	2
totally	14	21	11

In order to reduce human's subjective error, we invited 10 professional basketball coaches to enter the standard zone-defence graphs for our system. For each vertex of any graph, we assign the average of the 10 entered values (with respect to horizontal and vertical coordinates) to it.

3.3. LM-based state matching algorithm

As mentioned above, each zone state has its corresponding zone graph. Therefore, state matching can be transformed into the corresponding graph matching. In this section, we shall extend the Laplacian matrix-based algorithm proposed in [7] for matching zone graphs. The original algorithm proposed in [7] is demonstrated to be precise in matching image pairs; however, on one hand, it is invariant with respect to zoom, and on the other hand, it is very sensitive to the translation of single vertex. The main process of the extended algorithm is expounded as following:

1) Formulating the distance Laplacian Matrices for zone graph G and H:

$$L(G)=[l_{ij}]=\begin{cases}-\dfrac{\left\|V_{G_i}-V_{G_j}\right\|^2}{M^2} & (i\neq j)\\ -\sum\limits_{k\neq i} l_{ik} & (i=j,k,i,j=1,...,5)\end{cases} \tag{1}$$

$$L(H)=[l_{ij}]=\begin{cases}-\dfrac{\left\|V_{H_i}-V_{H_j}\right\|^2}{M^2} & (i\neq j)\\ -\sum\limits_{k\neq i} l_{ik} & (i=j,k,i,j=1,...,5)\end{cases} \tag{2}$$

Here, we take M as the diagonal line length of the half-court playfield.

Obviously, in addition to the spatial distance (SD) relationships as characterized by formula (1) and (2), the spatial direction (SD') relationships between defenders also play an indispensable role. Hence, additional direction Laplacian Matrices with respect to the direction relationships are formulated as:

$$L'(G)=[l'_{ij}]=\begin{cases}-\dfrac{R(V_{G_i},V_{G_j})^2}{\pi^2} & (i\neq j)\\ -\sum\limits_{k\neq i} l'_{ik} & (i=j,k,i,j=1,2,...,5)\end{cases} \tag{3}$$

$$L'(H)=[l'_{ij}]=\begin{cases}-\dfrac{R(V_{H_i},V_{H_j})^2}{\pi^2} & (i\neq j)\\ -\sum\limits_{k\neq i} l'_{ik} & (i=j,k,i,j=1,2,...,5)\end{cases} \tag{4}$$

where $R(V_{G_i},V_{G_j})$ denotes the direction relationship between V_{G_i} and V_{G_j}:

$$R(V_{G_i},V_{G_j})=\underset{[0,\pi]}{\arg\cos}\left(\frac{x_{V_{G_j}}-x_{V_{G_i}}}{\left\|V_{G_j}-V_{G_i}\right\|}\right) \tag{5}$$

N.B.: Single vertex translation has less effect on the direction Laplacian Matrices than the distance Laplacian Matrices.

2) Computing the Singular Value Decomposition (SVD) for each Laplacian Matrix respectively:

$$L(G)=U diag\{\lambda_1,...,\lambda_5\}U^T \tag{6}$$

$$L(H)=V diag\{\gamma_1,...,\gamma_5\}V^T \tag{7}$$

$$L'(G)=U' diag\{\lambda'_1,...,\lambda'_5\}(U')^T \tag{8}$$

$$L'(H)=V' diag\{\gamma'_1,...,\gamma'_5\}(V')^T \tag{9}$$

3) Sign adjusting [7] V and V' into V_a and V_b.

4) Constructing the matching distance between *i*th vertex in G and *j*th vertex in H:

$$P_{ij}=\left\|\lambda_i U^i-\gamma_j V_a^j\right\|^2=(\lambda_i^2+\gamma_j^2)-2\lambda_i\gamma_j U^i(V_a^j)^T \tag{10}$$

$$P'_{ij}=\left\|(U')^i-V_b^j\right\|^2=2[1-2(U')^i(V_b^j)^T] \tag{11}$$

N.B.: Here, Eigen-values are added to take into account of the effects from distance zoom. This is different from the algorithm proposed in [7].

5) Defining the matching relationship matrix:

$$C=UV_a^T=[U^i(V_a^j)^T]=[C_{ij}] \tag{12}$$

$$C'=UV_b^T=[U^i(V_b^j)^T]=[C'_{ij}] \tag{13}$$

6) Computing the matching distance of each vertex in G, with respect to its relationships to the vertices in H : $\forall i,j,k,t\in(1,2,...,5)$:

$$MD_i=\begin{cases}P_{ij}, \ if\ C_{ij}=\max(C_{it})\wedge C_{ij}=\max(C_{kj})\\ \max P_{kt}, \ else\end{cases} \tag{14}$$

$$MD'_i=\begin{cases}P'_{ij}, \ if\ C_{ij}=\max(C_{it})\wedge C_{ij}=\max(C_{kj})\\ \max P'_{kt}, \ else\end{cases} \tag{15}$$

7) Computing the compound matching distance between graph G and H:

$$Dis(G,H)=\sum_{i=1}^{5} MD_i \tag{16}$$

$$Dis'(G,H)=\sum_{i=1}^{5} MD'_i \tag{17}$$

Finally, the global matching distance between graph G and H is defined by:

$$D(G,H)=\mu Dis(G,H)+\mu' Dis'(G,H) \tag{18}$$

where $\mu+\mu'=1$.

As illustrated by the experimental results shown later in this paper, by taking $\mu=0.75$ and $\mu'=0.25$, the algorithm demonstrates an outstanding performance.

3.4. Zone-defence state-sequence matching algorithm

As mentioned earlier in the paper, a test basketball video clip can be denoted by a state-sequence [S_1^{test}, …, S_m^{test}], which in turn can be expressed as a graph-sequence [G_1^{test}, …, G_m^{test}], and the corresponding ball positions [$ball_1^{test}$, …, $ball_m^{test}$]. We shall match each test graph with the graphs in the standard zone-defence graph database.

Zone-defence state-sequence matching algorithm is given as below.

Firstly, we match the test graph-sequence with standard 2-3 zone graph-sequences $G^{23} = [G_1^{23}, ..., G_{n_{23}}^{23}]$ in terms of the following procedure:

Step 1:

For each $G_i^{test} \in G^{test}, i = 1, 2, ..., m$, compute the distances between G_i^{test} and graphs with the same ball position as G_i^{test} in standard 2-3 zone graph database, in terms of the graph matching algorithm presented in section 3.3:

$$D(G_i^{test}, G_{z_j}^{23}) = [D_{ij}^{23}] \tag{19}$$

where $ball_i^{test} = ball_{z_j}^{23}$, $z_j \in \{1, 2, ..., n_{23}\}$, $j = 1, 2, ..., n_p < n_{23}$, and n_p is the number of the graphs with the same ball position as G_i^{test} in 2-3 zone graph database.

Step 2:

Search the most similar graph compared with G_i^{test} in 2-3 zone graph database.

$$SG_i^{23} = G_{z_{j_i}}^{23},\ j_i = \arg\min_j([D_{ij}^{23}]) \tag{20}$$

Step 3:

Computing the similarity degree between the test state-sequence and 2-3 zone state-sequences:

$$Sim_{\text{test}}^{23} = \sum_{i=1}^{m} w_{z_{j_i}}^{23} \Big/ \min([D_{ij}^{23}]) \tag{21}$$

where $w_{z_{j_i}}^{23}$ denotes the weight of graph $G_{z_{j_i}}^{23}$ in 2-3 zone graph database, which are obtained from our coaches as well.

Secondly, in terms of the same procedure, we define the similarity degree between the test state-sequence and 1-3-1 zone state-sequences as:

$$Sim_{\text{test}}^{131} = \sum_{i=1}^{m} w_{z_{j_i}}^{131} \Big/ \min([D_{ij}^{131}]) \tag{22}$$

Thirdly, we define the similarity degree between the test state-sequence and 1-2-2 zone state-sequences in the same manner as:

$$Sim_{\text{test}}^{122} = \sum_{i=1}^{m} w_{z_{j_i}}^{122} \Big/ \min([D_{ij}^{122}]) \tag{23}$$

Finally, the zone-defence formation pattern of the test zone-defence video is defined as:

$$Z^{test} = \arg\max(Sim_{\text{test}}^{23}, Sim_{\text{test}}^{131}, Sim_{\text{test}}^{122}) \tag{24}$$

In summary, the flow chart of basketball zone-defence matching system can be shown as Fig. 3:

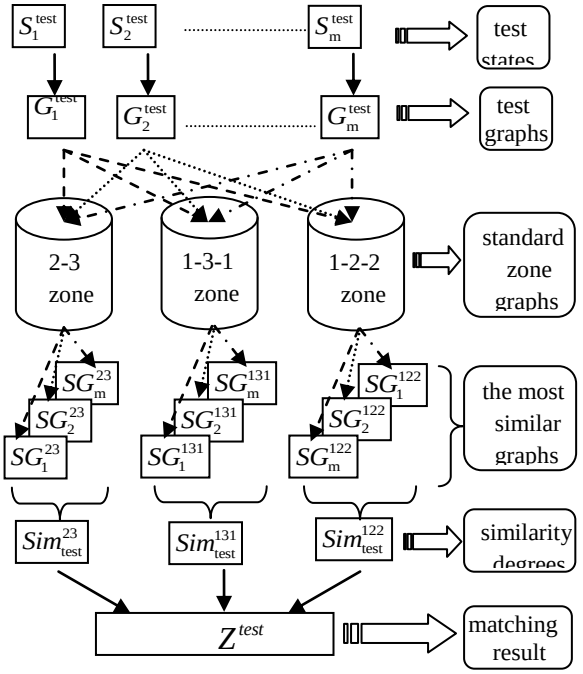


Fig.3. The flow chart of zone-defence matching system

As illustrated in Fig.3, in the first place, the test state-sequence is transformed into the corresponding graph-sequence, which is then matched with the graphs in the standard zone graph database. The compositive similarity degrees with each standard zone are then obtained according to the graph-sequence that is the most similar one compared with the test graph-sequence, which in turn, provide matching results to confirm which zone-defence formation does the test state-sequence belong to.

4. Experimental results

We tested our system with both simulated zone-defence data and real-life basketball zone-defence video data. For each zone-defence formation, with simulated data, we formulated 20 clips (state-sequences) provided by the professional coaches. Also, we have collected the real basketball zone-defence videos lasting about 1 hour, including 112 clips containing 3 to 8 states. The detected zone-defence video clips were manually decomposed into state-sequences and then represented by corresponding graphs. In addition, the normalization of the viewing angle of the camera and object-extracting has not been addressed in this paper. Table 2 shows the numbers of test clips and states in detail:

Table 2. The number structure of test data

	zone	total clips	total states
	2-3	52	286

real	1-3-1	31	221
	1-2-2	29	169
simulated	2-3	20	145
	1-3-1	20	161
	1-2-2	20	128

Firstly, we give an example of the matching (global) distances between a given test state-sequence and 3 standard zones. The second row are the corresponding graphs of the test state-sequence with 3 states as shown in the first row, where the rest rows are the most similar graph compared with each test graph in 2-3, 1-3-1, and 1-2-2 zone-defence strategies, as appearing in the row order. It is clear to see that the most similar zone-defence formation compared with the test state-sequence is the 2-3 zone-defence pattern, which agrees with the matching result from our algorithm.

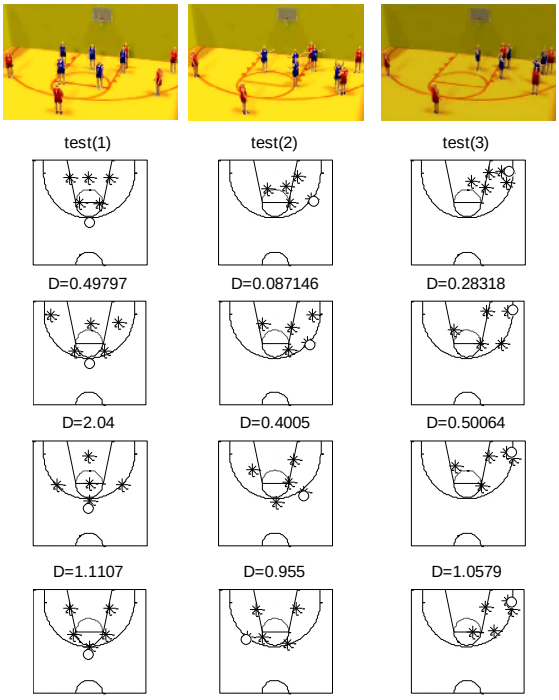


Fig.4. An example of basketball zone-defence video clip recognition

Table 3 below shows the matching precise for each zone-defence pattern. It indicates that the matching algorithm (SDD') proposed here, which takes into account of both spatial distance and spatial direction relationships, outperforms SD or SD' that only address spatial distance or spatial direction relationships, respectively.

Table 3. Matching precise for each zone-defence pattern

test data	zone	precision (%)			average precision
		SD	SD'	SDD'	
real	2-3	74.6	69.8	82.7	75.7
	1-3-1	65.9	63.1	77.4	68.8
	1-2-2	80.3	70.7	86.2	79.0
simulated	2-3	82	80	85	82.3
	1-3-1	91	89	95	91.6
	1-2-2	79	81	85	81.6
average precise:		78.8	75.6	85.2	

5. Conclusions and future work

Based on a formal characterization of time-series and state-sequences, we have introduced the computational technique and algorithm for detecting zone-defence patterns from basketball videos. The experimental results show that it is useful in helping the coach of the defence side to check whether the players play in a right zone-defence strategy, as well as the coach of the offensive side to detect the strategy of the opponent. Specially, we have extended the Laplacian Matrix-based algorithm to take account of the effects from zoom and single defender's translation in zone-defence graph matching. As the future work, we shall take account of the effects from rotation and expand the test dataset to explore the relationships between sequences of basketball zone-defence in order to obtain the best strategy. In addition, we shall test the method in other team-work sport games such as football, volleyball, and so on.

6. References

[1] Allen J. "Towards a General Theory of Action and Time", *Artificial Intelligence*, 23, 1984, pp:123-154.

[2] Allen J. F. and Hayes P. J. "Moments and Points in an Interval-based Temporal-based Logic", *Computational Intelligence*, 5(4), 1989, pp:225-238.

[3] Athitsos V, Papapetrou P, Potamias Ms, Kollios G, & Gunopulos D. "Approximate Embedding-Based Subsequence Matching of Time Series", *In proceedings of the ACM International Conference on Management of Data (SIGMOD)* , Vancouver, BC, Jun 9-12, 2008, pp:365-378.

[4] Chakrabartir S. *Data Mining: Know It All.* Elsevier Science Ltd. 2008.

[5] http://www.coachesclipboard.net.

[6] http://www.guidetocoachingbasketball.com.

[7] Liang D, Tong Q, Wang N, Bao W and Qu L. "A Laplacian Matrix Based Algorithm for Image Matching", Computer Engineering and Applications, 2005, 41(36): pp:31-38.

[8] Li X, Han J. "Mining Approximate Top-K Subspace Anomalies in Multi-dimensional Time-series data", *In Proceedings of the 33rd international conference on Very large data bases*, Vienna, Austria, Sep 23-27, 2007, pp: 447-458.

[9] Ma J, Bie R, Zhao G. "An ontological Characterization of Time-series and State-sequences for Data Mining",

in Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, Jinan Shandong, Oct 18-20, 2008, pp:325-329.

[10] Ma J & Knight B. "A General Temporal Theory", *The Computer Journal*, 37(2), 1994, pp:114-123.

[11] Moon Y, Whang K, & Han W. "General match: a subsequence matching method in time-series databases based on generalized windows", *In Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, Madison, Wisconsin, Jun 3-6, 2002, pp:382-393.

[12] Shanahan M. "A Circumscriptive Calculus of Events", *Artificial Intelligence*, 77, 1995, pp:29-384.

[13] Yankov D, Keogh E, Medina J, Chiu B, & Zordan V. "Detecting Time Series Motifs under Uniform Scaling". *In Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. San Jose, California, Aug 12-15, 2007, pp: 844-853.

Reasoning about Uncertain and Incomplete Temporal Knowledge

Jixin Ma[1] Aihua Zheng[1,2] Brian Knight[1] Miltos Petridis[1] Bin Luo[2]
[1]*The University of Greenwich, Greenwich, London, SE10 9LS, United Kingdom*
[2]*Anhui University, People's Republic of China*
{j.ma; a.zheng, b.knight; m.petridis}@gre.ac.uk luobin@ahu.edu.cn

Abstract

Absolute-time-stamping of temporal data provides an efficient indexing method for temporal information systems, but suffers from the requirement that precise time values for all temporal data need to be available. Temporal knowledge in many Artificial Intelligence systems can be uncertain due to the unavailability of complete and absolute temporal information. This paper introduces an inferential framework for reasoning about uncertain and incomplete temporal knowledge: the uncertainty is formalised in terms of temporal relations jointed by disjunctive connectives, while the incompleteness is due to the lacking of full temporal information. A graphical representation which allows expression of such uncertain and incomplete temporal knowledge is introduced, and based on which, the system can deliver a verdict to the question if a given set of statements is temporally consistent or not, and provide understandable logical inferences by linear programming and contradiction reasoning.

1. Introduction

The representation and manipulation of natural human understanding of temporal phenomena is a fundamental field of study in Computer Science, which aims both to emulate human thinking, and to use the methods of human intelligence to underpin engineering solutions. In particular, many Artificial Intelligence systems need to deal with the representation and reasoning about time in modeling natural phenomena and intelligent human activities. It has been noted that absolute-time-stamping of temporal data provides an efficient indexing method for temporal systems, but suffers from the requirement that precise time values for all temporal data need to be available. Generally speaking, in the domain of Artificial Intelligence, temporal knowledge can be uncertain and incomplete. For instances:

(a) Temporal references may be only relative (e.g., "during the time when the officer was in his office", "after 9 o'clock", etc., which refer to times that are known only by their relative temporal relations to other temporal reference), rather than being absolute (e.g., "8 pm on the 8th of August 2008", "the last week of August 2008", which refer to times with absolute values);

(b) Temporal duration may be only relative (e.g., "less than 6 hours", "more than 12 years but less than 15 years", etc., which refer to some uncertain amount of temporal granularity), rather than being absolute (e.g., "31 minutes", "18 hours", etc., which refer to some certain amount of temporal granularity);

(c) We may only know event E_1 occurred "Before" event E_2 without knowing their precise starting and finishing time, or what happened between E_1 and E_2.

Incomplete relative temporal knowledge such as these is typically derived from humans, where complete and absolute temporal information is rarely available and remembered for knowledge representation and reasoning. Allen's interval-based time theory [1] is a representative example of temporal systems addressing relative temporal relations including "Meets", "Met_by", "Equal", "Before", "After", "Overlaps", "Overlapped_by", "Starts", "Starts_by", "During", "Contains", "Finishes" and "Finished_by". It has been claimed in the literature that time intervals are more suited for expression of common sense temporal knowledge, especially in the domain of linguistics and artificial intelligence. In addition, approaches like that of Allen [1,2] that treat intervals as primitive temporal elements can successfully overcome/bypass puzzles like the *Dividing Instant Problem* [1,4,5,10,11], which is in fact an ancient historical puzzle encountered when attempting to represent what happens at the boundary point that divides two successive intervals. However, as Galton shows in his critical examination of Allen's interval logic [5], a theory of time based only on

This research is supported in part by National Nature Science Foundation of China (No. 60772122)

intervals is not adequate for reasoning correctly about continuous change. In fact, many common sense situations suggest the need for including time points in the temporal ontology as an entity different from intervals. For instance, it is intuitive and convenient to say that instantaneous events such as "The database was updated at 00:00am" [6], "The light was automatically switched on at 8:00pm" [1], and so on, occur at time points rather than intervals (no matter how small they are). Therefore, for general treatments, it is appropriate to include both points and intervals as primitives in the underlying time model, for making temporal reference to instantaneous phenomena with zero duration, and periodic phenomena which last for some positive duration, respectively.

The objective of this paper is to present a framework to assist representing and reasoning about uncertain and incomplete knowledge. In section 2, a time theory based on both points and intervals as the temporal primitive is introduced. Section 3 presents a graphical representation for uncertain and incomplete temporal knowledge. The necessary and sufficient condition for the consistency of a temporal reference is discussed in section 4. Finally, section 5 concludes the paper.

2. The time theory

In this paper, we shall simply adopt the general time theory proposed in [8], which takes a nonempty set, *T*, of primitive time elements, with an *immediate predecessor* relation, Meets, over time elements, and a *duration assignment* function, Dur, from time elements to non-negative real numbers. If Dur(t) = 0, then t is called a point; otherwise, that is Dur(t) >0, t is called an interval. The basic set of axioms concerning the triad (*T*, Meets, Dur) is given as below [8]:

T1. $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_1, t_2) \wedge \text{Meets}(t_1, t_3) \wedge \text{Meets}(t_4, t_2)$
$\Rightarrow \text{Meets}(t_4, t_3))$

That is, if a time element meets two other time elements, then any time element that meets one of these two must also meets the other. This axiom is actually based on the intuition that the "place" where two time elements meet is unique and closely associated with the time elements [3].

T2. $\forall t \exists t_1,t_2(\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$

That is, each time element has at least one immediate predecessor, as well as at least one immediate successor.

T3. $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_1, t_2) \wedge \text{Meets}(t_3, t_4)$
$\Rightarrow \text{Meets}(t_1, t_4)$
$\nabla\ \exists t'(\text{Meets}(t_1, t') \wedge \text{Meets}(t', t_4))$
$\nabla\ \exists t''(\text{Meets}(t_3, t'') \wedge \text{Meets}(t'', t_2)))$

where ∇ stands for "exclusive or". That is, any two meeting places are either identical or there is at least a time element standing between the two meeting places if they are not identical.

T4. $\forall t_1,t_2,t_3,t_4(\text{Meets}(t_3, t_1) \wedge \text{Meets}(t_1, t_4) \wedge \text{Meets}(t_3, t_2) \wedge \text{Meets}(t_2, t_4)) \Rightarrow t_1 = t_2)$

That is, the time element between any two meeting places is unique.

N.B. For any two adjacent time elements, that is time elements t_1 and t_2 such that $\text{Meets}(t_1, t_2)$, we shall use $t_1 \oplus t_2$ to denote their ordered union. The existence of such an ordered union of any two adjacent time elements is guaranteed by axioms T2 and T3, while its uniqueness is guaranteed by axiom T4.

