The Optimal Temporal Common Subsequence

Aihua Zheng, Xiaoyi Zhou, Jixin Ma, Miltos Petridis
School of Computing and Mathematical Sciences
The University of Greenwich, London, UK
{a.zheng, x.zhou, j.ma, m.petridis }@gre.ac.uk

Abstract— **Based on a formal characterization of time-series and state-sequences, this paper proposes a new algorithm named the Optimal Temporal Common Subsequence (OTCS) to measure the similarity between state-sequences. Distinguishing from the conventional Longest Common Subsequence based measurements, a new concept of common subsequence named 'temporal common subsequence' is proposed to describe the similarity of the temporal order over state-sequences, as well as the similarity of the other two essential and vital temporal characters, i.e., the temporal duration of each state and the temporal gaps between each pair of adjacent states. The experimental results on news video retrieval demonstrate the effectiveness and validity of OTCS.**

Keywords- state-sequence matching; optimal temporal common subsequence; news video retrieval

I. INTRODUCTION

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:

This research is supported in part by National Nature Science Foundation of China (No. 60772122) and the Scientific Research Fund of Hainan Provincial Education Department, Grant Number Hjkj2010-10

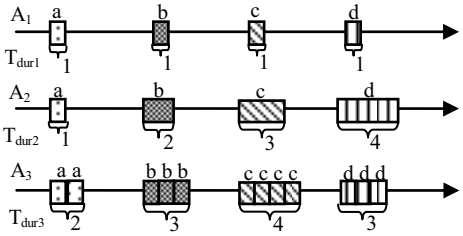


Figure 1. Various temporal duration in state-sequences

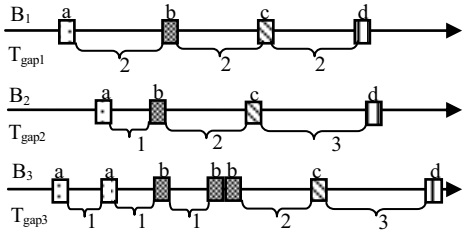


Figure 2. Various temporal gap in state-sequences

- The duration of each state. For instance, as shown in Fig. 1, 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 Fig. 1, 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 Fig. 2, where B_1 and B_2 = 'abcd', B_3 = 'aabbbcd' are with different temporal gap values.

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 A_3 and B_3 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 neighbor(s) (predecessor and successor), that is, there are no continuous duplications of states in temporal common subsequence. For instance, the temporal common subsequence of A_3 and B_3 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 $t_1, t_2, \ldots$ 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?
- 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 propose an Optimal Temporal Common Subsequence (OTCS) algorithm based on a formal characterization of time-series and state-sequences. The rest of this paper is organised as below: the formal characterization of time-series and state-sequences is introduced in section II. Our OTCS is presented and analysed in section III. Experiments on news video retrieval system are conducted and the corresponding results are analysed in section IV to demonstrate the effectiveness and validity of the proposed OTCS. Section V 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 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, $T_{dur}(t)$, and the temporal gap between adjacent elements t_1, t_2, $T_{gap}(t_1, t_2)$ can be defined as below:

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

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

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:

$$\begin{aligned} &Meets(t_1, t_2) \Leftrightarrow \exists p_1, p, p_2 \in R(t_1 = (p_1, p) \wedge t_2 = [p, p_2) \\ &\vee t_1 = [p_1, p) \wedge t_2 = [p, p_2)) \vee t_1 = (p_1, p) \wedge t_2 = [p, p_2] \\ &\vee t_1 = [p_1, p) \wedge t_2 = [p, p_2] \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2) \\ &\vee t_1 = [p_1, p] \wedge t_2 = (p, p_2) \vee t_1 = (p_1, p] \wedge t_2 = (p, p_2] \\ &\quad \vee t_1 = [p_1, p] \wedge t_2 = (p, p_2]) \end{aligned}$$

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], all of which can be derived in terms of the single relation Meets. 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 [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 $1 \le i \le n$.

GTS4) $T_{gap}(t_i, t_{i+1}) = p_{i+1} - q_i$ for some i where $1 \le i \le n-1$.

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) $\text{Meets}(t_i, t_{i+1})$, 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) (see details in [13]).

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) $\text{Holds}(s_i, t_i)$, 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 [14].

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 $\text{Meets}(t_1, t_2)$, we can denote the ordered union of t_1 and t_2 as $t_1 \oplus t_2$ If $\text{Holds}(s, t_1)$, $\text{Holds}(s, t_2)$, 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}'.

III. The Optimal Temporal Common Subseuqence

For two given state-sequences $S_m = [s_1, ..., s_m]$ and $S'_n = [s'_1, ..., s'_n]$, $\text{Holds}(s_i, t_i)$ and $\text{Holds}(s'_j, t'_j)$, $t_i = < p_i, q_i >$ and $t'_j = < p'_j, q'_j >$ 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 $\text{OTCS}_L(S_m, S'_n)$.