T5. $\forall t_1,t_2(\text{Meets}(t_1, t_2) \Rightarrow \text{Dur}(t_1) > 0 \vee \text{Dur}(t_2) > 0)$

That is, time elements with zero duration cannot meet each other.

T6. $\forall t_1,t_2(\text{Meets}(t_1, t_2) \Rightarrow \text{Dur}(t_1 \oplus t_2) = \text{Dur}(t_1) + \text{Dur}(t_2))$

That is, the "ordered union" operation over time elements is consistent with the conventional "addition" operation over the duration assignment function, i.e., Dur.

Analogous to the 13 relations introduced by Allen for intervals [1,2], there are 30 exclusive temporal relations over time elements including both time points and time intervals, which can be derived from the single Meets order relation and classified into the following 4 groups:

- Relations relating an interval to an interval:
  G0 =
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}
- Relations relating a point to a point:
  G1 =
  {Equal, Before, After}
- Relations relating a point to an interval:
  G2 =
  {Before, After, Meets, Met_by, Starts, During, Finishes}
- Relations relating an interval to a point:
  G3 =
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}

As emphasized in the introduction, in the domain of Artificial Intelligence, temporal knowledge can be uncertain and incomplete. First of all, for a given pair of time elements t_1 and t_2, it may be unknown which of the 30 possible temporal relations as classified in section 2 certainly holds between t_1 and t_2. We shall formalize this uncertain temporal knowledge in term of temporal relations jointed by disjunctive connectives. In this paper, we shall use a triad (T, R, D) to express the temporal reference of a given collection of temporal propositions, where:

- $T = \{t_1, \ldots, t_n\}$ is a finite set of time elements, expressing the knowledge (possibly incomplete) of what time elements are

involved with respect to the given collection of propositions;

- $R = \{R^{(ij)}\} \mid R^{(ij)} = r^{(ij)}_1 \vee \ldots \vee r^{(ij)}_{m(ij)}$, $1 \le i, j \le n$; $i \ne j\}$ is a collection of disjunctions of temporal relations over T, expressing the knowledge (possibly incomplete) as how the time elements in T are related to each other. Here, $r^{(ij)}_k$ is one of the possible temporal relations as classified in section 2.
- D is a collection of duration assignments (possibly incomplete) to time elements in T.

Generally speaking, if t_1 and t_2 are two time elements (specially, time intervals), we know that precisely one of the temporal predicates in G0 must apply for t_1 and t_2. Hence for $r \in G0$:

$$\neg r(t_1, t_2) \Leftrightarrow r_1(t_1, t_2) \vee r_2(t_1, t_2) \vee ... \vee r_{12}(t_1, t_2)$$

where $\{r_1, r_2, ..., r_{12}\} \cup \{r\} = G0$. Hence, to prove $r(t_1, t_2)$, we need to show that $r_i(t_1, t_2)$ is inconsistent for i = 1, …, 12. For instance, we may prove Before(t_1, t_2) by means of showing that, when applying to t_1 and t_2, Equal, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes and Finished_by are all inconsistent with the system. The task of checking the consistency of temporal knowledge shall be deal with later in the paper.

In addition, if it is known that t_1 and t_2 are two points, then G0 can be deduced to G1, and in the case where it is known that t_1 is a point and t_2 is an interval, then G0 can be deduced to G2; similarly, if it is known that t_1 is an interval and t_2 is a point, then G0 can be deduced to G3.

3. Graphical representations

A temporal reference (T, R, D) can be graphically expressed in terms of a directed graph, in which each time element of T is represented as a node, and the collection of disjunctions of temporal relations relating time element t_i and time element t_j is expressed as a directed arc from node t_i to node t_j which is correspondingly labeled with $R^{(ij)}\}$, for some i and j, where $1 \le i, j \le n$; $i \ne j$; the duration assignments of D are denoted as bricked numbers correspondingly attached to the nodes.

For instance, consider temporal reference (T, R, D), where

$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$;

$R = \{Meets(t_1,t_2) \vee Starts(t_1,t_2), Meets(t_1,t_3), Meets(t_2, t_5),$
$Meets(t_2,t_6) \vee Finishes(t_2,t_6),$
$Meets(t_3,t_4) \vee Overlaps(t_3,t_4),$
$Meets(t_4,t_7), Meets(t_5,t_8),$
$Meets(t_6,t_7) \vee Starts(t_6,t_7) \vee During(t_6,t_7),$
$Meets(t_7,t_8) \vee Overlaps(t_7, t_8)\}$

$D = \{Dur(t_2)=1, Dur(t_4)=0.5, Dur(t_6)=0, Dur(t_8)=0.3\}$

The graphical representation of temporal reference (T, R, D) is shown in Figure 1:

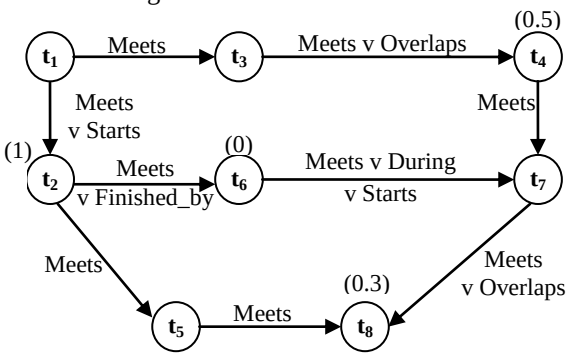


Figure 1. Graph representation of (T, R, D)

For the convenience of expression, in this paper, if $R = \{R^{(ij)}\} \mid R^{(ij)} = r^{(ij)}_1 \vee \ldots \vee r^{(ij)}_{m(ij)}$, $1 \le i, j \le n$; $i \ne j\}$ we shall define:

$|R| = \prod |R^{(ij)}| = \prod_{m(ij)}$, for all i, j appearing in R. That is, $|R^{(ij)}|$ denote the number of temporal relations jointed in $R^{(ij)}$ by disjunctive connectives.

Therefore, the graph of a given temporal reference can be split up into |R| graphs with no disjunctions, each of which expresses a possible case (T, R_k, D) with respect to the temporal reference addressed, k = 1, …, |R|. For example, the graph shown in Figure 1 can be split up into 48 (that is, 2x1x1x2x2x1x1x3x2) no-disjunction graphs.

In general, the temporal order relation between two time elements can be any one of those 30 as classified in section 2. However, as shown in [8], analogous to Allen and Hayes's approach [3], all the temporal can be defined as derived relations in terms of the single "Meets" relation. In fact, such definitions are straightforward. For example, "Before" can be defined as:

$$Before(t_1, t_2) \Leftrightarrow \exists t(Meets(t_1, t) \wedge Meets(t, t_2))$$

Therefore, for any possible case of (T, R, D), that is. (T, R_k, D) (k = 1, …, |R|), we can express R_k as a collection of Meets relations only, denoted as M_k, and obtain the corresponding triad (T, M_k, D) which has no disjunctions involved. Figure 2 below presents the corresponding graph representation of one of such no-disjunction and Meets-only graphs.

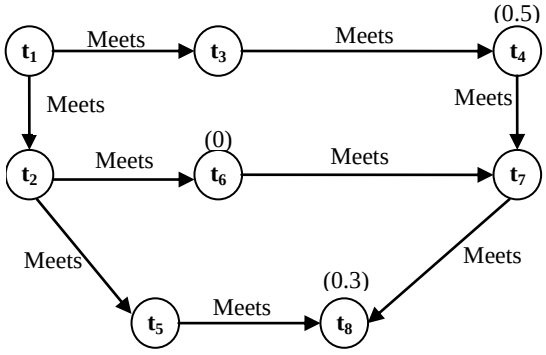


Figure 2. A no-disjunction and Meets-only graph

4. Temporal consistency checking

For a given temporal reference (T, R, D), we say it is temporal consistent if there is at least one of the corresponding no-disjunction and Meets-only cases (T, M, D) is consistent.

4.1 Checking the temporal consistency

In order to develop the consistency checking mechanism for a non-disjunction and Meets-only temporal reference (T, M, D), we shall introduce a graphical representation of (T, M, D) in terms of a directed and partially weighted graph [7] transformed from the corresponding graph of (T, M, D) as show in section 3. In such a graph, time elements are denoted as arcs rather than nodes, and the single Meets relation between time elements t_i and t_j is denoted by the node structure where Meets(t_i, t_j) is represented by t_i being an in-arc and t_j being out-arc to a common node, respectively. For time elements with known duration, the corresponding arcs are weighted by their durations respectively. For example, the transformed graph of Figure 2 is shown as Figure 3:

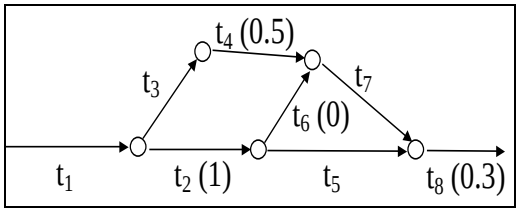


Figure 3. Graph representation of (T, M, D)

The necessary and sufficient condition for the consistency of a general temporal reference, (T, M, D), can be given as below:

1) For each simple circuit in the graph of (T, M, D), the directed sum of weights is zero;
2) For any two adjacent time elements, the directed sum of weights is bigger than zero.

Here, condition 1) guarantees that there exists a valid duration assignment function Dur to the time elements in T agreeing upon D; and condition 2) ensures that no two time points meet each other, that is between any two time points, there is an interval standing between them [8].

The consistency checking for a temporal reference with duration constraints involves searching for simple circuits, and constructing a numerical constraint for each circuit. The existence of a solution(s) to this set of constraints implies the consistency of the system, and each solution gives a possible case for the corresponding temporal scenario that can subsume the addressed temporal reference. Hence, the consistency checker for a random temporal reference is in fact a linear programming problem.

In fact, in the graph presented in Figure 3, there are two simple circuits as shown in Figure 4.

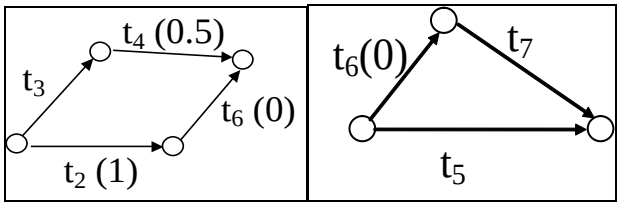


Figure 4. The two simple circuits

Setting the directed sum of weights in each of these two circuits as 0, we get 2 independent constraints:

$Dur(t_2) + Dur(t_6) = Dur(t_3) + Dur(t_4)$

$Dur(t_5) = Dur(t_6) + Dur(t_7)$

We can easily find a solution, for instance: $Dur(t_3) = 0.5$, $Dur(t_5) = Dur(t_7) = 1$. Actually, the duration assignment to t_5 and t_7 can be any positive real number, provided that $Dur(t_5) = Dur(t_7)$.

In some special cases where only relative temporal knowledge are addressed, that is there is no duration constraint involved, temporal reference (T, M, D) is reduced to a pair (T, M) and the consistency checking can be reformulated in a simpler form. In fact, (T, M) is consistent if and only if:

1)' There are no nodes with at least one point in-arc and at least one point out-arc;
2)' The associated reduced graph is acyclic, where the associated reduced graph is formed by means of removing every point arc in the graph of (T, D), and merging any two nodes connected by the point arc.

Here, again, condition 1)' preserves that no two time points meet each other, while condition 2)' preserves that time points are not decomposable, and excludes any circular time structure.

For example, consider a relative temporal reference (T, M), where

T = $\{t_1, t_2, t_3, t_4, t_5, t_6\}$

M = {Meets(t_1, t_2), Meets(t_1, t_3), Meets(t_2, t_6), Meets(t_3, t_4), Meets(t_4, t_5), Meets(t_5, t_6)}

The graphical representation of temporal reference (T, M) is shown in Figure5:

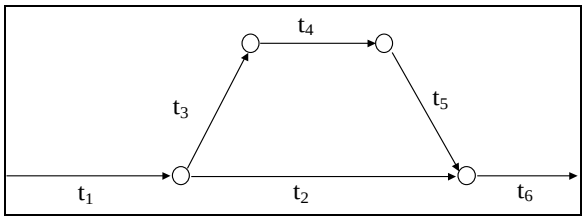


Figure 5. Graph representation of (T, M)

If t_2 is not known to be a time point, then the corresponding graph shown in Figure 5 is acyclic, and hence the temporal reference is consistent.

However, if t_2 is stated to be a point, then the graph in Figure 5 is reduced to the graph as shown in Figure 6.

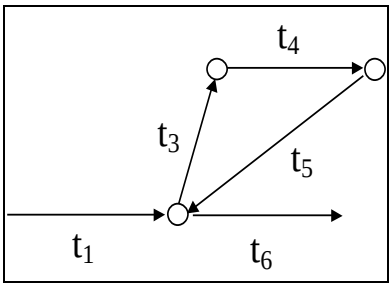


Figure 6. The reduced graph

In the reduced graph in Figure 6, there is a cycle, i.e., t_3 -> t_4 -> t_5 -> t_3. Therefore the temporal reference is inconsistent.

Now, further investigations are needed to deduce logical inferences from both temporal consistent cases and temporal inconsistent cases.

4.2 Deducing Inferences in consistent cases

As mentioned in section 4.1, the consistency checking for a general temporal reference is in fact a linear programming problem, where each solution to the linear programming problem gives a possible case for the corresponding temporal scenario that can subsume the addressed temporal reference. In the case where the temporal reference is consistent, there exists at least one solution to the linear programming problem. Of course, if the solution(s) is unique, we can use this solution construct the corresponding complete temporal reference which is also unique.

However, in general cases where a verdict that the temporal reference is consistent has been reached, there may be more than one, or even an infinite number of solutions to the corresponding linear programming. This may be due to various forms of incompleteness of the corresponding temporal reference, e.g., some referencing time elements may be missing, the duration of some time elements may be unknown, and so on. Therefore, we can only construct the possible complete scenarios which can subsume the addressed temporal reference.

In this case, we can at least find the minimal model(s) among these complete scenarios by means of defining and calculating the similarity degree between the complete temporal references and the original partial temporal reference.

Since each temporal reference can be expressed as a directed and partially weighted graph, the problem of matching temporal references can be transformed into conventional graph matching.

4.3 Deducing inferences in inconsistent cases

In the case where a verdict that the temporal reference is inconsistent has been reached, we can simply analyse and identify the linear equations which make the corresponding linear programming unsolvable, which, in turn, will identify which part(s) of the temporal reference actually leads to the inconsistency.

4.4 An illustrating example

In As an example, consider the situation where two persons, Peter and Jack, are suspected of committing a murder during the daytime. In court, Jack and Peter gave the following statements, respectively:

- Peter's statements:

I got home with Jack before 1pm. We had our lunch, and when Jack left I put on a video. The video lasts 2 hours. Before it finished, Robert arrived. When the video finished we went to the train station and waited until Jack came at 4 pm.

- Jack's statements:

Peter and me went to his home and arrived there before 1pm. When we finished our lunch there, Peter put on a video, and I left and went to the supermarket. I stayed there for between 1 and 2 hours. Then I drove to my home to collect some mail. It takes between 1.5 to 2 hours to reach my home, and about the same to the train station. I arrived at the train station at 4 pm.

- In addition, being a witness, Robert made these statements:

I left home at 2 pm and went to Peter's house. He was playing a video, and we waited till it finished. Then we went together to the train station and waited for Jack. Jack got to the train station at 4pm.

The temporal reference of the above temporal information involves the following time elements:

- i_1: the time (interval) over which Peter and Jack went to Peter's home;
- 1pm: the time (point) before which they arrived at Peter's home;
- i_2: the time (interval) over which Peter and Jack had lunch;
- i_3: the time (interval) over which Peter played the video ($Dur(i_3) = 2$);
- i_4: the time (interval) over which Jack went to the supermarket;

- p_1: the time (point) when Robert arrived at Peter's house;
- i_5: the time (interval) over which Peter and Robert went to the train station;
- i_6: the time (interval) over which Peter and Robert waited for Jack at the train station;
- 4pm: the time (point) when Jack arrived at the train station;
- i_7: the time (interval) over which Jack stayed in the supermarket ($1<Dur(i_7)< 2$);
- i_8: the time (interval) over which Jack drove to his home ($1.5<Dur(i_8)< 2$);
- i_9: the time (interval) over which Jack collected some post from his home;
- i_{10}: the time (interval) over which Jack drove to the train station ($1.5<Dur(i_{10})<2$);
- 2pm:the time (point) when Robert left home;
- i_{11}: the time (interval) over which Robert went to Peter's house;
- i_{12}: the time (interval) over which Peter and Robert watched the video together;
- i_{13}, i_{14}, ..., i_{27}: some extra relative time elements which are used for expressing the correspondingly relative duration knowledge, e.g., with i_{19}, i_{20}, i_{21}, i_{22}, and $Dur(i_{19}) = 1.5$ and $Dur(i_{21}) = 2$, we can express that $1.5 < Dur(i_8) < 2$ (Picture 3)

The graphical representation of the corresponding temporal reference for the above legal statements can be shown as Figure 7 as below:

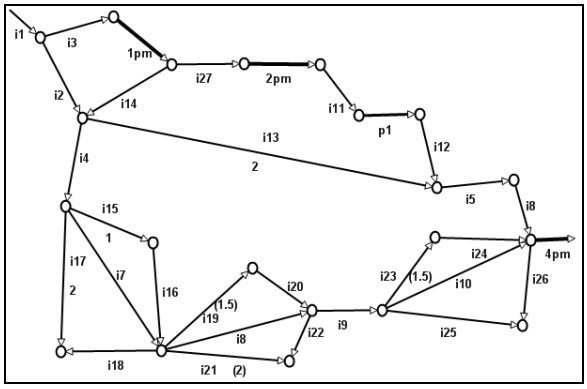


Figure 7. (T, M, D) of the legal statements

From Figure 7, we see that there are three time elements (i.e., two intervals, i_{11} and i_{12}, and one points, p_1) standing between 2pm and 4pm. Since each interval has a positive duration and each point has a non-negative duration, we can infer that:

$$Dur(i_5) + Dur(i_6) < 2$$

In addition, since $Dur(i_3) = 2$, hence

$$Dur(i_3) + Dur(i_5) + Dur(i_6) < 2 + 2 = 4$$

However,

$$Dur(i_4) + Dur(i_7) + Dur(i_8) + Dur(i_9) + Dur(i_{10}) > 0 + 1 + 1.5 + 0 + 1.5 = 4$$

Therefore, for the simple circuit, i.e., i_3, i_5, i_6, i_{10}, i_9, i_8, i_7, i_4, as shown below in Figure 8, there does not exist any duration assignment over T such that

$$Dur(i_3) + Dur(i_5) + Dur(i_6) = Dur(i_4) + Dur(i_7) + Dur(i_8) + Dur(i_9) + Dur(i_{10})$$

In other words, there is no solution to the following linear equation:

$$Dur(i_3) + Dur(i_5) + Dur(i_6) - Dur(i_4) - Dur(i_7) - Dur(i_8) - Dur(i_9) - Dur(i_{10}) = 0$$

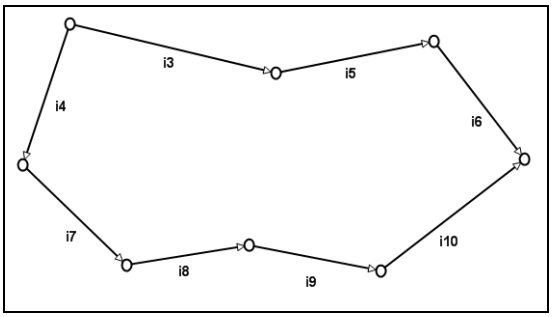


Figure 8. A simple circuit in the legal statements

Hence, the temporal reference shown in Figure 7 is inconsistent, and therefore we can directly confirm that some statements are untrue.

Suppose the video can be checked that it did actually last for two hours, we can confirm that there must be some falsity in either Robert's or Jack's statements. If it can be proved that Robert did leave home at 2 pm, then Jack must have lied in making his statements. Otherwise, to convince the jury that his statements are true, Jack must prove that Robert left home at some time before 2 o'clock in the afternoon.

5. Conclusions

In this paper, we introduced an inferential framework for temporal representation and temporal reasoning. It allows expression of both absolute and relative temporal knowledge, and provides graphical representation of temporal references in terms of directed and partially weighted/labelled graphs. Based on the temporal reference of a given scenario with partial temporal information, the framework can check if it is temporally consistent or inconsistent, and derive the corresponding logical inferences. The benefit of this approach is that the inferential framework has powerful analytic abilities, and its analysis is amenable to human scrutiny.

6. References

[1] Allen, J.: Maintaining knowledge about temporal intervals, Communications of the ACM, 26 (11), 832-843, (1983).
[2] Allen, J.: Towards a General Theory of Action and Time, Artificial Intelligence, 23, 123-154 (1984).
[3] Allen, J., Hayes, P.: Moments and Points in an Interval-based Temporal-based Logic, Computational Intelligence, 5, 225-238 (1989).
[4] van Benthem, J.: The Logic of Time, Kluwer Academic, Dordrech (1983).
[5] Galton, A.: Critical Examination of Allen's Theory of Action and Time, Artificial Intelligence, 42, 159-188 (1990).
[6] Jensen, J., Clifford, J., Gadia, S., Segev, A., Snodgrass, R.: A Glossary of Temporal Database Concepts, SIGMOD RECORD, 21(3), 35-43 (1992).
[7] Knight, B., Ma, J.: A General Temporal Model Supporting Duration Reasoning, Artificial Intelligence Communication, 5(2), 75-84 (1992).
[8] Ma, J., Knight, B.: A General Temporal Theory, the Computer Journal, 37(2), 114-123 (1994).
[9] Ma, J., Knight, B.: A Reified Temporal Logic, the Computer Journal, 39(9), 800-807 (1996).
[10] Ma, J., Knight, B.: Representing The Dividing Instant, the Computer Journal, 46(2), 213-222 (2003).
[11] Vila, L.: A Survey on Temporal Reasoning in Artificial Intelligence, AI Communication**,** 7(1), 4-28 (1994).

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES[9]

AIHUA ZHENG[1,2]
JIXIN MA[1]
MILTOS PETRIDIS[1]
JIN TANG[2]
BIN LUO[2]

[1]*School of Computing and Mathematical Sciences,*
the University of Greenwich, Old Royal Naval College, Park Row,
London, SE10 9TZ, United Kingdom

[2]*The Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education,*
Anhui University, No.3 Feixi Road,
Hefei, People's Republic of China

{a.zheng, j.ma, m.petridis}@gre.ac.uk, ahhftang@gmail.com, luobin@ahu.edu.cn

This paper proposes a framework for structure-based feature extraction in basketball zone-defence strategies. Firstly, a graphical representation for key-frames extracted from zone-defence video clips is introduced, where each key-frame is expressed in terms of a zone-defence graph, representing the positions of defenders and the ball. Secondly, defence-lines are defined and extracted from zone-defence graphs for each zone-defence strategy, based on which, a 10-dimentional feature vector with respect to the defence-lines is introduced to characterize the structure relationships. Experiments have been conducted for basketball zone-defence strategy detection on both simulated and real-life basketball zone-defence video database, which demonstrate the validation and practicability of such a structure based feature characterization, and, in particular, its robustness with respect to the disturbance of local transformation of subprime nodes in the graphs.

Keywords: Feature extraction; Graphical representation; Structure relationship; Video clip detection; Basketball zone-defence.

1. Introduction

Video detection is one of the hottest research topics in Content-based Video Retrieval and attracted more and more attentions. [Qi *et al*. 2007] proposed optimized multi-graph-based semi-supervised learning (OMG-SSL) algorithm in a regularization and optimization framework. A temporal reasoning method was proposed for events annotation in news video in [Marco *et al*. 2008]. As a popular worldwide media, sport video has become an increasingly important and active research area in video/image processing and pattern recognition including feature extraction, shot segmentation, event or highlight detection, and semantic annotation and so on. [Gong *et al*. 1995] presented an automatic system for parsing TV soccer program by domain knowledge, feature analysis and model matching techniques. [Babaguchi *et al*. 1999] Proposed an event based video indexing for football games achieved by the idea of intermodal collaboration which takes into account of the semantic dependency

[9] This research is supported in part by National Nature Science Foundation of China (No. 60772122).

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

between multimodal information streams including visual, auditory and text. [Chang *et al*. 1996] extracted the information in soccer video by an integrate speech understanding and image analysis algorithms. [Rui *et al*. 2000] presented a highlights extraction approach for baseball games on set-top devices in noisy environment. [Xu *et al*. 2001] proposed a grass-area-ratio based algorithm for soccer video segmentation. [Pan *et al*. 2001] proposed an automatic event detection and sports program summarization method based on detecting slow motion replay segments. [Efros *et al*. 2003] proposed a new motion descriptor to recognize human actions at a distance in soccer based on smoothed and aggregated optical flow measurements over a spatio-temporal volume centred on a moving figure. [Luo *et al*. 2003] presented object-based analysis and interpretation for baseball video based on automatic video object extraction, video object abstraction, and semantic event modelling. [Urtasun *et al*. 2005] presented a novel motion tracking approach in golf. [Bagdanov *et al*. 2007] proposed the multimedia ontology for soccer video detection.

A number of approaches have been proposed for basketball video analysis, including shot classification, scene recognition and event detection. [Tan *et al*. 2000] presented a camera motion based annotation and classification tool using the low-level information available directly from MPEG compressed basketball videos. [Nepal *et al*. 2001] proposed a goal detecting method in basketball videos by combining feature extraction techniques with domain specific knowledge. [Zhou *et al*. 2000] proposed a supervised rule based basketball video classification system after investigating the use of video content analysis and feature extraction and clustering. [Kim *et al*. 2002] proposed a semantic information extracting mechanism for basketball video sequence using audio and video features. [Xu *et al*. 2004] proposed an audio keywords generating approach for basketball video based on low-level audio features and applied audio keywords together with heuristic rules to event detection. [Kim *et al*. 2005] presented a summarization method for basketball videos. [Perse *et al*. 2009] proposed trajectory-based approach to the automatic recognition of complex multi-player behavior in a basketball game.

However, few of them focused on zone-defence detection, which is essential and crucial in basketball games. On one hand, the defensive coach needs to layout the zone-defence strategy and check whether the team is playing in the right strategy or not all the time; on the other hand, the offensive coach also needs to know which zone-defence strategy the defenders are adopting.

Zone-defence is a common strategy adopted in basketball games. It is different from man-to-man defence in that, instead of guarding a particular player, each zone defender is responsible for guarding an area on the court (or "zone") and any offensive player that comes into that area. Zone defenders move their position on the court according to where the ball moves. Zone-defence can disrupt the opponent's offensive plan by means of protecting the paint area and forcing the opponent to shoot from outside. In addition, changing defences from man-to-man to various can make the offense off-balance and confused.

In particular, feature extraction is one of the most significant tasks plays a basic and essential role in Zone-defence detection. The original approach is the common features such as color, texture and shape. It's noted that they are not competent due to the distinct structure character in zone-defence strategies. Graphic representation has been investigated for zone-defence detection. Graph matching (GM) algorithms and their improved variants have been well applied to match graph patterns [Zheng *et al*. 2009 and Ma *et al*. 2007]. However, the efficiency and accuracy of most graph matching algorithms depend very much on the tested graphs constructed according to the expectation or artificial criteria, rather than real-life applications [Zheng *et al*. 2009], which in turn means most graph matching algorithms are sensitive to the outliers or local bias such as the translation of subprime notes

in the graph. [Chin *et al*. 2005] proposed a Spatial-Relationship (SR) based approach to describe the position relationship between defenders. However, it relies on the accuracy of identification of each defender, which is hardly achievable.

Generally speaking, the defence-lines and the structure relationship between defence-lines play a crucial role in team sports, such as basketball, football, volleyball and so on. Therefore, analysing the structure relationship between defence-lines plays an important role in basketball zone-defence strategy detection. Therefore, in this paper, a structure-based feature descriptor in terms of a 10-dimentional feature vector is proposed for zone-defence strategy. The basic idea is to describe the distinct structure relationship between defence-lines based on the graphical representation of key-frames.

In what follows in this paper, the graphical representation of key-frames in basketball zone-defence videos is introduced in section 2. Section 3 elaborates the structure-based feature extraction in basketball zone-defence graphs and the corresponding algorithms. Based on the extracted structure features, section 4 designs the actual algorithm for the overall basketball zone-defence detection system. Experimental results are provided, analyzed and evaluated in section 5, demonstrating the good performance of proposed feature descriptor. Finally, section 6 provides a brief summary and concludes the paper.

2. Graphical Representation in Basketball Zone-defence Video

Videos can be organized at different levels for various research purposes. In this paper, basketball videos are organised in terms of clips. Each clip represents a certain round of offense (or defence) and is denoted as a list of images, or the so-called key-frames sequence: $I = [I_1, \ldots, I_n]$, which consists of the key-frames extracted one per 2 seconds from the clip. We premise that:

(1) The defenders have adjusted to their best defensive positions at the moment when the ball is just to be passed or dribbled;

(2) As the zone-defence strategy is to defence the offensive opponent to attack into interior playfield, we only consider the key-frames when the ball is in the midfield, the wing and the corner as key-frames.

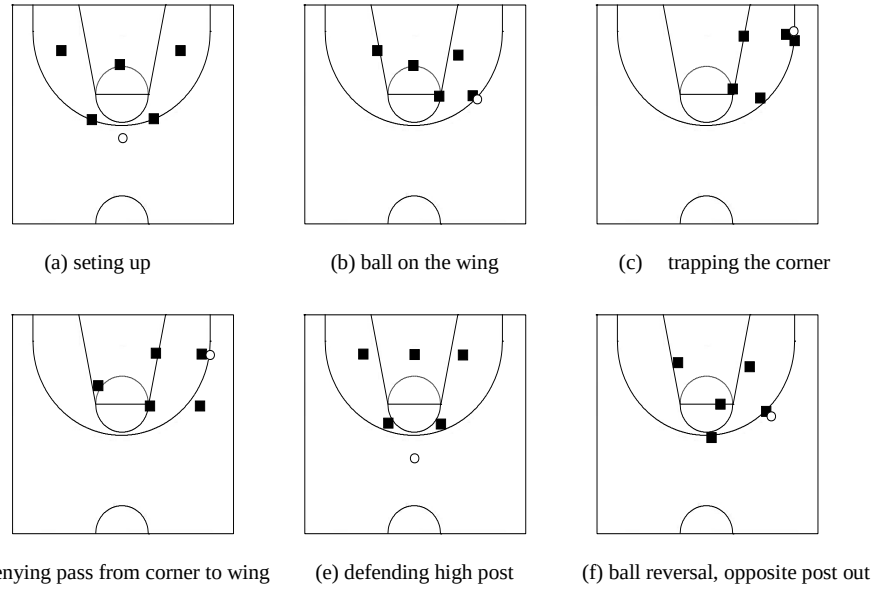
The metric position detection of defenders and the ball is implemented similarly as in [Assfalg *et al*. 2003]: The ball's position, which is either in the midfield, in the wing, or in the corner, is obtained from its motion described in terms of camera motion, which in turn, is captured by image motion estimation algorithm [Baldi *et al*. 1999]. As for defenders position, in the first place, the defend side and offensive side are distinguished by the colour difference of sportswear; template matching and projective transformation are then implemented to determine the metric position of defenders [Assfalg *et al*. 2003].

Each key-frame I_i ($i = 1, \ldots, n$) can be described by its corresponding six-note graph G_i structured by the 5 defenders' position (horizontal and vertical coordinates) plus the ball's position. Following the conventional notations in graph theory, we represent a zone-defence graph as $G = <V, E>$, where V and E denote the set of the notes (defenders' position) and the set of edges respectively, and $E \subseteq V \times V$. In particular, here, $|V| = 6$. Assuming $V = \{V_b, V_1, V_2, V_3, V_4, V_5\}$ has been ascending ordered by the Euclidean distance to the ball (V_b).

Zone-defence can be divided into various strategies, including 2-3, 1-3-1, 1-2-2, 3-2, 2-2-1, 2-1-2 and 1-2-1-1 zone-defence strategies, where the first three strategies, which have been noted as the most typical ones employed in actual basketball games, are focused in our paper.

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

A standard zone-defence graph database of these 3 typical zone-defence strategies (2-3, 1-3-1 and 1-2-2 zone-defence) is constructed and populated with graph data corresponding to some of the pictures illustrated on two basketball coaching web sides. For instance, a typical round of 2-3 zone-defence can be expressed as Fig. 1 where 5 squares and the circle denote the 5 defenders and the ball respectively.



(a) seting up (b) ball on the wing (c) trapping the corner

(d) denying pass from corner to wing (e) defending high post (f) ball reversal, opposite post out

Fig.1 A typical round of 2-3 zone-defence strategy

Table 1. The number of standard zone-defence graphs

Zone-defence / Ball's position	2-3	1-3-1	1-2-2
Midfield	4	3	2
Wing	4	12	7
Corner	6	6	2
Totally	14	21	11

Table 1 below shows the detailed number of zone-defence graphs we have currently collected as standard zone-defence graphs for each strategy in different ball's position. Analogously, only the three typical zone-defence strategies and only the key-frames when the ball is in the midfield, the wing and the corner are considered.

Fig.2 shows the flow chart of basketball zone-defence detection system. For each test zone-defence video clip, it is decomposed into a sequence of key-frames. Each key-frame is represented by a zone-defence graph as mentioned above and matched with the graphs in the standard zone graph database. The global distance with each standard zone are then obtained according to the graph-sequence that is the most similar one (has the smallest distance) to the test graph-sequence, which in turn, provide matching results to confirm which zone-defence strategy does the test key-frame sequence belong to.

It is worth pointing out that, in the framework presented in this paper, zone-defence key-frames are transferred into zone-defence graphs by means of graphical representation. However, instead of using conventional graph matching algorithms, a structure-based

feature extraction algorithm, which will be discussed in detail in next section, is proposed to measure the similarity between zone-defence graphs.

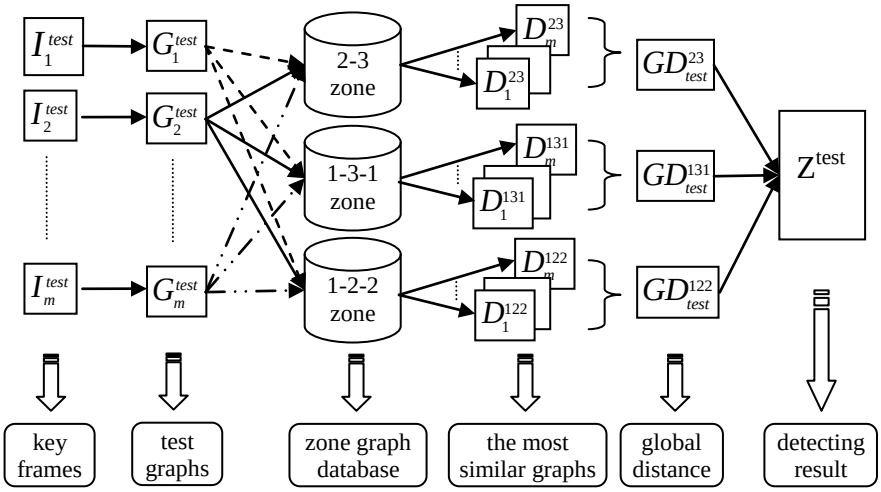


Fig.2 The flow chart of basketball zone-defence detection system

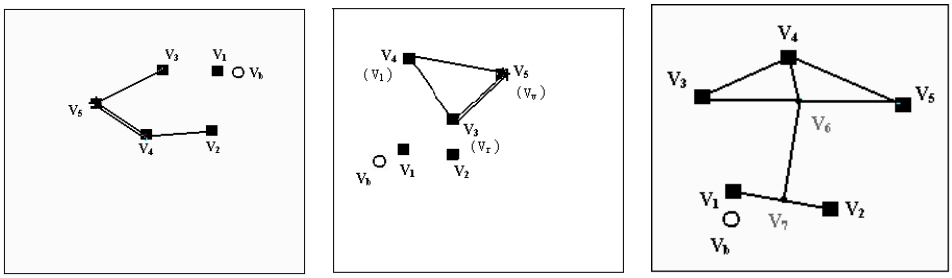
3. Structure-based Feature Extraction in Basketball Zone-defence Strategies

Different zone-defence strategy has different number and type of defence-lines in basketball, For instance, there are two defence-lines in 2-3 zone-defence strategy. Generally, we define that the 2 defenders in the front line construct the first defence-line and the rest 3 defenders construct the second defence-line. In addition, different zone-defence strategy, as what it's named, has its own typical defence-line. For instance, the typical defence-line of 2-3 zone defence strategy is the second defence-line. We shall define the structure-based features to describe the structure relationship between defence-lines. The angle formed by the typical defence-line in each zone-defence strategy is named corresponding character-angle, the definition of which is crucial to the extraction of the other structure features.

3.1. Structure-based Features in 2-3 Zone-defence Strategy

In standard 2-3 zone-defence strategy, normally, we define that the 2 defenders closest to the ball construct the first defence-line; and the rest 3 defenders construct the second defence-line which is defined as the 2-3 character line. The angle formed from the 2-3 character line is defined as "2-3 character-angle" and denoted by shorthand writing as CA_{23}. There are two folds regarding the definition of CA_{23}:

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES



(a) a counterexample (b) another counterexample (c) a general example

Fig.3 Zone graph examples in 2-3 zone-defence

(1) Which 3 notes construct CA_{23}?

Normally, CA_{23} is composed of the 3 defenders farthest from the ball. However, in some zone graphs, CA_{23} may not exactly be constructed by the 3 defenders farthest from the ball by common sense from human understanding of zone-defence strategies. For instance, in Fig.3 (a), assume that $V = \{V_b, V_1, V_2, V_3, V_4, V_5\}$ has been ascending ordered by the distance to the ball (V_b) and V_3 and V_2 have an approximately same distance to the ball. Obviously, the CA_{23} should be constructed by V_2, V_4 and V_5, which is more reasonable according to common sense than that constructed by the farthest 3 notes (V_3, V_4 and V_5).

In other word, if the difference between the distances from the third and forth farthest notes to the ball is smaller than a given threshold, then the one forming a larger angle with the segment constructed by the farthest two notes will be taken to form the character line. The algorithm is described as following:

If $(\left|\overline{V_2V_b} - \overline{V_3V_b}\right| < \delta) \,\&\, (\angle(V_2, \overline{V_4V_5}) > \angle(V_3, \overline{V_4V_5}))$

$CN_{23} = \{V_2, V_4, V_5\}$

Else

$CN_{23} = \{V_3, V_4, V_5\}$

End.

---

where δ=0.05 (the distance of diagonal of half-court is normalized to 1), CN_{23} denotes the set of notes constructing CA_{23} and $\angle(X, \overline{YZ})$ represents the angle between note X and segment YZ which is defined as:

$$\angle(X, \overline{YZ}) = \begin{cases} \angle XYZ, & |XY| > |XZ| \\ \angle XZY, & \text{else} \end{cases} \tag{1}$$

(2) Which one is the vertex of CA_{23}?

For the reason of simple description, without losing the generality, we assume CN_{23} = $\{V_3, V_4, V_5\}$, as shown in Fig.3(b) and arrange $\{V_3, V_4, V_5\}$ into $\{V_l, V_v, V_r\}$ in clockwise order with respect to the ball, where l, v, r $\in\{3, 4, 5\}$. In general, node V_v is then taken as the vertex of CA_{23} while V_l, V_r are the end-points of CA_{23}. However, if Angle<V_v, V_b, V_l> (or Angle<V_v, V_b, V_r>) is smaller than a given threshold, and $|V_lV_b|<|V_vV_b|$ (or $|V_rV_b|<|V_vV_b|$) then V_l (or V_r) will be re-taken as the vertex of CA_{23}. For instance, in Fig.2, CN_{23} = $\{V_3, V_4, V_5\}$. Assume that V_4, V_5 and V_3 are in the clockwise order with respect to the ball. V_3 should be defined to be the vertex of CA_{23}, which is more reasonable than regarding V_5 as the vertex of CA_{23}. The algorithm is described as following:

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

If $(\angle V_l V_b V_v < \theta) \& (|V_l V_b| < |V_v V_b|)$

$CA_{23} = \angle V_v V_l V_r$

Else

If $(\angle V_r V_b V_v < \theta) \& (|V_r V_b| < |V_v V_b|)$

$CA_{23} = \angle V_v V_r V_l$

Else

$CA_{23} = \angle V_l V_v V_r$

End

End

where $\theta = \pi/12$ and we appoint CA_{23} as the obtuse angle if its vertex is biased towards the ball compared with its two end points.

The first 4 structure features with respect to CA_{23} are correspondingly defined as below (As for a general example illustrated in Fig.3(c), $\overline{V_1V_2}$ is the first defence-line and V_3, V_4, V_5 construct the second defence-line, and V_6, V_7 are the midpoints of $\overline{V_3V_5}$, $\overline{V_1V_2}$ respectively):

I. $CA_{23} = \angle V_3 V_4 V_5$: Character-Angle of 2-3 zone-defence.

As explained earlier, this angle characterises the defenders' positions on the character line of 2-3 zone-defence.

II. $FSA_{23} = \angle(\overline{V_7V_6}, \overline{V_3V_5})$: Angle formed by the first and the second defence-lines.

where $\angle(\overline{XY}, \overline{ZW})$ denotes the angle formed by segment $\overline{XY}$ and segment $\overline{ZW}$ that is no bigger than $\pi/2$. It characterises the structure relationship between the first and the second defence-lines.

III. $BCA_{23} = \angle(\overline{V_4V_6}, \overline{V_3V_5})$: the bias of the CA_{23}.

which is an angle presents the bias of the vertex on second defence-lines of 2-3 zone-defence.

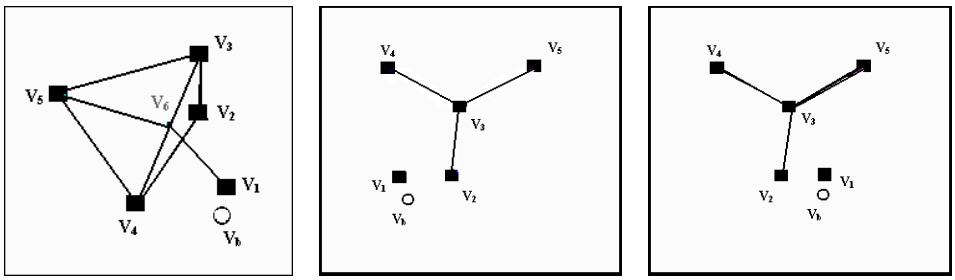
IV. $RFSA_{23} = (|V_1V_2| / |V_3V_5|) \angle (\overline{V_1V_2}, \overline{V_3V_5})$: restricted FSA_{23}.

which denotes the restricted angle of the first and the second defence-lines of 2-3 zone-defence. The shorter of $\overline{V_1V_2}$ comparing with $\overline{V_3V_5}$, the angle of segment $\overline{V_1V_2}$ and segment $\overline{V_3V_5}$ has less effect to zone graphs. So it's reasonable to take into account a coefficient to the angle.

3.2. Structure-based Features in 1-3-1 Zone-defence Strategy

In 1-3-1 zone-defence, the defender nearest to the ball constructs the first defence-line. The second defence-line is constructed by 3 defenders, presenting the basic character of 1-3-1 zone-defence, which is defined as the 1-3-1 character line. The angle formed from the 1-3-1 character line is defined as "1-3-1 character-angle" and denoted as CA_{131}. The key point here is to define the vertex and two end points of CA_{131}.

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES



(a) $CA_{23} < \pi$ (b) $CA_{23} > \pi$ case 1 (c) $CA_{23} > \pi$ case 2

Fig.4 Zone graph examples in 1-3-1 zone-defence

Based on CA_{23} as what we have extracted, there are two cases to define CA_{131}: (Here, we also use V_1, V_2, V_3, V_4 and V_5 to denote the 5 defenders, and assume V_1 is the nearest defender to the ball, $CA_{23} = \angle V_3V_5V_4$ in Fig.4(a) and $CA_{23} = \angle V_4V_3V_5$ in Fig.4(b) and (c)). If the corresponding CA_{23} is smaller than π (as shown in Fig.4(a)), then CA_{131} has the same two end-points (V_3 and V_5) as that of CA_{23}, and the vertex of CA_{131} is the node (V_2) from the rest 3 which is neither the closest to the ball nor the vertex of CA_{23}; otherwise (as shown in Fig.4(b) and (c)), CA_{131} will have the same vertex as that of CA_{23}, and the node which is neither on the 2-3 character line and nor the closest to the ball will be taken as one of the two end-points of CA_{131} where the other end-point is one of the two end-points of CA_{23} which will ensure that CA_{131} divides the rest two nodes sit on each side of the 1-3-1 character line, respectively.

The detection algorithm is expounded below:

If $CA_{23} = \angle V_3V_5V_4 < \pi$

$CA_{131} = \angle V_3V_2V_4$

Else $CA_{23} = \angle V_4V_3V_5 \geq \pi$

case 1: $V_1 \in area(V_2V_3V_4)$

$CA_{131} = \angle V_2V_3V_4$

case 2: $V_1 \in area(V_2V_3V_5)$

$CA_{131} = \angle V_2V_3V_1$

End

End

Where, $area(V_2V_3V_4)$, $area(V_2V_3V_5)$ and $area(V_4V_3V_5)$ denote 3 plane areas divided by the beam $\overrightarrow{V_3V_2}$, $\overrightarrow{V_3V_4}$ and $\overrightarrow{V_3V_5}$. Obviously, V_1 cannot belong to $area(V_4V_3V_5)$.

The next 3 features with respect to CA_{131} are defined below (As for a general example illustrated in Fig.4(a) and assume V_6 is the midpoint of segment $\overline{V_3V_4}$):

V. $CA_{131} = \angle V_3V_2V_4$: Character-Angle of 1-3-1 zone-defence.

which characterises the defenders' positions on the character line of 1-3-1 zone-defence analogously.

VI. $FSA_{131} = \angle(\overline{V_1V_6}, \overline{V_3V_4})$: Angle formed by the first and the second defence-lines.

which characterises the structure relationship between the first and the second

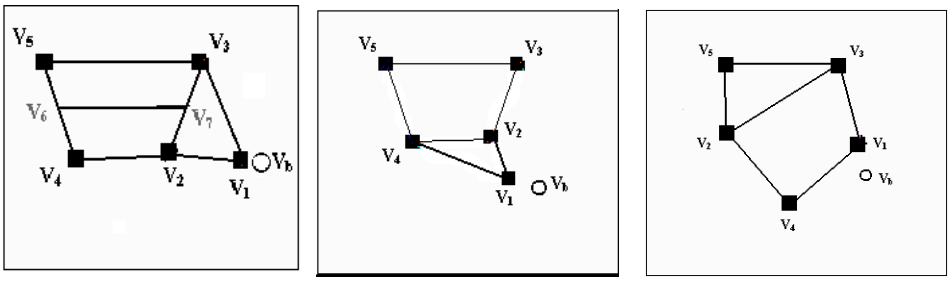
defence-lines of 1-3-1 zone-defence.

VII. $STA_{131} = \angle(\overline{V_5V_6}, \overline{V_3V_4})$: Angle formed by the second and the third defence-lines.

which characterises the structure relationship between the second and the third defence-lines of 1-3-1 zone-defence.

3.3. Structure-based Features in 1-2-2 Zone-defence Strategy

In 1-2-2 zone-defence, the defender closest to the ball forms the first defence-line. As the examples shown in Fig.5, assume that V_1 is the closest defender; $\angle V_4V_2V_3$ is the CA_{131}. If $\angle V_4V_2V_3$ is equal or larger than π (Fig.5(a) and (b)), the vertex of CA_{131} and the nearer one to the first defence-line of the two end-points of CA_{131} construct the second defence-line; the rest two defenders construct the third defence-line. Otherwise (Fig.5(c)), the two end-points of CA_{131} construct the second defence-line and the rest two defenders construct the third defence-line. The first and the second defence-lines present the basic character of 1-2-2 zone-defence, which define the 1-2-2 character line. The angle formed from the 1-2-2 character line is defined as "1-2-2 character-angle" and denoted as CA_{122}.



(a) $CA_{131} \geq \pi$ case 1 (b) $CA_{131} \geq \pi$ case 2 (c) $CA_{131} < \pi$

Fig.5 Zone graph examples in 1-2-2 zone-defence

The algorithm is described as following (CA_{122}, SDL_{122} and TDL_{122} denote the Character Angle, the second defence-line and the third defence-line of 1-2-2 zone-defence, respectively):

If $\angle V_4V_2V_3 \geq \pi$

case 1: $|V_1V_3| < |V_1V_4|$

$CA_{122} = \angle V_2V_1V_3$, $SDL_{122} = \overline{V_2V_3}$, $TDL_{122} = \overline{V_4V_5}$

case 2: $|V_1V_3| \geq |V_1V_4|$

$CA_{122} = \angle V_2V_1V_4$, $SDL_{122} = \overline{V_2V_4}$, $TDL_{122} = \overline{V_3V_5}$

End

Else

$CA_{122} = \angle V_3V_1V_4$, $SDL_{122} = \overline{V_3V_4}$, $TDL_{122} = \overline{V_2V_5}$

End

The last 3 features with respect to CA_{122} are defined as below (assume that $CA_{122} = \angle V_2V_1V_3$, $SDL_{122} = \overline{V_2V_3}$ and $TDL_{122} = \overline{V_4V_5}$, V_6 and V_7 are the midpoints of segment $\overline{V_4V_5}$ and segment $\overline{V_2V_3}$ respectively as shown in Fig. 5(a)):

VIII. $RCA_{122} = (\min(|V_1V_2|, |V_1V_3|)/\max(|V_1V_2|, |V_1V_3|))\angle V_2V_1V_3$.

Here, we add a coefficient to take into account the effect from the movement of node V_1 along the circle formed from V_1, V_2 and V_3.

IX. $RSTA_{122} = (|V_2V_3|/|V_4V_5|)\angle(\overline{V_2V_3}, \overline{V_4V_5})$.

It's with respect to the restricted angle of segment $\overline{V_2V_3}$ and segment $\overline{V_4V_5}$ and reflects the structure relationship between the second and the third defence-lines of 1-2-2 zone-defence.

X. $BST_{122} = \angle(\overline{V_6V_7}, \overline{V_2V_3})$.

which reflects the bias between the second and the third defence-lines of 1-2-2 zone-defence.

The feature vector is constructed by the above 10 features with respect to those 3 typical zone-defence strategies:

f = {CA_{23}, FSA_{23}, BCA_{23}, $RFSA_{23}$, CA_{131}, FSA_{131}, STA_{131}, RCA_{122}, $RSTA_{122}$, BST_{122}}

The feature vector is not only listed by the 10 components one by one, but also has internal relationships. The features of one typical zone-defence also reflect the structures relationship in other typical zone-defences.

4. Video Detection System of Basketball Zone-defence Strategy

According to the structure-based features extracted above, the test basketball zone-defence video clip with n key-frames (that is, n zone-defence graphs) can be represented by a $n \times 10$ feature matrix $F_{clip} = \{f_1, f_2, ..., f_n\}'$ and a ball's position vector $ball_{clip} = \{ball_1, ball_2, ..., ball_n\}$, where $f_i = \{f_{i1}, f_{i2}, ..., f_{i10}\}$ and $ball_i$ denotes the feature vector and the ball's position of the *i*th key-frame of the detected clip respectively. Analogously, the 3 standard zone-defence databases are represented by 3 corresponding feature matrices with their ball's position vectors respectively. For instance, the standard 2-3 zone-defence database is represented by $F_{23} = \{f_1^{23}, f_2^{23}, ..., f_{14}^{23}\}'$ and $ball_{23} = \{ball_1^{23}, ball_2^{23}, ..., ball_{14}^{23}\}$.

Firstly, compute the similarity between test clip and standard 2-3 zone-defence strategy.

Step 1: For each $f_i \in F_{clip}$, compute the Euclidean Distance(which has been experimented that performs better than other two famous distances Mahalanobis distance and Manhattan distance in our case) between f_i and each feature vector with the same ball position as f_i in standard 2-3 zone graph database:

$$ED(f_i, f_{z_j}^{23}) = [d_{ij}^{23}] \quad (2)$$

where $ball_i = ball_{z_j}^{23}$, $z_j \in \{1, 2, ..., 14\}$, $j = 1, 2, ..., n_p < n_{23}$, and n_p is the number of the graphs with the same ball position as G_i^{test} in 2-3 zone graph database.

Step 2: Determine the distance between f_i and 2-3 zone-defence strategy.

$$D_i^{23} = \arg\min_j([d_{ij}^{23}]) \tag{3}$$

Step 3: Compute the global distance between the test clip and 2-3 zone-defence strategy:

$$GD_{\text{test}}^{23} = \sum D_i^{23} \tag{4}$$

Secondly, in terms of the same procedure, we define the global distance between the test clip and 1-3-1 zone-defence strategy as:

$$GD_{\text{test}}^{131} = \sum D_i^{131} \tag{5}$$

(5)

Thirdly, we define the global distance between the test clip and 1-2-2 zone-defence strategy in the same manner as:

$$GD_{\text{test}}^{122} = \sum D_i^{122} \tag{6}$$

Finally, the zone-defence strategy pattern of the test zone-defence video clip is defined as:

$$Z^{test} = \arg\min(GD_{\text{test}}^{23}, GD_{\text{test}}^{131}, GD_{\text{test}}^{122}) \tag{7}$$

5. Experimental Results

The system has been tested with both simulated and real basketball zone-defence videos. Firstly, we formulated 40 simulated zone-defence video clips (key-frame sequences), where the scenario and the defenders' position of each video clip were constructed by the professional coaches according to their rich experience. We also collected about 1 hour of the real basketball zone-defence videos, including 112 clips containing 3 to 8 key-frames each as listed in Table 2. According to the detection system illustrated in Fig 2, each clip denotes once defence with a particular zone-defence strategy.

Table 2. The number structure of test data

	Zone-defence strategy	Total clips		Total key-frames
Simulated	2-3	20	40	145
	1-3-1	20		161
	1-2-2	20		128
Real-life	2-3	52	112	286
	1-3-1	31		221
	1-2-2	29		169

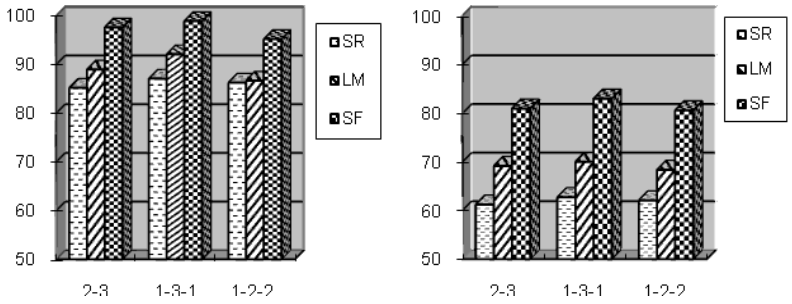
There are few systems focused on feature description of basketball zone-defence graphs. Here, we compare the algorithm proposed in this paper with LM-based algorithm [Zheng *et al*. 2009] and SR-based algorithm [Chin *et al*. 2005]. Table 3 below reports the detection result of each algorithm on both simulated and real-life data. Here detection results of "Correct MPD (Metric Position Detection)" are the results detected on the test clips with correct MPD. Generally speaking, the detected rate in simulated data is higher than that in real-life data for each approach. In particular, compared with the other approaches, as shown in Table 3, the structure feature (SF) based algorithm can detect more video clips in both simulated data and real-life data. This is due to the fact that the structure feature (SF) based algorithm takes into account of the structure relationship between defenders where it is neglected or inadequately dealt with in other algorithms. The results are more satisfied with regard to correct MPD since the correct MPD of defenders may lead much more likely to the correct detecting results.

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

Table 3. Detection result of 3 algorithms on different data

Database		**Video clips**		**Correct MPD**	
	Results	Test	Detected	Test	Detected
Simulated data	SR	40	35	38	34
	LM		36		35
	SF		**37**		**37**
Real-life data	SR	112	70	91	69
	LM		78		74
	SF		**91**		**85**

Fig.6 shows the detecting precision comparing with the other two algorithms in both simulated data and real-life data on each zone-defence. From Fig.6, one can see that the SF-based detecting method has the highest detecting precision in both simulated and real-life data, where the SR-based approach performs worst due to its inadequate dealing with the structure relationship between defenders.



(a) Simulated data (b) Real-life data

Fig.6 Detecting precision for each zone-defence pattern with different methods

It's frequent for defenders to have some translational motion comparing with the standard position in standard zone graphs. So the translational motion of the farthest defence-line from the ball in each zone-defence graph, which is regarded to have least influence to the global strategy, is added to the test video clip as a disturbance to test the robust of proposed approach. For each note V on the farthest defence-line in each zone-defence, we add the disturbance α as:

$$V' = V \pm \alpha(\cos\beta - \sin\beta) \tag{8}$$

where α denotes the movement distance of note V to V' and β denotes the angle between α and the x-axis (the mid-field line) as shown in Fig.7.

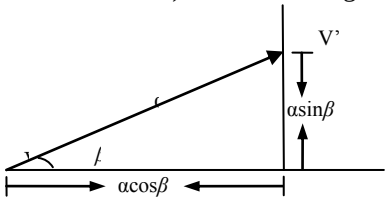
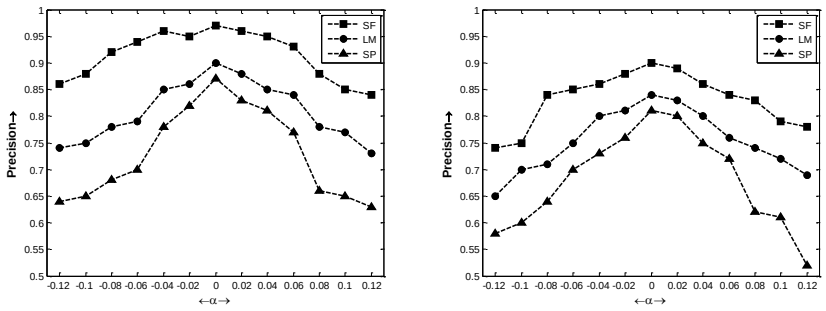


Fig.7 Disturbance of the node on the farthest defence-line

Fig.8 shows the efficiency in each zone-defence with different disturbance. In order to

eliminate the interference of the error from position detection, the statistics were calculated on the data with correct MPD.



(a) Simulated data (b) Real-life data

Fig.8 Precision influence with disturbance in each method

The precision comes down with growing disturbance in every method. But the SF-based method drops much slower than the other two and still has a tolerable performance even with a high disturbance, which demonstrates that the SF-based method is robust for the detecting system.

6. Conclusions and Future Work

In this paper, a structure-based feature descriptor describing the structure relationships between defence-lines has been proposed for video clip detection in basketball zone-defence. Comparing with other methods, the structure-based feature descriptor has a robust performance in both simulation and real-life applications especially when disturbance exists. It is reasonable and validly to describe the structure relationship between defenders in basketball zone-defence strategies. It is robust for the disturbance deriving from translational motion of defenders on subprime defence-lines.

For the future work, we shall extend the approach proposed in this paper to other team-work sports such as football, volleyball, etc., to describe the corresponding structure relationships. It is crucial to develop the corresponding metric position detection algorithms on zone-defence graphs which are very influential in the detection system. In addition, it seems reasonable and realistic to adopt clustering approaches and algorithms to develop generalized method(s) for various kinds of both existing and possible future zone-defence strategies. This remains also as future work.

References

Assfalg J., *et al.* (2003): Semantic annotation of soccer videos: automatic highlights identification. Computer Vision and Image Understanding, 92(2-3), pp:285-305.

Bagdanov A., *et al.* (2007): Semantic annotation and retrieval of video events using multimedia ontologies, Proc. of International Conference on Semantic Computing (ICSC), Irvine, California, pp:713-720.

Baldi G., Colombo C, Bimbo A. (1999): A compact and retrieval-oriented video representation using mosaics, Proc. of 3rd International Conference on Visual Information Systems VISual99, Springer Lecture Notes on Computer Science, Amsterdam, The Netherlands, pp:171–178.

APPENDIX C STRUCTURE BASED FEATURE EXTRACTION IN BASKETBALL ZONE-DEFENCE STRATEGIES

Babaguchi N., Kawai Y., Kitahashi T. (1999): Event Based Video Indexing by Intermodal Collaboration, Proc. First International Workshop on Multimedia Intelligent Storage and Retrieval Management (MISRM'99) in conjunction with ACM Multimedia Conference 1999, Orlando, pp:1-9.

Chang Y.L., *et al.* (1996): Integrated Image and Speech Analysis for Content-based Video Indexing, In IEEE Conf. On Multimedia Systems and Computing, pp:0306.

Chin S., *et al.* (2005): An Application Based on Spatial-Relationship to Basketball Defensive Strategies, Embedded and Ubiquitous Computing (EUC) Workshops, Nagasaki, Japan, Dec 8-9, pp:180–188.

Efros A.A., *et al.* (2003): Recognizing action at a distance, Ninth IEEE International Conference on Computer Vision (ICCV), Nice, France, vol. 2, pp: 726–733.

Gong Y., *et al.* (1995): Automatic Parsing of TV Soccer Programs, In IEEE Conf on Multimedia Computing and Systems, p.167.

Hua Q., Rui Y. (2007): Optimizing Multi-Graph Learning Towards A Unified Video Annotation Scheme, Proc. of the ACM International Conference on Multimedia (ACM MM), Augsburg, Bavaria, pp:17-26.

Liu S., *et al.* (2006): Multimodal Semantic Analysis and Annotation for Basketball Video, EURASIP Journal on Applied Signal Processing, pp:1–13.

Luo Y., Tzong-Der W., Jenq-Neng H. (2003): Object-based analysis and interpretation of human motion in sports video sequences by dynamic Bayesian networks, Computer Vision and Image Understanding 92 (2–3), pp:196–216.

Ma J., Zhao G., Hancock E. (2007): A Navigation-based Algorithm for Matching Scenario Patterns. Proceedings of International Conference on Artificial Intelligence and Pattern Recognition(AIPR-07), Orlando, Florida, pp:151-157.

Marco B, Bimbo A, Giuseppe S. (2008): Video Event Annotation using Ontologies with Temporal Reasoning, Proc. of 4th Italian Research Conference on Digital Library Systems (IRCDL), Padova, Italy, pp:24-25.

Nepal S., *et al.* (2001): Automatic detection of goal segments in basketball videos. In Proc. ACM Multimedia, Ottawa, Canada, pp:261-269.

Pan H., Van B. P., Sezan M.I. (2001): Detection of slowmotion replay segments in sports video for highlights generation, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp:1649-1652.

Perse M, *et al.* (2009): A trajectory-based analysis of coordinated team activity in a basketball game. Computer Vision and Image Understanding, Volume 113, Issue 5 pp612-621

Rui Y., Gupta A., Acero A. (2000): Automatically Extracting Highlights for TV Baseball Programs, Proceeding ACM Multimedia, p105-115.

Tan Y. P., *et al.* (2000): Rapid estimation of camera motion from compressed video with application to video annotation. IEEE Trans. CSVT, vol. CSVT-10, pp:133-146.

Urtasun R., Fleet D.J., Fua P. (2005): Monocular 3d tracking of the golf swing. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, June, pp: 932–938.

Xu H. and Chua T. (2004): The fusion of audio-visual features and external knowledge for event detection in team sports video, Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval, New York, NY, USA, pp:127-134.

Xu M., *et al.* (2004): HMM-Based Audio Keyword Generation, In Proc. PCM, Tokyo, Japan, pp:566-574.

Xu P., *et al.* (2001): Algorithms and systems for segmentation and structure analysis in soccer video. IEEE International Conference on Multimedia and Expo (ICME), Tokyo, Japan, pp: 184–187.

Zheng A., *et al.* (2009): Temporal Pattern Recognition in Basketball Video Clips. Proc. of 5th International Conference on Computer and Information Science, Shanghai, China, pp:416-421.

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING

A Robust Approach to Subsequence Matching

Aihua Zheng[1,2], Jixin Ma[2], Miltos Petridis[2] , Jin Tang[1], Bin Luo[1]

[1] Anhui University, Hefei, 230039, People's Republic of China
[2] The University of Greenwich, Greenwich, London, SE10 9LS, United Kingdom
E-mail: {a.zheng, j.ma, m.petridis}@gre.ac.uk, ahhftang@gmail.com, luobin@ahu.edu.cn

Summary. In terms of a general time theory which addresses time-elements as typed point-based intervals, a formal characterization of time-series and state-sequences is introduced. Based on this framework, the subsequence matching problem is specially tackled by means of being transferred into bipartite graph matching problem. Then a hybrid similarity model with high tolerance of inversion, crossover and noise is proposed for matching the corresponding bipartite graphs involving both temporal and non-temporal measurements. Experimental results on reconstructed time-series data from UCI KDD Archive demonstrate that such an approach is more effective comparing with the traditional similarity model based algorithms, promising robust techniques for lager time-series databases and real-life applications such as Content-based Video Retrieval (CBVR), etc.

1 Introduction

Time-series are typical patterns in date mining and knowledge discovery, particularly, in statistics, signal processing as well as other areas including rule discovery, prediction, detection, clustering and classification, and so on. Generally speaking, a time series presents a sequence of data, measured and/or spaced typically at successive times, which can be either points or intervals.

The notion of state is fundamental for many state-based applications, which represents the static snapshot of the world in discourse, while the dynamic historical scenarios of the world can be characterized in terms of temporally ordered state-sequences. State-sequence matching has been noticed as a popular research topic in time-series data which has attracted a lot of researchers' interests. In particular, how to find out the most similar patterns for the query state-sequence in time-series data is an essential work for many real life state-based applications. Normally, state-sequence matching can be divided into two categories: whole matching [2, 4] (each sequence has the same length) and subsequence matching [10, 13] (with various lengths). Obviously, the whole matching problem is in fact a special case of the subsequence matching which we shall tackle in this paper.

One of the popular topics in subsequence matching is the similarity model between state-sequences. Specially, temporal similarity between state-sequences plays a vital role, where three aspects regarding the temporal information of state-sequences need to be

This research is supported in part by National Nature Science Foundation of China (No. 60772122)

addressed: (1) The "before/after" relations over the states which decide how state-sequences are temporally ordered; (2) The temporal distances between pairs of consecutive states; and (3) The duration of various times over which the corresponding states are associated with.

Various similarity models based on Euclidean distance, have been specially introduced for subsequence matching, [2]. An efficient category of these is the so-call sliding window based algorithms [9, 10]. However, most of them are very brittle even with slight misalignment in time axis and the time-consuming problem limits their application on large database. Subsequently, some successful models such as Dynamic time warping (DTW) [6], Longest Common Sequence (LCSS) [15], Edit distance [1] and their variants have been proposed. DTW is robust to time warping such as stretching and shrinking. However, no states will be skipped including noise. LCSS can skip some states including outliers but ignores how many states it skips. ED takes into account of the number being skipped, however, which kind of states being skipped is ignored. And common reordering such as crossover or backward is not allowed.

In addition, in most of proposed formalisms and models, the fundamental time theories based on which time-series and state-sequences are formed up are usually not explicitly specified. Time-series are simply expressed as lists in the form of t_1, t_2, …., t_n, or as sequences of collection of data, and so on, where formal characterizations with respect to the temporal basis have been neglected.

The objective of this paper is to present a robust framework for matching subsequence patterns. As the fundamental formalism, a formal characterization of time-series and state-sequence is introduced in section 2, A bipartite graph representation for subsequence matching is presented in section 3. Section 4 introduces a hybrid similarity model which integrates both non-temporal similarity and temporal similarity. Experimental results on UCI time-series data are provided, analyzed and evaluated in section 5. Finally, section 6 provides a brief summary and concludes the paper with the prospects for future work.

2 Time-elements, time-series and state-sequences

For general treatments, in this paper, we shall define time-elements as typed point-based intervals, allowing expression of both absolute time values and relative temporal relations [7]. We shall use R to denote the set of real numbers, and T, the set of time-elements. Each time-element t is defined as a typed (left-open & right-open, left-closed & right-open, left-open & right-closed, left-closed & right-closed) subset of the set of real numbers R. I.e., each time-element must be in one of the following four forms:

$$(p_1, p_2) = \{p \mid p \in R \wedge p_1 < p < p_2\}, [p_1, p_2) = \{p \mid p \in R \wedge p_1 \le p < p_2\}$$
$$(p_1, p_2] = \{p \mid p \in R \wedge p_1 < p \le p_2\}, [p_1, p_2] = \{p \mid p \in R \wedge p_1 \le p \le p_2\}$$

In the above, p1 and p2 are real numbers, which are called the left-bound and right-bound of time-element t, respectively. The absolute values as for the left and/or right bounds of some time-elements might be unknown. In this case, real number variables are used for expressing relative relations to other time-elements (see later). In addition, if the left-bound and right-bound of time-element t are the same, it is called a time point;

otherwise it is called a time interval. Without confusion, time-element [p, p] is taken as identical to point p. Also, if a time-element is not specified as open or closed at its left (right) bound (that is, the left (right) type of the time-element is unknown), we shall use "<" (or ">") instead of "(" and "[" (or ")" and "]") as for its left (or right) bracket. Also, the duration of a time-element t, Dur(t), is defined as the difference between its left bound and right bound. In other words:

$$t = \langle p_1, p_2 \rangle \Leftrightarrow Dur(t) = p_2 - p_1$$

Following Allen's terminology [3], we shall use "Meets" to denote the immediate predecessor order relation over time-elements, which can be formally defined as:

$$Meets(t_1, t_2) \Leftrightarrow \exists p_1,p,p_2 \in R(t_1 = (p_1, p) \wedge t_2 = [p, p_2) \vee t_1 = [p_1, p) \wedge t_2 = [p, p_2)$$
$$\vee t_1 = (p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = [p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2)$$
$$\vee t_1 = [p_1, p] \wedge t_2 = (p, p_2) \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2] \vee t_1 = [p_1, p] \wedge t_2 = (p, p_2])$$

It is easy to see that the intuitive meaning of $Meets(t_1, t_2)$ is that, on the one hand, time-elements t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point); on the other hand, there is not any other time-element standing between them.

Analogous to the 13 relations introduced by Allen for intervals [3], there are 30 exclusive temporal order relations over time-elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations that relate points to points:
  {Equal, Before, After}
- Relations that relate points to intervals:
  {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations that relate intervals to points:
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations that relate intervals to intervals:
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

The definition of these derived temporal order relations in terms of the single relation Meets is straightforward. E.g.: $Before(t_1, t_2) \Leftrightarrow \exists t \in T(Meets(t_1, t) \wedge Meets(t, t_2))$.

Based on such a time theory, a time-series ts is defined as a vector of time-elements temporally ordered one after another [8]. Formally, a general time-series is defined in terms of the following schema:

GTS1) $ts = [t_1, \ldots, t_n]$

GTS2) $Meets(t_j, t_{j+1}) \vee Before(t_j, t_{j+1})$, for all $j = 1, \ldots, n-1$

GTS3) $Dur(t_k) = d_k$, for some k where $1 \leq k \leq n$, d_i is a non-negative real number.

Generally speaking, a time-series may be incomplete in various ways [7]. Correspondingly, a complete time-series is defined in terms of the schema as below:

CTS1) $ts = [t_1, \ldots, t_n]$

CTS2) $Meets(t_j, t_{j+1})$, for all $j = 1, \ldots, n-1$

CTS3) $Dur(t_i) = d_i$, for all $i = 1, \ldots, n$ where d_i is a non-negative real number.

The validation of data is usually dependent on time. We shall use fluents to represent Boolean-valued, time-varying data, and denote proposition "fluent f holds true over time t" by formula Holds(f, t) [3]:

(F1) $(f, t) \Rightarrow \forall t_1(Part(t_1, t) \Rightarrow Holds(f, t_1))$

That is, if fluent f holds true over a time-element t, then f holds true over any part of t.

(F2) $\forall t_1(\text{Part}(t_1, t) \Rightarrow \exists t_2(\text{Part}(t_2, t_1) \wedge \text{Holds}(f, t_2))) \Rightarrow \text{Holds}(f, t)$

That is, if any part of time t contains a part of itself over which fluent f holds true, then f holds true over t. Here, $\text{Part}(t_1, t) \Leftrightarrow \text{Equal}(t_1, t) \vee \text{Starts}(t_1, t) \vee \text{During}(t_1, t) \vee \text{Finishes}(t_1, t)$.

(F3) $\text{Holds}(f_1, t) \vee \text{Holds}(f_2, t) \Rightarrow \text{Holds}(f_1 \vee f_2, t)$

That is, is fluent f_1 or fluent f_2 holds true over time t, then at least one of them holds true over time t.

(F4) $\text{Holds}(\text{not}(f), t) \Leftrightarrow \forall t_1(\text{Part}(t_1, t) \Rightarrow \neg\text{Holds}(f, t_1))$

That is, the negation of fluent f holds true over time t if and only if fluent f does not hold true over any part of t.

(F5) $\text{Holds}(f, t_1) \wedge \text{Holds}(f, t_2) \wedge \text{Meets}(t_1, t_2) \Rightarrow \text{Holds}(f, t_1 \oplus t_2)$

That is, if fluent f holds true over two time-elements t_1 and t_2 that meets each other, then f holds over the ordered-union of t_1 and t_2.

A state is defined as a collection of fluents. Following the approach proposed in [11], we shall use Belongs(f, s) to denote that fluent f belongs to the collection of fluent representing state s. For the reason of simple expression, if $f_1, \ldots, f_m$ are all the fluents that belong to state s, we shall represent s as $<f_1, \ldots, f_m>$. Also, without confusion, we shall use formula Holds(s, t) to denote that s is the state of the world with respect to time t, provided that:

(F6) $s = <f_1, \ldots, f_m> \Rightarrow (\text{Holds}(s, t) \Leftrightarrow \text{Holds}(f_1, t) \wedge \ldots \wedge \text{Holds}(f_m, t))$

That is, a state s holds true over time t if and only if every fluent in the s holds true over time t.

A state-sequence ss is defined as a list of states together with its corresponding time-series ts. A general state-sequence is defined in terms of the schema as below:

GSS2.1) $ss = [s_1, \ldots, s_n]$

GSS2.2) $\text{Holds}(s_i, t_i)$, for all $i = 1, \ldots, n$

where $[t_1, \ldots, t_n]$ is a time-series. Correspondingly, a state-sequence is defined as complete if and only if the corresponding time-series is complete [8]**.**

3 Bipartite graphical representation for subsequence matching

We shall systematically introduce the procedure of transforming subsequence matching into bipartite graph matching problem in this section.

Table 1. Notations used in this paper

Notation	Definition
$Q = [q_1, q_2, \ldots, q_m]$	Query state-sequence
$SS = [s_1, s_2, \ldots, s_n]$	A state-sequence in database
$D = [SS_1, \ldots, SS_L]$	The database with L state-sequences
$NN(q_i, SS, k)$	Set of kNN of q_i in SS
$NN(Q, SS, k)$	Set of kNN of all q_i in Q in SS
$BG = <Q, SS, E>$	Bipartite graph between Q and SS
$MSM(Q, SS)$	The set of MSM between Q and SS

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING

$MSM(Q, D)$	The set of MSM between Q and all SS_j in D
M	A normal matching in $MSM(Q, D)$
$\overleftarrow{M}$	An inverse-ordered matching of M

The list of notations that will be used in this paper is given in Table 1. The procedure can be briefly described as following:

Step 1: Employ Dynamic Query Ordering (*DQO*) algorithm [12] to implement *kNN* (*k* Nearest Neighbours) search for each state q_i in *SS* within a given maximum distance d_{max}. Output the state set $NN(q_i, SS, k)$ for each state q_i. NB.: $|NN(q_i, SS, k)| \in [0, k]$.

Step 2: Construct un-weighted bipartite graph $BG = <Q, SS, E>$ for $NN(Q, SS, k)$, $E \subseteq Q \times SS$ is the edge set denoting *kNN* mapping between *Q* and *SS*, as showing in Fig.1:

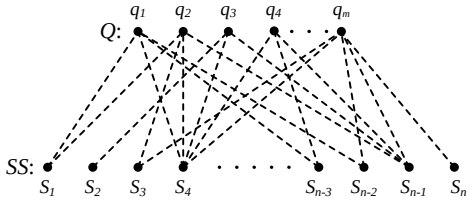


Fig.1 Bipartite graph representation

Step 3: Employ Maximum Size Matching (*MSM*) algorithm [14] to produce a set of 1-1 matching between *Q* and *SS* with the maximum size for the corresponding *BG*. NB.: the output of *MSM* in general is not unique.

For each given state-sequence *SS*, the above procedure produces a corresponding set of 1-1 matching $MSM(Q, SS)$ between *Q* and *SS* with the maximum size. Therefore, if we denote the set of such matching between *Q* and all SS_j in *D* as $MSM(Q, D)$, we have:

$$MSM(Q, D) = \bigcup_{j=1}^{L} MSM(Q, SS_j) \tag{1}$$

The remaining main problem is then to develop an appropriate similarity measurement for searching the corresponding optimal matching.

4 Hybrid similarity model

As mentioned earlier, for a given a matching $M \in MSM(Q, D)$, both temporal similarity and non-temporal similarity should be taken into account. On one hand, the non-temporal similarity is defined according to the Euclidean distance between each mapping.

Non-temporal similarity: The non-temporal similarity is measured by the total similarity which is in inverse proportion to the Euclidean distance between each matched state pair.

$$Sim_{NT} = \sum (1 - dis(q_i, s_j)) \Big/ (\sqrt{d}\,|Q|) \tag{2}$$

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING

where $dis(q_i, s_j)$ denotes the Euclidean distance between each matched state pair q_i and s_j (which has achieved during *kNN* search) in the matching M and d denotes the feature dimension of each state. Obviously, the similarity value falls into [0, 1].

On the other hand, as the distinctive feature of time-series data, temporal similarity needs special treatments with respect to the following three measurements:

Temporal order similarity: There may be some pairs of state-sequences with the same non-temporal similarity but with different temporal order. Here, we shall use the idea of LCSS [15] to measure temporal order similarity. However, in existing normal LCSS based formalisms, the typical reordering situations inversion in time-series data have been neglected. In order to catch such kind of reordering, we define the temporal order similarity as below:

$$Sim_{TO} = \max(LCS(M), LCS(\overleftarrow{M}))/|Q| \tag{3}$$

which takes into account of both normal order and inverse order.

Temporal alignment similarity: In normal LCSS formalisms, in subsequence matching, unmatched states are simply skipped regardless how many of them. ED [1] is an alternative measurement distinguishing the number of unmatched states that being skipped. However, crossover, which should be compatible since it is ubiquitous, is not allowed in ED since it just matches in the single forward direction. Following the approach proposed in [13], we define the following temporal alignment similarity:

$$Sim_{TA} = 2|M|/(|Q|+|SS|) \tag{4}$$

which takes into account the number of unmatched states and accepts crossover.

Temporal concentration similarity: It is easy to see that the distribution of matched (or unmatched) states and the internal temporal distance (or similarity) is ignored in Sim_{TA}. For instance, by (4), sequence [1, 2, 3, 4, 5] will be taken as having the same similarity with [1, a, a, 2, 3, 4, a, 5], [1, 2, a, 3, a, 4, a, 5] and [1, b, c, 2, 3, 4, d, 5]. In addition, the duration of various times over which the corresponding states are associated with is not addressed in (4). Here, we introduce a similarity measurement to govern such temporal concentration. In what follows in this paper, we use *CD* and *DD* to denote the Concentration similarity Degree and the Discrete similarity Degree:

$$CD = Dur(CMS_1) + \sum_{i=2} (Dur(CMS_i) \Big/ \sum_{t=1}^{i} Dur(CMS_t)) \tag{5}$$

$$DD = \sum_i (\lambda_i Dur(CUS_i) \Big/ \sum Dur(CUS)) \tag{6}$$

where CMS_i and CUS_i are defined as "Continuous Matched Subsequence" and "Continuous Unmatched Subsequence", respectively, in descending ordered with respect to the length of these subsequences; and $Dur(CMS)$ and $Dur(CUS)$ denote the list of the duration of each continuous subsequence in *CMS* and *CUS*, respectively. λ represents the internal temporal distance with respect to each adjacent continuous matched and unmatched subsequences. In fact, if $CUS_i = [s_t, \ldots, s_p]$

$$\lambda = \begin{cases} \sum_{i=t}^{p} dis(s_{p+1}, s_i) \Big/ |CUS_i| & if \ \ t = 1 \\ \sum_{i=t}^{p} dis(s_{t-1}, s_i) \Big/ |CUS_i| & if \ \ p = length(SS) \\ \sum_{i=t}^{p} (dis(s_{t-1}, s_i) + dis(s_{p+1}, s_i)) \Big/ 2|CUS_i| & else \end{cases} \tag{7}$$

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING

In order to reduce the computing complexity, we replace s_{t-1} and s_{p+1} by their corresponding query states in Q since the Euclidean distance in (2) between each state in Q and a state in SS has achieved in the *kNN* search stage.

The temporal concentration similarity can be defined:

$$Sim_{TC} = (CD - DD)/|Q| \tag{8}$$

Normally, the overall similarity can be simply defined as the average of individual similarities. However, as we have argued earlier, the individual similarity measurements introduced in this paper have various features. In fact, while the non-temporal similarity and the temporal similarity may be treated in parallel, the three temporal similarities are progressive one after the other. Therefore, it is not appropriate to simply accumulate all of them together. In what follows, we use a hybrid approach to combine the four similarity measurements.

Hybrid similarity model:

Step 1: reorder $MSM(Q, D)$ as $MSM(Q, D)$' by Sim_{TO}, Sim_{TA}, and Sim_{TC}:

Firstly, reorder it by the Sim_{TO}; then for the matchings with the same Sim_{TO}, reorder them by Sim_{TA}; analogously, reorder by Sim_{TC} if there exists some matchings with the same Sim_{TA}.

Step 2: Integrate temporal similarity: get the integrated temporal similarity $Sim_{TS} = Adjust(Sim_{TO})$. For those $\mu = j\text{-}i+1$ matchings $[M'_i,\ldots,M'_j]$ with the same Sim_{TO}, evenly stretch their similarities into $[Sim_{TO}+\sigma/2, Sim_{TO}-\sigma/2]$ where σ denotes the adjust operator defined as below:

$$\sigma = \begin{cases} (Sim_{TO_{i-1}} - Sim_{TO_{j+1}})/3\mu & if \ \ i \neq 1, j \neq x \\ Sim_{TO_{j+1}}/2\mu & if \ \ i = 1 \\ Sim_{TO_{i-1}}/2\mu & if \ \ j = x \end{cases} \tag{9}$$

Step 3: Overall similarity: reorder $MSM(Q, D)$' as $MSM(Q, D)$'' in terms of overall similarity Sim which defined as the average of non-temporal similarity and integrated temporal similarity:

$$Sim = (Sim_{NT} + Sim_{TS})/2 \tag{10}$$

5 Experimental Results

We experiment our method on Synthetic Control Chart Time Series in UCI KDD (Knowledge Discovery in Databases) Archive [5]. The database consists of 600 synthetically generated control charts state-sequences with length of 60 for each, including 6 different classes (100 examples each): In order to avoid the influence of segmenting error to the proposed similarity model, we shall use the original database in the form of individual 600 state-sequences with length 60 for each as the training data. Several query sets are reconstructed as following:

Original Query Set (*OQS*): which consists of 60 (the first 10 state-sequences from each class) state-sequences;

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING

Reordered Query Set (*RQS*): α percent reordered (in inverse order while α=1) to each state-sequence in *OQS*;

Shortened Query Set (*SQS*): each state-sequence of this set is with length of (1-β)*60;

Noised Query Set (*NQS*): add a Gaussian noise to each state-sequence in *OQS*.

For each query state-sequence, by means of following the procedure presented in section 3 we obtain a set of optional matching in the training database, and according to the hybrid similarity model proposed in section 4, we then calculate the overall similarity respectively. The precision is defined as the ratio of the number of state-sequences with the same class as the query state-sequence out of the first 100 optimal matching in *MSM*(*Q, D*)''. We focus on the performance of our similarity model compared with that of [13] (shorthanded by Shen in following figures), which is just simply defined by the average of its individual similarity measurements. Meanwhile, another two models which employ ED and LCSS as temporal similarity have been tested respectively.

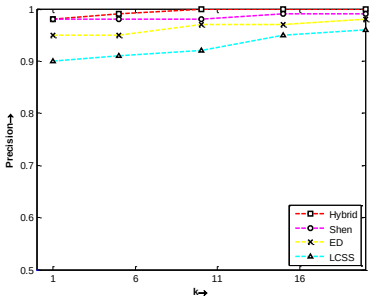


Fig. 2 Precision of *OQS* against *k*

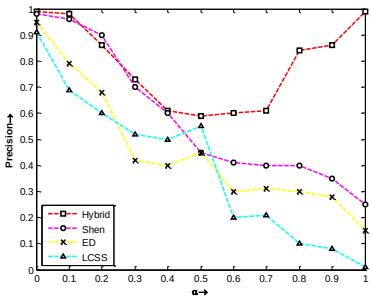


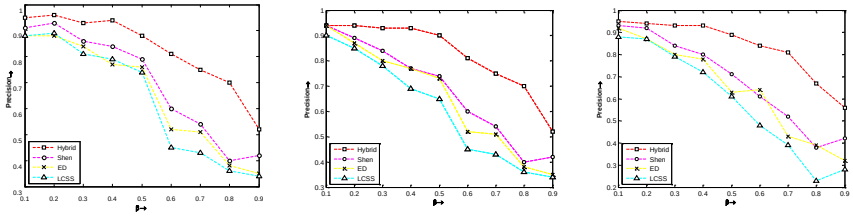
Fig. 3 Precision of *RQS* against α

Fig. 2 shows the precision of the *OQS* with different k in *kNN* search. We can see that there is no distinct influence of *k*. In order to reduce the complexity of our matching system, we default *k*=5 if not specified. The following 3 figures show the matching precision on different query sets against their corresponding reconstructive parameters.

Firstly, Fig.3 shows the precision of *RQS* against α, in order to reveal the performance of the progressive temporal similarity measurement we proposed in this paper, we omit the non-temporal similarity in each method. From which we can see, in our method, the precision has an approximate quadratic distribution with the subject to α, which means it can better detect the reordered state-sequences than the others.

Then, to evaluate the effect of β, we form the *SQS* by deleting β*60 states in different position: evenly, from the beginning and the end. Fig.4 shows the matching results against different β. Generally speaking, our method is more robust than others no matter the state-sequences are shortened evenly, from the beginning or from the end. The precision drops much slower in our method especially for $\beta \in [0.1, 0.5]$. In addition, according to our statistic, the query set shortened from the beginning has a slight higher precision than the other two sets shortened evenly and at the end in our similarity model. Generally speaking, the position (where being shortened) doesn't affect the precision very much in any similarity model.

APPENDIX D A ROBUST APPROACH TO SUBSEQUENCE MATCHING



(a) Evenly (b) From the beginning (c) From the end

Fig. 4 Precision of *SQS* against β

Fig.5 shows the results of noised data with Gaussian noise in different mean ([0, 2]) and variance ([0.1, 1]). Visually, our method has higher precision and smaller fluctuation. Table 2 below shows the average mean and standard deviation (STD) of each subfigure in Fig.5.

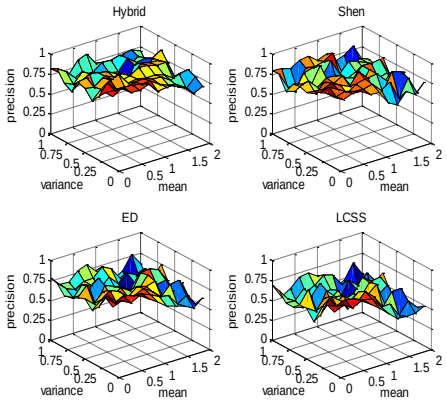


Fig. 5 Precision of query set with Gaussian noise against mean and variance

Table 2. Statistic of the precision of noised query set

	Hybrid	**Shen**	**ED**	**LCSS**
Mean (%)	76.46	72.81	68.83	59.12
STD	0.0764	0.0878	0.0877	0.1043

6 Conclusion and future work

In this paper, based on a formal characterization of time-series and state-sequences, we introduced a framework for subsequence matching. A hybrid similarity model addressing both non-temporal and temporal relationship between state-sequences, which are represented by

bipartite graphs, has been proposed. The experimental results on UCI time-series database demonstrated that the proposed similarity model is robust to states alignment with different numbers and different values, and various reordering including inversion, crossover, compared with the LCSS and ED based similarity models. We hope this model will provide a steady usage on lager time-series databases and real-life applications such as Content-based Video Retrieval.

References

[1] Adjeroh D, Lee M & King I.: A distance measure for video sequences. Computer Vision and Image Understanding. 75(1-2), 1999, pp: 25-45.

[2] Agrawal R, Faloutsos C, & Swami A.: Efficient similarity search in sequence databases. In Proc. of the 4th Int'l Conf. on Foundations of Data Organization and Algorithms, Chicago, Illinois, USA , Oct 13-15, 1993, pp:69-84.

[3] Allen J.: Towards a General Theory of Action and Time. Artificial Intelligence 23, 1984, pp:123-154.

[4] Beckmann N, Kriegel H, Schneider R & Seeger B.: The r*-tree: An efficient and robust access method for points and rectangles. In Proc. of the 1990 ACM SIGMOD Int'l Conf. on Management of Data, Atlantic City, NJ, May 23-25, 1990. pp:322-331.

[5] http://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.html

[6] Keogh E.: Exact indexing of dynamic time warping. in Proc. of the 28th Int'l Conf. on Very Large Data Bases, Hong Kong, China, Aug20-23, 2002, pp:406–417.

[7] Ma J & Hayes P.: Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?. the Computer Journal, 49(1), 2006, pp:32-41.

[8] Ma J, Bie R, Zhao G.: An ontological Characterization of Time-series and State-sequences for Data Mining. Proc. of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, Jinan, Shandong, Oct 18-20, 2008, pp:325-329.

[9] Moon Y, Whang K & Han W.: General Match A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows. In Proc. of the 8th ACM SIGMOD Int'l Conf. on Management of data, Madison, Wisconsin, USA, Jun 4-6, 2002, pp:382 - 393.

[10] Moon Y, Whang K & Loh W.: Duality-based subsequence matching in time-series databases. In Proc. of the 17th Int'l Conf. on Data Engineering, Santa Barbara, California, May 21-24, 2001, pp: 263-272.

[11] Shanahan M.: A Circumscriptive Calculus of Events. Artificial Intelligence 77, 1995, pp:29-384.

[12] Shao J, Huang Z, Shen H, Zhou X, Lim E & Li Y.: Batch nearest neighbour search for video retrieval. IEEE Transactions on Multimedia, 10(3), 2008, pp:409-420.

[13] Shen H, Shao J, Huang Z & Xiaofang Zhou.: Effective and Efficient Query Processing for Video Subsequence Identification. IEEE Transactions on Knowledge and Data Engineering, 21(3), 2009, pp:321-334.

[14] Shier D.: Matchings and assignments. in Handbook of Graph Theory, J. L. Gross and J. Yellen, Eds. CRC Press, 2004, pp:1103–1116.

[15] Vlachos M, Gunopulos D & Kollios G.: Discovering similar multidimensional trajectories. In Proc. of the 18th Int'l Conf. on Data Engineering, San Jose, CA,USA, Feb 26-Mar 1 2002, pp:673–684.

Efficient and Effective State-based Framework for News Video Retrival

Aihua Zheng[1,2], Jixin Ma[1], Xiaoyi Zhou[1], Bin Luo[2]
[1]*The University of Greenwich, London, UK*
[2]*Anhui University, Hefei, China*
{a.zheng, j.ma, x.zhou }@gre.ac.uk luobin@ahu.edu.cn

Abstract

In this paper, an efficient and effective framework is proposed for news video retrieval. Firstly, the 64-dimensional colour histogram is extracted as the feature vector. Then the pair quantizer is adopted to transfer the news video retrieval problem into multi-dimensional string matching problem, which conduces to the efficiency to the framework. Secondly, a new measurement named 'optimal temporal common subsequence', which distinguishes the difference caused by rich temporal characteristics including temporal order, temporal duration and temporal gap, is designed to match state-sequence, followed by the point & interval-based formal characterization of time-series and state-sequences. Thirdly, we tested the proposed measurement on news video retrieval. The performance shows the proposed algorithm is more effective for news video retrieval.

Keywords*: state-sequence matching; optimal temporal common subsequence; news video retrieval*

1. Introduction

With the development and the progress of information age, multimedia information, especially video information, is becoming an active and hot research object including video retrieval, video structural representation, video annotation and so on. Content-based Video Retrieval (CBVR) has attracted more and more researchers in recent decades. Normally, video database can be organized as figure 1. Videos are stored in terms of clips each of which contents a sequential key frames (static images) obtained by specific key frame extraction algorithm. In order to cater for recognition or matching, feature vectors are extracted for key frames. From figure 1, we can see, the video retrieval can actually be transformed into the matching problem between feature vector sets where feature vectors are sequential.

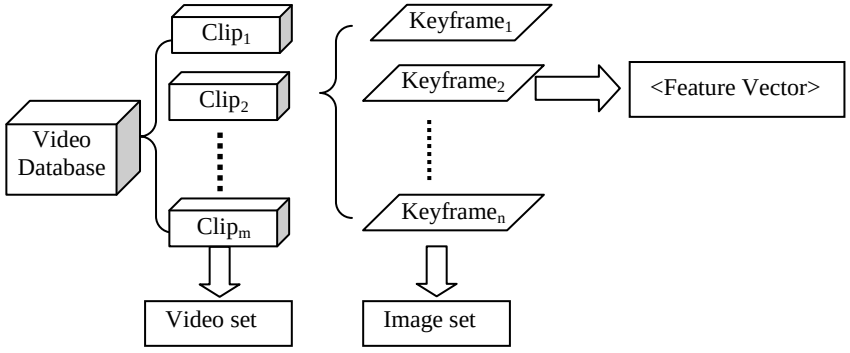


Figure 1. Video database organization

Different from image retrieval, the task is to search out the most similar image (key frame) set, not only the single image. Which in turn means the temporal relationship between key frames should be highly regarded. State-sequence matching, as an effective approach in temporal pattern recognition, has been actively researched recently, where the key frames in videos are regarded as states in time-series.

The notion of state is fundamental for many state-based applications, which represents the static snapshot of the world in discourse, while the dynamic historical scenarios of the world can be characterised in terms of temporally ordered state-sequences. Generally speaking, a state-sequence presents a sequence of data, measured and/or spaced typically at successive times, which can be either points or intervals. State-sequence matching has been noticed as a popular research topic in state-based systems has been well applied in various areas such as

financial data analysis [1], audio recognition [2], visual information retrieval [3], etc. Normally, state-sequence matching can be divided into two categories: whole matching [4, 5] (i.e., all state-sequences have the same length) and subsequence matching [3, 6] (i.e., state-sequences have various lengths). Obviously, the whole matching problem is in fact a special case of the subsequence matching. In general, state-sequence matching needs to accommodate three temporal features:

i. Temporal Order: the temporal relation over the states in the two given state-sequences. This issue has been well dealt with in most existing state-sequence matching algorithms benefiting from the dynamic programming.

ii. Temporal Duration:

- The duration of each state. For instance, as shown in Figure. 2, the two state-sequences A_1 and A_2, that is 'abcd', have different temporal duration assignment function T_{dur1} = [1, 1, 1, 1] and T_{dur2} = [1, 2, 3, 4], respectively.
- The overall duration of continuous duplications of states. For instance, as shown in Figure. 2, for state-sequences A_1 = 'abcd' and A_3 = 'aabbbccccddd', the common subsequence 'abcd' have various overall duration, even if the duration value of each state is identical as 1.

iii. Temporal Gap: the time element standing between two adjacent states as shown in Figure. 3, where B_1 and B_2 = 'abcd', B_3 = 'aabbbcd' are with different temporal gap values.

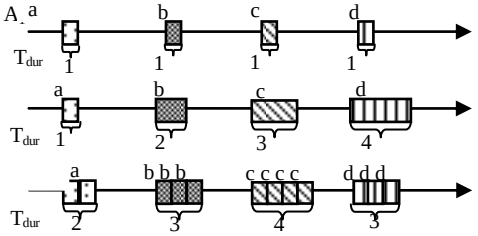


Figure 2. Various Temporal Duration in State-sequences

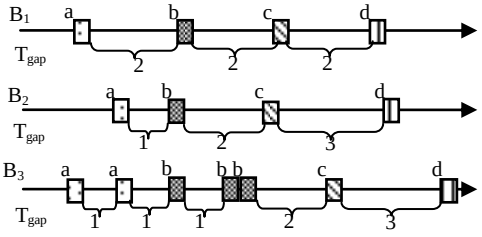


Figure 3. Various temporal gap in state-sequences

The Longest Common Subsequence (LCS) is a typical similarity measurement for subsequence matching.The basic idea of the original LCS algorithm [7] is to find the longest subsequence common to two state-sequences along the same temporal order. For instance, the longest common subsequence of A3 and B3 is 'aabbbcd'. In this paper, distinguished from this concept of common subsequence in conventional LCS, we define the temporal common subsequence of two state-sequences as the common subsequence where each state is different from its neighbour(s) (predecessor and successor), that is, there are no continuous duplications of states in temporal common subsequence. For instance, the temporal common subsequence of A3 and B3 is 'abcd', rather than 'aabbbcd'. Correspondingly, the optimal temporal common subsequence (OTCS) is the one with the highest overall similarity integrated by the length of temporal common subsequence, the temporal duration difference and temporal gap difference (see the actual algorithm in section III).

Several algorithms based on the original LCS have been proposed. Some representative variants of these are: Time-warped LCS (T-WLCS) [8] which counts continuously duplicated common states in the spirit of Dynamic Time Warping (DTW) [9] algorithm; Compacted LCS (CLCS) [10] where only the common subsequence, the

continuous length of which is longer than the specified threshold (*th*) is counted; All Common Subsequence (ACS) [11] which measures the similarity by means of counting the number of all common subsequences (including empty string in actual algorithm) and taking the strategy that the more common subsequences a pair of state-sequences have, the more similar they are.

However, in most of these representative algorithms, many problems (see details analysed in section III) occur due to the neglect of richer temporal features such as temporal duration and temporal gap, etc. While time-series and state-sequences have been simply expressed as ordered lists $t_1, t_2, \ldots, t_n$ (or $s_1, s_2, \ldots, s_n$), leaving some critical issues unaddressed. E.g.:

- What a sort of objects do these t_1, t_2, … and t_n belong to? In other word, are they time points, time intervals, or simply some absolute values from the real numbers, integers, or the clock?
- What are the temporal order relationships between these t_1, t_2, … and t_n, and/or between the sequence of collections? Are they simply well-ordered as the natural numbers, or they may be relatively ordered by means of relations such as "Before", "Meets", "During", and so on?
- What are the associations between time-series/ sequences and non-temporal data that represent various states of the world in discourse?

The objective of this paper is to design an effective and efficient framework for news video retrieval. The rest of this paper is organised as below: the quantization procedure is presented in section II. The formal characterization of time-series and state-sequences is introduced in section III. An Optimal Temporal Common Subsequence (OTCS) algorithm based on a formal characterization of time-series and state-sequences is presented and analyzed in section IV. Experiments on news video retrieval system are conducted and the corresponding results are analysed in section V to demonstrate the effectiveness and validity of the proposed OTCS. Section VI provides a brief summary and concludes the paper with the prospects for future work.

2. Video clip and state-sequence

As mentioned in the introduction, the key frames in video clips are regarded as states in time-series, which in turn means the video clips are regarded as state-sequences. In order to apply state-sequence matching algorithm to video clip retrieval, quantization is employed to map the sequential feature vectors into assigned character bins. The uniform quantization is the most common and efficient choice which can be defined as:

$$(k-1)\bullet S_{Step} < B_k \le k\bullet S_{Step}, k = 0,1,...N \tag{1}$$

where N denotes the number of the bin. $S_{Step} = \|Max-Min\|/\mathrm{N}$ denotes the step size and the Euclidian distance is employed to calculate the maximum value (*Max*) and the minimum value (*Min*) among feature vectors. By this quantization, most of the similar feature vectors, the distance between which is within the tolerance (step-size) will be quantified into the same bin. However, the similar feature vectors may be mapped into different bins if they are located on different sides of the cut edges ($k\bullet S_{Step}, k = 1,...N-1$), even though they are very similar to each other.

Therefore, in this paper, we adopt the paired quantization method [10] for feature quantization. The two quantizers Q_1, Q_2 are defined as following:

$$(k-1)\bullet S_{Step} < B_k^{Q_1} \le k\bullet S_{Step} \tag{2}$$

$$(k-1)\bullet S_{Step} + \frac{S_{Step}}{2} < B_k^{Q_2} \le k\bullet S_{Step} + \frac{S_{Step}}{2} \tag{3}$$

The feature vector will be quantified into the *k*th bin if it satisfies either quantizer Q_1 or Q_2. So it can relieve the problem pointed in single quantizer.

3. Formal characterization of time-series and state-sequences

APPENDIX E EFFICIENT AND EFFECTIVE STATE-BASED FRAMEWORK FOR NEWS VIDEO RETRIVAL

In this section, we introduce a formal characterization of time-series and state-sequences. For the sake of allowing expression of both absolute time values and relative temporal relations, in this paper, time-elements are defined as typed point-based intervals, each of which must be in one of the following four forms [12]:

$$(p, q) = \{r \mid r \in R \wedge p < r < q\}$$
$$[p, q) = \{r \mid r \in R \wedge p \leq r < q\}$$
$$(p, q] = \{r \mid r \in R \wedge p < r \leq q\}$$
$$[p, q] = \{r \mid r \in R \wedge p \leq r \leq q\}$$

In the above, R stands the set of real numbers, and real numbers p and q are called the left-bound and right-bound of time-element t, respectively. The absolute values as for the left and/or right bounds of some time-elements might be unknown. In this case, real number variables are used for expressing relative relations to other time-elements (see later). If the left-bound and right-bound of time-element t are the same, t is called a time point; otherwise it is called a time interval. Without confusion, time-element [p, p] is taken as identical to point p. Also, if a time-element is not specified as open or closed at its left (right) bound (that is, the left (right) type of the time-element is unknown), we shall use "<" (or ">") instead of "(" and "[" (or ")" and "]") as for its left (or right) bracket. In addition, the temporal duration of a time-element t, Tdur(t), and the temporal gap between adjacent elements t_1, t_2, Tgap (t_1, t_2) can be defined as below:

$$t = <p, q> \Leftrightarrow T_{dur}(t) = q - p$$
$$t_1 = <p_1, q_1>, t_2 = <p_2, q_2> \Leftrightarrow T_{gap}(t_1, t_2) = |p_2 - q_1|$$

Following Allen's terminology [13], we shall use "Meets" to denote the immediate predecessor order relation over time-elements, which can be formally defined as:

$$\begin{aligned}\text{Meets}(t_1, t_2) \Leftrightarrow \exists p_1,p,p_2 \in R(&t_1 = (p_1, p) \wedge t_2 = [p, p_2) \\ \vee\ &t_1 = [p_1, p) \wedge t_2 = [p, p_2)) \vee t_1 = (p_1, p) \wedge t_2 = [p, p_2] \\ \vee\ &t_1 = [p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2) \\ \vee\ &t_1 = [p_1, p] \wedge t_2 = (p, p_2) \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2] \\ \vee\ &t_1 = [p_1, p] \wedge t_2 = (p, p_2])\end{aligned}$$

It is easy to see that the intuitive meaning of Meets(t_1, t_2) is that, on the one hand, time-elements t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point); on the other hand, there is not any other time-element standing between them.

Analogous to the 13 relations introduced by Allen for intervals [13], there are 30 exclusive temporal order relations over time-elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations that relate points to points:
  {Equal, Before, After}
- Relations that relate points to intervals:
  {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations that relate intervals to points:
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations that relate intervals to intervals:
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

The definition of these derived temporal order relations in terms of the single relation Meets is straightforward. E.g.:

$$\text{Before}(t_1, t_2) \Leftrightarrow \exists t \in T(\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$$

Based on such a time theory, a time-series T_n can be defined as a vector of time-elements temporally ordered one after another [14]. Formally, a general time-series is defined in terms of the following schema:

GTS1) $T_n = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

GTS2) $\text{Meets}(t_i, t_{i+1}) \vee \text{Before}(t_i, t_{i+1})$, for all i = 1, …, n-1

GTS3) $T_{dur}(t_i) = q_i - p_i$, for some i where $1 \leq i \leq n$.

GTS4) $T_{gap}(t_i, t_{i+1}) = p_{i+1} - q_i$ for some i where $1 \leq i \leq n-1$.

Generally speaking, a time-series may be incomplete in various ways. For example, if the relation between t_j and t_{j+1} is "Before" rather than "Meets", it means that the knowledge about the time-element(s) between t_j and t_{j+1} is not available. In addition, if $T_{dur}(t_k)$ is missing for some k, it means that duration knowledge as for time-element t_k is unknown. Correspondingly, a complete time-series is defined in terms of the schema as below:

CTS1) $T_n = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

CTS2) $Meets(t_i, t_{i+1})$, for all i = 1, …, n-1.

CTS3) $T_{dur}(t_i) = q_i - p_i$, for all i = 1, …, n.

CTS4) $T_{gap}(t_i, t_{i+1}) = 0$ for all i = 1, …, n-1.

The validation of data is usually dependent on time. For instance, $1000 (account balance) can be valid before and on 1 January 2003 but become invalid afterwards. We shall use fluents to represent Boolean-valued, time-varying data, and denote proposition "fluent f holds true over time t" by formula Holds(f, t) [13]:

(F1) $(f, t) \Rightarrow \forall t_1(Part(t_1, t) \Rightarrow Holds(f, t_1))$
That is, if fluent f holds true over a time-element t, then f holds true over any part of t.

(F2) $\forall t_1(Part(t_1, t) \Rightarrow \exists t_2(Part(t_2, t_1) \wedge Holds(f, t_2))) \Rightarrow Holds(f, t)$
That is, if any part of time t contains a part of itself over which fluent f holds true, then f holds true over t. Here,

$Part(t_1, t) \Leftrightarrow Equal(t_1, t) \vee Starts(t_1, t) \vee During(t_1, t) \vee Finishes(t_1, t)$

(F3) $Holds(f_1, t) \vee Holds(f_2, t) \Rightarrow Holds(f_1 \vee f_2, t)$
That is, is fluent f_1 or fluent f_2 holds true over time t, then at least one of them holds true over time t.

(F4) $Holds(not(f), t) \Leftrightarrow \forall t_1(Part(t_1, t) \Rightarrow \neg Holds(f, t_1))$
That is, the negation of fluent f holds true over time t if and only if fluent f does not hold true over any part of t.

(F5) $Holds(f, t_1) \wedge Holds(f, t_2) \wedge Meets(t_1, t_2) \Rightarrow Holds(f, t_1 \oplus t_2)$
That is, if fluent f holds true over two time-elements t_1 and t_2 that meets each other, then f holds over the ordered-union of t_1 and t_2.

A state is defined as a collection of fluents. Following the approach proposed in [14], we shall use Belongs(f, s) to denote that fluent f belongs to the collection of fluent representing state s. For the reason of simple expression, if $f_1, \ldots, f_m$ are all the fluents that belong to state s, we shall represent s as $<f_1, \ldots, f_m>$. Also, without confusion, we shall use formula Holds(s, t) to denote that s is the state of the world with respect to time t, provided that:

(F6) $s = <f_1, \ldots, f_m> \Rightarrow (Holds(s, t) \Leftrightarrow Holds(f_1, t) \wedge \ldots \wedge Holds(f_m, t))$
That is, a state s holds true over time t if and only if every fluent in the s holds true over time t.

Consequently, a state-sequence S is defined as a list of states together with its corresponding time-series T_n. A general state-sequence is defined in terms of the schema as below:

GSS1) $S_n = [s_1, \ldots, s_n]$

GSS2) $Holds(s_i, t_i)$, for all i = 1, …, n
where $[t_1, \ldots, t_n]$ is a time-series.

Correspondingly, a state-sequence is defined as complete if and only if the corresponding time-series is complete [15].

According to the basic set of axioms with respect to the point & interval based time-series theory [12], for any two adjacent time elements t_1 and t_2 such that $Meets(t_1, t_2)$, we can denote the ordered union of t_1 and t_2 as $t_1 \oplus t_2$ If $Holds(s, t_1)$, $Holds(s, t_2)$, we have:

$$Holds(s, t_1 \oplus t_2)$$
$$T_{dur}(t_1 \oplus t_2) = T_{dur}(t_1) + T_{dur}(t_2)$$

That is, the "ordered union" operation over time elements is consistent with the conventional "addition" operation over the duration assignment function, i.e., 'T_{dur}'.

4. The optimal temporal common subsequence

For two given state-sequences $S_m = [s_1, ..., s_m]$ and $S'_n = [s'_1, ..., s'_n]$, Holds(s_i, t_i) and Holds(s'_j, t'_j), $t_i =< p_i, q_i >$ and $t'_j =< p'_j, q'_j >$ for i = 1, …, m and j = 1, …, n, the algorithm of the optimal temporal common subsequence can be illustrated as below: Firstly, the following algorithm calculates the longest temporal common subsequence.

Input: two state-sequences S_m and S'_n.

Output: the length of the longest temporal common subsequences $OTCS_L(S_m, S'_n)$.

1) Initiation: $s_0 = s'_0 = null$

for i = 0 : m: $OTCS_L(i, 0) = 0$

for j = 0 : n: $OTCS_L(0, j) = 0$

2) Recursion:

for i = 1: m

for j = 1 : n

if $s_i = s'_j$ # matched

case 1: $s_{i-1} = s'_{j-1} = s_i = s'_j$

$OTCS_L(i, j) = OTCS_L(i - 1, j - 1)$

case 2: $s_{i-1} = s'_{j-1} \neq s_i = s'_j$

$OTCS_L(i, j) = OTCS_L(i - 1, j - 1) + 1$

case 3: $(s_{i-1} \neq s'_{j-1}) \& (either\ s_{i-1}\ or\ s'_{j-1} = s_i = s'_j)$

$OTCS_L(i, j) = \max(OTCS_L(i-1, j), OTCS_L(i, j-1))$

case 4: $(s_{i-1} \neq s'_{j-1}) \& (neither\ s_{i-1}\ nor\ s'_{j-1} = s_i = s'_j)$

$OTCS_L(i, j) = OTCS_L(i - 1, j - 1) + 1$

else # $s_i \neq s'_j$, unmatched

$OTCS_L(i, j) = \max(OTCS_L(i-1, j), OTCS_L(i, j-1))$

end # *end of if*

end # end of j

end # end of i

3) Accomplishment

$OTCS_L(S_m, S'_n) = OTCS_L(m, n)$

In above algorithm, the continuously duplicated states are not re-counted as new common states in any state-sequence. Secondly, in the same manner, we simultaneously record $Ind_k = (f_k, l_k)$ and $Ind'_k = (f'_k, l'_k)$ as the first and the last index of the kth common state between S_m and S'_n, where k = 1, …, L = $OTCS_L(S_m, S'_n)$, $f_k, l_k \in$ [1, m] and $f'_k, l'_k \in$ [1, n]. According to the typed point-based intervals, the temporal duration difference $OTCS_D(S_m, S'_n)$ and temporal gap difference $OTCS_G(S_m, S'_n)$ are calculated as below:

$$OTCS_D(S_m, S'_n) = \sum_{k=1}^{L} \left| (q_{l_k} - p_{f_k}) - (q'_{l'_k} - p'_{f'_k}) \right| \quad (4)$$

$$OTCS_G(S_m, S_n') = \begin{cases} 0 & if\ k = 1 \\ \sum_{k=2}^{L} \left| (p_{f_k} - q_{l_{k-1}}) - (p'_{f_k} - q'_{l_{k-1}}) \right| & else \end{cases} \quad (5)$$

Finally, the overall similarity with respect to the temporal order, temporal duration and temporal gap is defined as:

$$\begin{aligned} OTCS(S_m, S_n') &= w_L \bullet OTCS_L(S_m, S_n') \\ &\quad - w_D \bullet OTCS_D(S_m, S_n') - w_G \bullet OTCS_G(S_m, S_n') \end{aligned} \quad (6)$$

Comparing with the conventional LCS based measurements introduced in section I, the main advantage of OTCS is that it does deal with the difference caused by the temporal duration and the temporal gap during state-sequences. For example, given state-sequences C_1 = [abcd], C_2 = [aaaaabc], C_3 = [aabbccdd], C_4 = [aaebbfccgdd] and C_5 = [aaaabbb]. For the reason of simple illustration, the temporal duration of each state is set as 1 and the temporal gap between each pair of adjacent states is set as 0 if they are identical or 1 if they are different. Table 1 reports the similarity between state-sequences measured by different algorithms. For OTCS, the similarity is reported in terms of a triad [$OTCS_L$, $OTCS_D$, $OTCS_G$] which will be integrated with $w_L = 1$ and $w_D = w_G = 0.1$.

Table 1. Similarity Between Example State-sequences With Different Measurements

Similarity		C_1	C_2	C_3	C_4	C_5
LCS	C_1	4	3	4	4	2
	C_2	3	7	4	4	5
	C_3	4	4	8	8	4
	C_4	4	4	8	11	4
	C_5	2	5	4	4	7
CLCS (th=2)	C_1	4	3	0	0	2
	C_2	3	7	3	0	5
	C_3	0	3	8	0	4
	C_4	0	0	0	11	0
	C_5	2	5	8	0	7
ACS	C_1	16	8	16	16	4
	C_2	8	128	16	16	32
	C_3	16	16	256	256	16
	C_4	16	16	256	2048	16
	C_5	4	32	16	16	128
T-WLCS	C_1	4	7	8	8	7
	C_2	7	11	10	10	11
	C_3	8	10	12	12	9
	C_4	8	10	12	15	9
	C_5	7	11	9	9	12
OTCS	C_1	[4, 0, 0]	[3, 4, 0]	[4, 4, 0]	[4, 4, 6]	[2, 5, 0]
	C_2	[3, 4, 0]	[3, 0, 0]	[3, 5, 0]	[3, 5, 4]	[2, 3, 0]
	C_3	[4, 4, 0]	[3, 5, 0]	[4, 0, 0]	[4, 0, 6]	[2, 3, 0]
	C_4	[4, 4, 6]	[3, 5, 4]	[4, 0, 6]	[7, 0, 0]	[2, 3, 2]
	C_5	[2, 5, 0]	[2, 3, 0]	[2, 3, 0]	[2, 3, 2]	[2, 0, 0]

The "non-uniqueness" problem (different state-sequences have the same similarity to the query state-sequence) is ubiquitous when applying those conventional algorithms due to the lacking of dealing with temporal duration difference and temporal gap difference. For instance, given three state-sequence pairs (C_1, C_1), (C_1, C_3) and (C_1, C_4) with the same temporal common subsequence 'abcd', we shall get Sim(C_1, C_1) = Sim(C_1, C_3) by using LCS and ACS, which states that the two state-sequences, C_3 and C_1, have the same similarity to C_1, where in fact they have different temporal durations. Also we shall get Sim(C_1, C_3) = Sim(C_1, C_4) by using CLCS, LCS, ACS and T-WLCS, which states C_3 and C_4 have the same similarity to C_1 where in fact they are with different temporal gaps. The proposed OTCS in this paper is the only one that can distinguish the different temporal duration or temporal gap, and in fact we have OTCS(C_1, C_1) > OTCS(C_1, C_3) > OTCS(C_1, C_4).

In addition, some other abnormal or unreasonable results occur in those existing algorithms when continuously duplicated common states exist frequently in state-sequences. For example, following CLCS, LCS, ACS or T-WLCS, one will get $Sim(C_2, C_5) > Sim(C_2, C_3)$. However, according to the definition of temporal common subsequence, the similarity degree between C_3 and C_2 should be in fact higher than that between C_5 and C_2. This is corrected in OTCS by reaching that $OTCS(C_2, C_3) > OTCS(C_2, C_5)$.

Furthermore, in particular, CLCS is very fluctuant since the continuity of matched common subsequences may be destroyed easily by the unmatched states (e.g., resulting as $CLCS(C_4, C_1) = CLCS(C_4, C_2) = CLCS(C_4, C_3) = CLCS(C_4, C_5) = 0$) or by the continuously duplicated common states (e.g., resulting as $CLCS(C_1, C_3) = 0$). In ACS, the similarity becomes extremely large (such as C_3 and C_4) when continuously duplicated common states exist frequently in state-sequences and will therefore underestimate the high similarity between C_3 and C_1. T-WLCS even cannot guarantee the query state-sequence has the highest similarity with itself: for instance, $T\text{-}WLCS(C_1, C_1) < T\text{-}WLCS(C_1, C_2)$. Such a problem becomes absurd if, for instance, we have C_2' = 'aaaaaaaaaaaa', which will lead to $T\text{-}WLCS(C_1, C_2') = 12$ due to the unreasonable treatment to continuously duplicated common states.

5. Experimental Results

To demonstrate the performance of OTCS, we test it on a news video retrieval system. We have collected over 300 news video clips (state-sequences) lasting up to 5 hours as our database. The number of key-frame (state) of each video clip varies from 10 to 65. For each key-frame, we extract the 64-dimensional colour histogram as the feature vector which is then quantized by the paired quantizer introduced above where the similar key-frames will be quantized as the identical state. Several query sets are reconstructed:

Original Query Set (OQS): 60 state-sequences randomly selected from the database;

Shortened Query Set (SQS): each state-sequence of this set is with length of (1-α%)*60 by deleting α%*60 states from OQS randomly;

Lengthened Query Set (LQS): each state-sequence of this set is with length of (1+β%)*60 by duplicating β% predecessors with random position in OQS.

Figure 4. An example of key-frame sequence in video clip database

Figure 4 shows an example of key-frame sequence of video clip with various temporal duration and temporal gap. The similar key frames (key-frame 7 ~11) will be quantized as the identical state, the duration of which is equal to the sum of their duration.

We compare the performance with LCS, CLCS, T-WLCS and ACS. Again for OTCS, the temporal duration of each key-frame is set as 1 and the temporal gap between each pair of adjacent key-frames is set as 0 if they are identical or 1 if they are different. We set $w_L = 1$ and test the experiment with w_D and w_G varying from {1, 1/2, 1/4, …, 1/128} and choose the values leading to the optimal performance.Table 2 shows the retrieval precision on OQS against top number (the number of the most similar video clips compared with the query video clip). Obviously, all similarity measurements perform better with the increase of top number, but generally speaking, OTCS outperforms the others. In following experiments, the top number is fixed to 8 where the precision of these five measurements has the largest standard deviation (std).

Table 2. Retrieval precision on OQS

Top number / Method	2	4	6	8	10	12	14	MEAN
LCS	.72	.73	.76	.80	.86	.94	.98	.83
CLCS	.70	.71	.73	.73	.77	.80	.85	.76
ACS	.78	.80	.84	.90	.93	.95	.99	.88
T-WLCS	.75	.81	.81	.86	.90	.92	.98	.86
OTCS	.84	.85	.92	.93	.96	.98	.99	**.92**
STD	.055	.058	.074	**.080**	.073	.069	.056	

Figure 5 shows the retrieval precision on SQS and LQS. It's clear to see that OTCS is much more robust than the others since by means of adjusting the value of the weight, it can handle temporal duration difference and temporal gap difference caused by deletion and insertion. CLCS is most fluctuant with worst precision especially in LQS since insertion operation may weaken the continuity of common subsequence. LCS is robust (with smallest variance) but not as effective as OTCS. In addition, LCS has less influence on LQS since it can skip the duplicated key-frames. ACS and T-WLCS are sensitive to the insertion and deletion degree as CLCS.

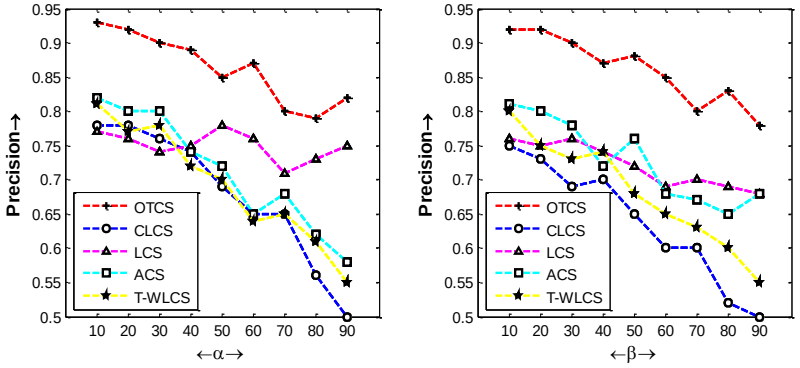
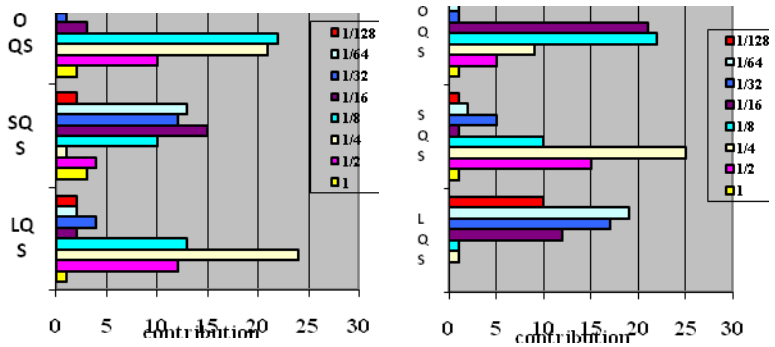


Figure 5. Retrieval Precision on SQS against α and LQS against β

Figure 6 shows the weight contribution of the temporal characters on different query sets. Generally speaking, the length of the longest temporal common subsequence contributes more significance than temporal duration and temporal gap on any query set. As for OQS, the temporal duration plays a slightly more significant role than temporal gap because of the existence of approximate adjacent key-frames which may be quantized as identical key-frames in video clips. For SQS, due to the deletion of some key-frames, the temporal gap plays a more important role than temporal duration while contrarily in LQS since the insertion operation generates more duplications of key-frames.



(a) temporal duration (b) temporal gap

Figure 6. Weights contribution of temporal characters in OTCS

6. Conclusion and Future Work

State-sequence matching is a very hot research topic in data mining [16]. In this paper, we have presented an efficient and effective state-sequence matching algorithm for news video retrieval. The fundamental formal representation of time-series and state-sequence is introduced in detail, based on which, we proposed a new concept of temporal common subsequence different from the traditional common subsequence. A new LCS-based algorithm named Optimal Temporal Common Subsequence (OTCS) which takes into account rich temporal information (including temporal order, temporal duration and temporal gap) between state-sequences is finally designed and tested on news video retrieval. The experimental results demonstrate the effectiveness and robustness of the new algorithm.

Linear combination is the most direct method to combine the three temporal characters. However, it will be sensitive to the weight selection. Also, redundant calculation for the other two temporal characters seems to be able to be optimized, which will remain as our future work.

7. References

H. Wu, B. Salzberg, and D. Zhang, “Online Event-Driven Subsequence Matching over Financial Data Streams”, Proc. Int’l Conf. Management of Data (SIGMOD 04), ACM Press, Jun. pp: 23-34, 2004.

Y. Zhu and D. Shasha, “Warping indexes with envelope transforms for query by humming”, In Proc. Int. Conf. on Management of Data (SIGMOD 03), ACM Press, Jun. pp:181–192, 2003.

H. T. Shen, J. Shao, Z. Huang and X. Zhou, “Effective and Efficient Query Processing for Video Subsequence Identification”. IEEE Transactions on Knowledge and Data Engineering, 21(3), pp:321-334, 2009.

R. Agrawal, C. Faloutsos and A. Swami, “Efficient similarity search in sequence databases”. In Proc. of the 4th Int'l Conf. on Foundations of Data Organization and Algorithms (FODO’93), Springer Press, Oct. pp:69-84, 1993.

N. Beckmann, H. Kriegel, R. Schneider and B. Seeger. “The r*-tree: An efficient and robust access method for points and rectangles”. In Proc. of the Int'l Conf. on Management of Data (SIGMOD 99), ACM Press,, May. pp:322-331, 1990.

Y. Moon, K. Whang and W. Loh, “Duality-based subsequence matching in time-series databases”. In Proc. of the 17th Int'l Conf. on Data Engineering (ICDE’01, May. pp: 263-272, 2001.

L. Bergroth and H. Hakonen and T. Raita. "A Survey of Longest Common Subsequence Algorithms". Proc. Seventh International Symposium on String Processing and Information Retrieval (SPIRE’00), IEEE Press, pp:39–48.

A. Gao, H. Siegelmann, “Time-Warped Longest Common Subsequence Algorithm for Music Retrieval”. 5th International Conference on Music Information Retrieval (ISMIR’04),Oct.2004.

E. Keogh, “Exact indexing of dynamic time warping”, Proc. of the 28th Int’l Conf. on Very Large Data Bases (VLDB’02), VLDB Endowment, Aug. pp:406–417, 2002.

APPENDIX E EFFICIENT AND EFFECTIVE STATE-BASED FRAMEWORK FOR NEWS VIDEO RETRIVAL

Y. Kim and T. Chua, "Retrieval of News Video Using Video Sequence Matching". Proc of the 11th Int. Multimedia Modelling Conference (MMM'05), IEEE Press, Jan. pp: 68 – 75, 2005.

H. Wang, "All Common Subsequences", Proc. 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07), Jan. pp:635-640, 2007.

J. Ma & P. Hayes, "Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?", the Computer Journal, 49(1), pp:32-41, 2006.

J. Allen. "Towards a General Theory of Action and Time", Artificial Intelligence, 23, pp:123-154, 1984.

M. Shanahan. "A Circumscriptive Calculus of Events", Artificial Intelligence, 77, pp:29-384, 1995.

J. Ma, R. Bie, G. Zhao, "An ontological Characterization of Time-series and State-sequences for Data Mining", Proc. of the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'08), IEEE Press, Oct. pp:325-329, 2008.

Y. Peng, G. Kou, Y. Shi, and Z. Chen, "A Descriptive Framework for the Field of Data Mining and Knowledge Discovery", International Journal of Information Technology and Decision Making, 7 (4), pp639 – 682, 2008.

Order, Duration and Gap —Take Them All

Aihua Zheng, Jixin Ma, Miltos Petridis
School of Computing and Mathematical Sciences
The University of Greenwich, London, UK
{a.zheng, j.ma, m.petridis}@gre.ac.uk

Bai Xiao
Department of Computer Science
Beihang University, Beijing, China
little_point_baixiao@hotmail.com

***Abstract*—Based on a formal characterization of time-series and state-sequences, a new distance measurement dealing with both non-temporal and temporal distances for state-sequence matching is proposed in this paper. In addition to formulating the temporal order over state-sequences, it also takes into account of temporal distances in terms of both the temporal duration of each state and the temporal gaps between adjacent pairs of states, which are neglected in most existing approaches to time-series and state-sequence matching. In particular, when specialized as a real-penalty-style measurement by means of reifying the cost functions, it is more flexible with regards to real-life applications than binary-value-style distance measurements. In addition, it is more robust than those existing real-penalty-style distance measurements since it can filter out noise during the matching procedure. Experimental results on reconstructed time-series data from UCI KDD Archive demonstrate that it can tackle the most general problems in matching time-series data with rich temporal information.**

Keywords - Pattern Recongition; Time-series; State-sequence Matching

I. INTRODUCTION

Temporal pattern recognition of time-series and state-sequences (also known as state-sequence matching) plays a very important role in data mining and has been well applied in various areas such as financial data analysis, audio recognition, visual information retrieval, etc. One of the most active and essential research topics in state-sequence matching is the distance (or similarity) measurement. On one hand, for general treatment, a versatile distance measurement should be able to deal with both of the non-temporal and temporal distances for any two given state-sequences, where

1) *Non-temporal distance:* denotes the difference between those states appearing in that two given state-sequences, ignoring any temporal issues.
2) *Temporal distance:* consists of 3 characters:
   a) *Temporal Order:* the temporal relation over the states to be matched in the two given state-sequences.
   b) *Temporal Duration*: the duration of each state, e.g., T_{dur} as shown in Fig. 1.
   c) *Temporal Gap:* the time interval standing between two adjacent states, e.g., T_{gap} as shown in Fig. 1.

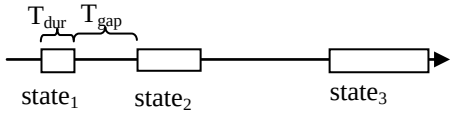


Figure 1 Temporal Gap and Temporal Duration

Various distance measurements have been developed over the past half century, for state-sequence matching, including Lp-Norms [5], the Longest Common Subsequence (LCSS) [11], Dynamic Time Warping (DTW) [6], and Edit Distance [7] and its variants such as Edit Distance on Real Sequence (EDR) [3], Edit Distance with Real Penalty (ERP) [4] and Time Warp Edit Distance (TWED) [10], etc. However, most of these existing distance measurements characterize temporal distance in terms of only the temporal order over the state-sequences, where other important temporal features such as temporal duration of each state itself and temporal gap between two adjacent states have been neglected. The only noted exception is TWED which addresses temporal gap difference in term of the temporal index of states while temporal duration of states is not dealt with at all. In addition, in all the existing distance measurements, time-series and state-sequences are simply expressed as lists (timestamps) in the form of $t_1, t_2, \ldots, t_n$ (or $s_1, s_2, \ldots, s_n$), where the fundamental time theories based on which time-series and sequences are formed up are usually not explicitly specified. Therefore, the formal characterizations with respect to the temporal basis are neglected, leaving some critical issues unaddressed. E.g.:

- What a sort of objects do these $t_1, t_2, \ldots$ and t_n belong to? In other word, are they time points, time intervals, or simply some absolute values from the real numbers, integers, or the clock?
- What are the temporal order relationships between these $t_1, t_2, \ldots$ and t_n, and/or between the sequence of collections? Are they simply well-ordered as the natural numbers, or they may be relatively ordered by means of relations such as "Before", "Meets", "During", and so on?

This research is supported in part by National 973 Project (No. 2010CB327902)

- What are the associations between time-series/ sequences and non-temporal data that represent various states of the world in discourse?

On the other hand, distance measurements can be classified into two categories, with respect the ways in which the cost function is reified: (a) binary-value-style distance measurements, where the cost functions take binary value (0/1) as matching cost which is not sensitive to noise since they treat the noise and unmatched states with the same cost (1); (b) real-penalty-style distance measurements, in which the cost functions take real difference as matching cost. In general, the real-penalty-style distance measurements outperform the binary-value-style distance measurements. However, the real-penalty-style distance measurements are much more sensitive to noise since the real difference between noise and non-noise states may lead the overall distance to an abnormal degree.

The objective of this paper is to propose a new distance measurement (NDM) which tackles both non-temporal distance and temporal distance including all the three temporal characters as described above, as well as the disturbance of noise. The rest of this paper is organised as below: the formal characterization of time-series and state-sequences is introduced in section II, where our new distance measurement is presented in section III. The generality of the NDM is demonstrated in section IV by means of showing other existing distance measurements as special cases. Section V addresses the reification of the cost functions with respect to the 3 temporal characters. Experiments on reconstructed time-series data from UCI KDD Archive are conducted and the corresponding results are analysed in section VI. Section VII provides a brief summary and concludes the paper with the prospects for future work.

II. FORMAL CHARACTERIZATION OF TIME-SERIES AND STATE-SEQUENCES

In this section, we present the formal characterization of time-series and state-sequences. For the sake of allowing expression of both absolute time values and relative temporal relations, in this paper, time-elements are defined as typed point-based intervals, each of which must be in one of the following four forms [9]:

$$(p, q) = \{r \mid r \in R \wedge p < r < q\}$$
$$[p, q) = \{r \mid r \in R \wedge p \leq r < q\}$$
$$(p, q] = \{r \mid r \in R \wedge p < r \leq q\}$$
$$[p, q] = \{r \mid r \in R \wedge p \leq r \leq q\}$$

In the above, R stands the set of real numbers, and real numbers p and q are called the left-bound and right-bound of time-element t, respectively. The absolute values as for the left and/or right bounds of some time-elements might be unknown. In this case, real number variables are used for expressing relative relations to other time-elements (see later). If p = q, t is called a time point; otherwise it is called a time interval. Without confusion, time-element [p, p] is taken as identical to point p. Also, if a time-element is not specified as open or closed at its left (right) bound, we shall use "<" (or ">") as for its left (or right) bracket. In addition, the temporal duration of a time-element t, $T_{dur}(t)$, and the temporal gap between adjacent elements t_1, t_2, $T_{gap}(t_1, t_2)$ can be defined as below:

$$t = \langle p, q\rangle \Leftrightarrow T_{dur}(t) = q - p$$
$$t_1 = \langle p_1, q_1\rangle, t_2 = \langle p_2, q_2\rangle \Leftrightarrow T_{gap}(t_1, t_2) = |p_2 - q_1|$$

Following Allen's terminology [1], we shall use "Meets" to denote the immediate predecessor order relation over time-elements, which can be formally defined as:

$$\begin{aligned}&Meets(t_1, t_2) \Leftrightarrow \exists p,r,q \in R(t_1 = (p, r) \wedge t_2 = [r, q)\\ &\vee t_1 = [p, r) \wedge t_2 = [r, q)) \vee t_1 = (p, r) \wedge t_2 = [r, q]\\ &\vee t_1 = [p, r) \wedge t_2 = [r, q] \vee t_1 = (p, r] \wedge t_2 = (r, q)\\ &\vee t_1 = [p, r] \wedge t_2 = (r, q) \vee t_1 = (p, r] \wedge t_2 = (r, q]\\ &\vee t_1 = [p, r] \wedge t_2 = (r, q])\end{aligned}$$

It is easy to see that the intuitive meaning of $Meets(t_1, t_2)$ is that, on the one hand, time-elements t_1 and t_2 don't overlap each other (i.e., they don't have any part in common, not even a point); on the other hand, there is not any other time-element standing between them.

Analogous to the 13 relations introduced by Allen for intervals [1], there are 30 exclusive temporal order relations over time-elements including both time points and time intervals, which can be classified into the following 4 groups:

- Relations that relate points to points:
  {Equal, Before, After}
- Relations that relate points to intervals:
  {Before, After, Meets, Met_by, Starts, During. Finishes}
- Relations that relate intervals to points:
  {Before, After, Meets, Met_by, Started_by, Contains, Finished_by}
- Relations that relate intervals to intervals:
  {Equal, Before, After, Meets, Met_by, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by}

The definition of these derived temporal order relations in terms of the single relation Meets is straightforward. E.g.:

$$Before(t_1, t_2) \Leftrightarrow \exists t \in T(Meets(t_1, t) \wedge Meets(t, t_2))$$

Based on such a time theory, a time-series T_n can be defined as a vector of time-elements temporally ordered one after another [8]. Formally, a general time-series is defined in terms of the following schema:

GTS1) $T_n = [t_1, \ldots, t_n] = [\langle p_1, q_1\rangle, \ldots, \langle p_n, q_n\rangle]$

GTS2) $Meets(t_i, t_{i+1}) \vee Before(t_i, t_{i+1})$, for all $i = 1, \ldots, n-1$

GTS3) $T_{dur}(t_i) = q_i - p_i = d_i$, for some i where $1 \leq i \leq n$.

GTS4) $T_{gap}(t_i, t_{i+1}) = p_{i+1} - q_i = g_i$ for some i where $1 \leq i \leq n$, and g_0 is initialized as 0.

Generally speaking, a time-series may be incomplete in various ways. For example, if the relation between t_i and t_{i+1} is "Before" rather than "Meets", it means that the knowledge about the time-element(s) between t_i and t_{i+1} is not available. In addition, if $T_{dur}(t_i) = d_i$ is missing for some i, it means that duration knowledge as for time-element t_i is unknown.

Correspondingly, a complete time-series is defined in terms of the schema as below:

GTS1) $T_n = [t_1, \ldots, t_n] = [<p_1, q_1>, \ldots, <p_n, q_n>]$

GTS2) $\text{Meets}(t_i, t_{i+1})$, for all i = 1, …, n-1.

GTS3) $T_{dur}(t_i) = q_i - p_i = d_i$, for i = 1, …, n.

GTS4) $T_{gap}(t_i, t_{i+1}) = p_{i+1} - q_i = g_i$ for i = 1, …, n-1; and $g_0 = 0$.

The validation of data is usually dependent on time. For instance, $1000 (account balance) can be valid before and on 1 January 2003 but become invalid afterwards. We shall use fluents to represent Boolean-valued, time-varying data, and denote proposition "fluent f holds true over time t" by formula Holds(f, t) (see details in [1]).

Consequently, a state-sequence S_n is defined as a list of states together with its corresponding time-series T_n. A general state-sequence is defined in terms of the schema as below:

GTS1) $S_n = [s_1, \ldots, s_n]$

GTS2) $\text{Holds}(s_i, t_i)$, for all i = 1, …, n

where $[t_1, \ldots, t_n] = T_n$ is a time-series.

Correspondingly, a state-sequence is defined as complete if and only if the corresponding time-series is complete [8].

III. NEW DISTANCE MEASUREMENT FOR FORMAL STATE-SEQUENCE MATCHING

Based on the above characterization of time-series, the triple domain $U = S \times D \times G$ is defined for state-sequences where:

$S \subset R^d$: *d*-dimensional domain of non-temporal data ordered in consequential (that is, "Meets or Before") temporal order;

D, G ⊂ R: the domains of temporal duration and temporal gap respectively.

So the formal characterization of two given state-sequences can be expressed as $A_m = [a_1, \ldots, a_m]$ and $B_n = [b_1, \ldots, b_n] \in U$ where

for $i = 1, \ldots, m$, $j = 1, \ldots, n$:

$a_i = <s_i', d_i', g_i'> \in S \times D \times G$ and $b_j = <s_j'', d_j'', g_j''> \in S \times D \times G$;

$s_i' = [s_{i1}', \ldots, s_{id}']$ and $s_j'' = [s_{j1}'', \ldots, s_{jd}'']$

$\text{Holds}(s_i', t_i')$ and $\text{Holds}(s_j'', t_j'')$

$t_i' = <p_i', q_i'>$ and $t_j'' = <p_j'', q_j''>$;

$d_i' = T_{dur}(t_i') = q_i' - p_i'$ and $d_j'' = T_{dur}(t_j'') = q_j'' - p_j''$;

for $i = 1, \ldots, m-1$, $j = 1, \ldots, n-1$:

$g_i' = T_{gap}(t_i', t_{i+1}') = p_{i+1}' - q_i'$ and $g_j'' = T_{gap}(t_j'', t_{j+1}'') = p_{j+1}'' - q_j''$

and $g_0' = g_0'' = 0$.

With respect to the non-temporal information and rich temporal information for these two state-sequences, the general distance measurement is defined as:

$$NDM(A_m, B_n) = w_{ntem} Dis_{ntem}(A_m, B_n) + w_{tem} Dis_{tem}(A_m, B_n) \quad (1)$$

where $Dis_{ntem}(A_m, B_n)$ and $Dis_{tem}(A_m, B_n)$ denote the non-temporal distance and temporal distance, respectively with the corresponding weight w_{ntem} and w_{tem}.

A. Non-temporal Distance

Non-temporal matching stands for the elemental state matching of the state-sequences A_m and B_n, due to the fact that elemental state appearing in state-sequences are not actually ordered by their index, that is, the state-sequence is actually regarded as a set of states. Therefore, in the first place, pairing two given state-sequences involves a combinational permutation problem. In general, for m≥n, there are ${}^m\text{Pr}_n = \text{m!n!/(m-n)!}$ ways of pairing A_m with B_n. Let Pr denote the set of all possible ordered vectors formed by selecting, in order, n random elemental states from A_m. It seems reasonable to take the pairing which gives the minimal overall distance. Hence, in this paper, we shall define the non-temporal distance between A_m and B_n as:

$$Dis_{ntem}(A_m, B_n) = min_{pr \in Pr} Dis_{ntem}(pr, B_n) \quad (2)$$

Where $dis_{ntem}(pr, B_n) = \sqrt{\sum_{j=1}^{n} w_{ipr} dis_{Lp}(pr_j, s_j'')^2} \Big/ \sqrt{\sum_{i=1}^{n} w_{ipr}}$

$pr = [pr_1, \ldots, pr_n]$

B. Temporal Distance

Based on the triad representation of state-sequences, the temporal distance between two given state-sequences A_m with B_n with respect to the 3 temporal characters, that is, temporal order, temporal gap and temporal duration, is defined recursively as below:

$$Dis_{tem}(A_m, B_n) = \min \begin{cases} Dis_{tem}(A_{m-1}, B_n) + W_{del} \bullet Cost(a_m \to \phi) \\ Dis_{tem}(A_m, B_{n-1}) + W_{ins} \bullet Cost(\phi \to b_n) \\ Dis_{tem}(A_{m-1}, B_{n-1}) + W_{sub} \bullet Cost(a_m \to b_n) \end{cases} \quad (3)$$

where m, n ≥ 1, $Cost(a_m \to \phi)$, $Cost(\phi \to b_n)$ and $Cost(a_m \to b_n)$ denote the cost function for edit operations *deletion*, *insertion* and *substitution*, respectively, where

$$Cost(x \to y) = \sum w_i \cdot Cost_i(x \to y), i = \{Tord, Tdur, Tgap\} \quad (4)$$

and $(x \to y) \in \{(a_m \to \phi), (\phi \to b_n), (a_m \to b_n)\}$

The initialization is set as below:

$$Dis_{tem}(A_0, B_0) = 0,$$
$$Dis_{tem}(A_0, B_j) = \infty, \text{ for } j \geq 1 \quad (5)$$
$$Dis_{tem}(A_i, B_0) = \infty, \text{ for } i \geq 1$$

IV. THE GENERALITY OF NDM

NDM proposed here addresses all the 3 temporal characters, including temporal order, duration and gap. In fact, as illustrated in Table I, most of those existing measurements can be taken as special cases of NDM by means of specifying the non-temporal and temporal weights, and the cost functions, correspondingly. N.B. For LCSS, instead of taking the minimum value, the maximum value is

accumulated since it counts the number of common states of two state-sequences instead of the cost of matching them.

TABLE I. MEASUREMENTS SUBSUMED FROM NMD

Measurement	Settings
ED	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=w_{Tgap}=0$ $Cost_{Tord}(a_m,\phi)=Cost_{Tord}(\phi,b_n)=1,$ $Cost_{Tord}(a_m,b_n)=Cost_{Tord}^{ED}(a_m,b_n)$
EDR	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=w_{Tgap}=0$ $Cost_{Tord}(a_m,\phi)=Cost_{Tord}(\phi,b_n)=1,$ $Cost_{Tord}(a_m,b_n)=Cost_{Tord}^{EDR}(a_m,b_n)$
DTW	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=w_{Tgap}=0$ $Cost_{Tord}(a_m,\phi)=Cost_{Tord}(\phi,b_n)=Cost_{Tord}(a_m,b_n)$ $=Cost_{Tord}^{DTW}(a_m,b_n)$
ERP	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=w_{Tgap}=0$ $Cost_{Tord}(a_m,\phi)=d_{Lp}(a_m,g) \quad , \quad Cost_{Tord}(\phi,b_n)=d_{Lp}(g,b_n) \quad ,$ $Cost_{Tord}(a_m,b_n)=d_{Lp}(a_m,b_n)$
LCSS	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=0, W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=w_{Tgap}=0$ $Cost_{Tord}(a_m,\phi)=Cost_{Tord}(\phi,b_n)=0,$ $Cost_{Tord}(a_m,b_n)=Cost_{Tord}^{LCSS}(a_m,b_n)$
TWED	$w_{ntem}=0, w_{tem}=1 \quad W_{del}=W_{ins}=W_{sub}=1$ $w_{Tord}=1, w_{Tdur}=0, w_{Tgap}=v$ $\begin{cases} Cost(a_m,\phi)=Cost_{Tord}^{TWED}(a_m,a_{m-1})+v\cdot Cost_{Tgap}^{TWED}(a_m,a_{m-1})+\lambda \\ Cost(\phi,b_n)=Cost_{Tord}^{TWED}(b_{n-1},b_n)+v\cdot Cost_{Tgap}^{TWED}(b_{n-1},b_n)+\lambda \\ Cost(a_m,b_n)=Cost_{Tord}^{TWED}(a_m,b_n)+Cost_{Tord}^{TWED}(a_{m-1},b_{n-1})+ \\ \qquad v\cdot(Cost_{Tgap}^{TWED}(a_m,b_n)+Cost_{Tgap}^{TWED}(a_{m-1},b_{n-1})) \end{cases}$

V. COST FUNCTION REIFICATION

With regards to the cost function, distance measurements for state-sequence matching can be grouped into two categories: binary-value-style distance measurements such as LCSS, ED and EDR, and real-penalty-style distance measurements such as ERP, DTW and TWED. As discussed in the introduction, the later outperform the former but are much more sensitive to noise.

To filter out the noise or release its influence, the cost function in NDM is defined as below:

For $i = \{T_{ord}, T_{dur}, T_{gap}\}$

$$Cost(x\to y)=\begin{cases}\sum_i w_i\cdot Cost_i(x\to y) & if\ Cost_i(x\to y)\le\delta\\ c & else\end{cases} \quad (6)$$

where $(x\to y)\in\{(a_m\to\phi),(\phi\to b_n),(a_m\to b_n)\}$ and c is a constant usually set either as 0 (to filter out the noise), or as the current maximum cost (to release the influence of the noise).

The main difference among the three typical real-penalty-style distance measurements ERP, DTW and TWED is: when *insertion* (or *deletion*) is required to align state-sequence A_m and B_n, ERP inserts a constant (usually 0) into A_m while DTW duplicates the previous state in A_m and TWED duplicates the previous state in B_n in terms of the graphical editor paradigm [10]. These different disposals will lead to different costs for operation *insertion*, *deletion* and *substitution*. We shall follow the approach of EDR and use weights W_{del}, W_{ins} and W_{sub} to adjust the corresponding operations. In fact, the cost functions of NDM are defined as below:

$$Cost_{Tord}(a_i\to b_j)=\begin{cases}dist_{Lp}(0,s_j'') & if\ s_i'=\phi\\ dist_{Lp}(s_i',0) & if\ s_j''=\phi\\ dist_{Lp}(s_i',s_j'') & else\end{cases} \quad (7)$$

$$Cost_{Tdur}(a_i\to b_j)=\begin{cases}dist_{Lp}(0,d_j'') & if\ d_i'=0\\ dist_{Lp}(d_i',0) & if\ d_j''=0\\ dist_{Lp}(d_i',d_j'') & else\end{cases} \quad (8)$$

$$Cost_{Tgap}(a_i\to b_j)=\begin{cases}dist_{Lp}(0,g_{j-1}'') & if\ g_{i-1}'=0\\ dist_{Lp}(g_{i-1}',0) & if\ g_{j-1}''=0\\ dist_{Lp}(g_{i-1}',g_{j-1}'') & else\end{cases} \quad (9)$$

Formulae (2), and (6)-(9) accommodate non-temporal and all the 3 temporal distances, as well as the cost function, which illustrates the integrity and generality of NDM.

VI. EXPERIMENTAL RESULTS

A. Experiment set up

The NDM has been tested on Synthetic Control Chart Time Series in UCI KDD Archive. The database consists of 600 state-sequences with length of 60 for each, including 6 different classes (100 samples each). Several query sets are reconstructed: *Original Query Set* (*OQS*): consists of 180 (the first 30 state-sequences from each class) state-sequences; *Shortened Query Set* (*SQS*): each state-sequence is with length of (1-α)*60 by deleting α*60 states evenly (EV), from the beginning (FB) and from the end (FE) of the corresponding state-sequence in *OQS*; *Lengthened Query Set (LQS)*: each state-sequence is with length of (1+β)*60 by inserting β*60 states evenly (EV), from the beginning (FB) and from the end (FE) into the corresponding state-sequence in *OQS*; *Noised Query Set* (*NQS*): each state-sequence is obtained by adding a Gaussian noise to each state-sequence in *OQS*. We shall simply take $w_{ntem} = w_{tem} = 1$ in the following experiments.

B. Comparison with Binary-value-style Measurements

As a real-penalty-style distance measurement, we firstly compare its performance with binary-value-style

measurements EDR and LCSS with the setting in Table I. The retrieval experiment is implemented with threshold varying from {0.1, 0.2, …, 0.9} for each measurement and the one which leads to the best performance has been chosen as the optimal threshold. Table II shows the targets recall and precision on different query sets defined above, where the average of the results on α, β∈{0.1, 0.2, 0.3, 0.4, 0.5} is calculated. Generally speaking, NDM is more robust and outperforms EDR and LCSS, benefited from its real-penalty-style.

TABLE II. RECALL AND PRECISION COMPARISON OF EDR, LCSS AND NDM WITH DIFFERENT QUERY DATA SET

Data / Measurement		*OQS*	*SQS*			*LQS*		
			EV	FB	FE	EV	FB	FE
Re-call	**EDR**	0.76	0.52	0.64	0.62	**0.76**	0.68	0.76
	LCSS	0.84	0.68	0.8	**0.74**	0.60	0.74	0.72
	NDM	**0.90**	**0.86**	**0.80**	**0.74**	0.74	**0.83**	**0.84**
Pre-cision	**EDR**	0.96	0.76	0.76	**0.88**	0.76	**0.88**	0.84
	LCSS	0.95	0.76	0.85	0.64	0.64	**0.88**	**0.98**
	NDM	**0.99**	**0.83**	**0.88**	0.85	**0.85**	**0.88**	0.97

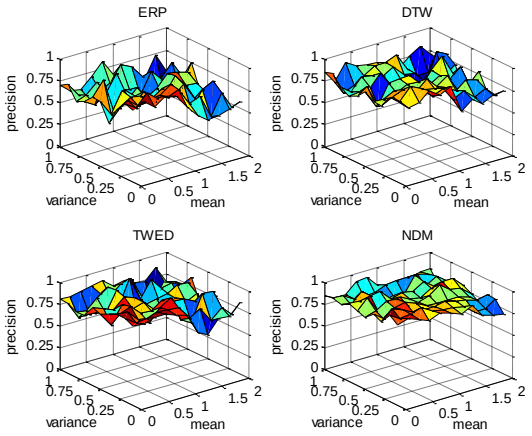


Figure 2 Precision of *NQS* against mean and variance

TABLE III. STATISTIC OF THE PRECISION OF *NQS*

Method / Statistic	ERP	DTW	TWED	NDM
Mean (%)	64.98	75.72	78.80	**85.59**
STD	0.1172	0.0841	0.0971	**0.0583**

C. Comparison with Real-penalty-style Measurements

Comparing with real-penalty-style measurements such as ERP, DTW, TWED, the main advantage of NDM is that it's not sensitive to noise. Fig. 2 shows the results on *NQS* with Gaussian noise in different mean ([0, 0.2,…, 2]) and variance ([0.1, 0.2,…, 1]). The best results for w_{Tgap} = {10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}} are selected. Visually, NDM has higher precision and smaller fluctuation. Table III above shows the average mean and standard deviation (STD) of each subfigure in Fig. 2, which statistically lists the digital values for the corresponding subfigures in Fig. 2.

D. Capability to Handle Temporal Difference

We construct different temporal duration and gap distribution for each class. Fig. 3 shows the examples of 6 different distributions of duration and corresponding temporal gap. We set w_{Tord} = 1 and select the best result for w_{Tdur} and w_{Tgap} from {10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}}. Table IV shows the classification error number for each class with different combinations of distance characters. From which we can see the NDM can tackle most matching tasks involving in time-series and state-sequence data, especially with different temporal matching requirements.

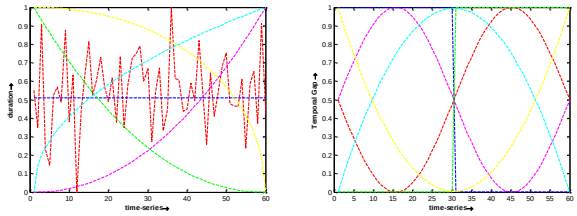


Figure 3 Examples of temporal duration and temporal gap distribution

TABLE IV. CLASSIFICATION ERROR NUMBER OF EACH CLASS WITH DIFFERENT COMBINATIONS OF DISTANCE CHARACTERS

Data / Characters	*OQS*	*SQS*			*LQS*		
		EV	FB	FE	EV	FB	FE
$\mathbf{T_{ord}+T_{gap}}$	1	3	5	4	3	3	4
$\mathbf{T_{ord}+T_{dur}}$	2	4	6	3	4	3	2
$\mathbf{T_{ord}+T_{gap}+T_{dur}}$	1	4	5	5	3	4	3

VII. CONCLUSION AND FUTURE WORK

In this paper, a new distance measurement (NDM), which takes into account of both non-temporal and temporal characters, has been introduced for subsequence matching. Benefiting from a formal characterization of time-series and state-sequences, this measurement is able to deal with temporal order, temporal duration and temporal gap. In particular, when it is specialised as a real-penalty-style distance measurement, it can deduce the influence of noise by means of using inequality filter to filter out the noise.

In order to be applied on large scale database, it's very important to adopt proper pruning strategies, which remain the future work to be conducted.

VIII. REFERENCES

Allen, J. 1984. Towards a General Theory of Action and Time. Artificial Intelligence 23: 123-154.

Bergroth, L., Hakonen, H. and Raita, T. 2000. A Survey of Longest Common Subsequence Algorithms. In Proceedings of the Seventh

International Symposium on String Processing and Information Retrieval, A Curuna, Spain, 39–48.

Chen, L. and Ng, R. 2004. On the Marriage of LP-Norm and Edit Distance. In Proceedings of International Conference on Very Large Data Bases, Toronto, Canada 792-803.

Chen, L., Ozsu, M.T. and Oria, V. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In Proceedings of ACM SIGMOD International Conference Management of Data, Baltimore, Maryland, 491-502.

Faloutsos, C., Ranganathan, M. and Manolopoulos, Y. 1994. Fast subsequence matching in time-series databases. In Proceedings of ACM SIGMOD International Conference Management of Data, Minnesota, United States, 419–429.

Keogh, E. and Pazzani, M. 2000. Scaling up dynamic time warping for data mining applications. In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**,** Massachusetts, United States, 285-289.

Levenshtein, V.I. 1965. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady 10(8): 845-848.

Ma, J., Bie, R. and Zhao, G. 2008. An ontological Characterization of Time-series and State-sequences for Data Mining. In Proceedings of the fifth International Conference on Fuzzy Systems and Knowledge Discovery, Shandong, China, 325-329.

Ma, J. and Hayes, P. 2006. Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?. the Computer Journal 49(1): 32-41.

Marteau, P.F. 2008. Time Warp Edit Distances with Stiffness Adjustment for Time Series Matching. IEEE Transaction on Pattern Analysis and Machine Intelligence. In Press(0):1-15.

Vlachos M, Gunopulos D & Kollios G. 2002. "Discovering similar multidimensional trajectories". In Proceedings of the 18th International Conference on Data Engineering. San Jose, CA, 673–68