1) Initiation: $s_0 = s'_0 = null$
for i = 0 : m: $\text{OTCS}_L(i, 0) = 0$
for j = 0 : n: $\text{OTCS}_L(0, j) = 0$

2) Recursion:

for i = 1: m
 for j = 1 : n
  if $s_i = s'_j$ # *matched*
   case 1: $s_{i-1} = s'_{j-1} = s_i = s'_j$
    $\text{OTCS}_L(i, j) = \text{OTCS}_L(i - 1, j - 1)$
   case 2: $s_{i-1} = s'_{j-1} \neq s_i = s'_j$
    $\text{OTCS}_L(i, j) = \text{OTCS}_L(i - 1, j - 1) + 1$
   case 3: $(s_{i-1} \neq s'_{j-1}) \,\&\, (either\ s_{i-1}\ or\ s'_{j-1} = s_i = s'_j)$
    $\text{OTCS}_L(i, j) = \max(\text{OTCS}_L(i-1, j), \text{OTCS}_L(i, j-1))$
   case 4: $(s_{i-1} \neq s'_{j-1}) \,\&\, (neither\ s_{i-1}\ nor\ s'_{j-1} = s_i = s'_j)$
    $\text{OTCS}_L(i, j) = \text{OTCS}_L(i - 1, j - 1) + 1$
  else # $s_i \neq s'_j$, *unmatched*
   $\text{OTCS}_L(i, j) = \max(\text{OTCS}_L(i-1, j), \text{OTCS}_L(i, j-1))$
  end # *end of if*
 end # *end of j*
end # *end of i*

3) Accomplishment

$\text{OTCS}_L(S_m, S'_n) = \text{OTCS}_L(m, n)$

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 = $\text{OTCS}_L(S_m, S'_n)$, $f_k, l_k \in$[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:

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

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

Finally, the overall similarity with respect to the temporal order, temporal duration and temporal gap is defined as:

$$\begin{aligned} OTCS(S_m, S'_n) = {} & w_L \bullet OTCS_L(S_m, S'_n) \\ & - w_D \bullet OTCS_D(S_m, S'_n) - w_G \bullet OTCS_G(S_m, S'_n) \end{aligned} \tag{3}$$

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 I 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 $w_D = w_G = 0.1$.

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-WLCS(C_1, C_1) < T-WLCS(C_1, C_2). Such a problem becomes absurd if, for instance, we have C_2' = 'aaaaaaaaaaaa', which will lead to T-WLCS(C_1,C_2') = 12 due to the unreasonable treatment to continuously duplicated common states.

TABLE I. 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]

IV. 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 pair quantizer proposed in [10] 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): shortened by deleting α% states from OQS randomly; Lengthened Query Set (LQS): lengthened by duplicating β% predecessors with random position in OQS.

Fig. 3 shows an example of key-frame sequence of video clip with various temporal duration and temporal gap.

Figure 3. An example of key-frame sequence in video clip database

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 II 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).

Fig.4 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.

TABLE II. RETRIEVAL PRECISION ON OQS

Top num / 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	

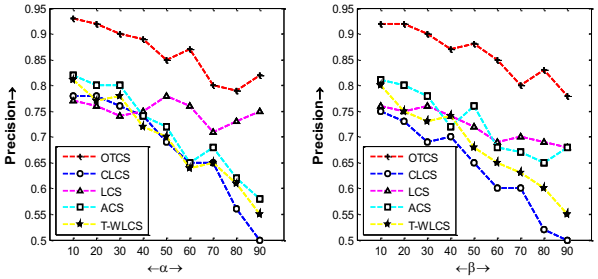


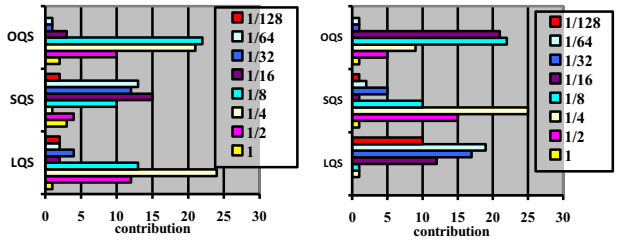
Figure 4. Retrieval Precision on SQS against α and LQS against β

Fig.5 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.

V. CONCLUSON AND FUTURE WORK

State-sequence matching is a very hot research topic in data mining [15]. In this paper, we have proposed a new concept of temporal common subsequence for state-sequence matching. An optimal temporal common subsequence algorithm has been developed, which takes into account of all the three essential temporal characters including temporal order, temporal duration and temporal gap for state-sequence matching. The comparison experiment with several representative LCS based algorithms conducted on news video retrieval demonstrates the effectiveness and robustness of the new algorithm.

Parameter (the values of weights) selection is a vital and arduous task in OTCS. How to automatically select the optimal values of weight remains as one of the research tasks in the future. In addition, more intelligent computation of the temporal durance difference and temporal gap difference appears interesting as our future work.



(a) temporal duration (b) temporal gap

Figure 5. Weights contribution of temporal characters in OTCS

REFERENCES

[1] 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. 2004, pp. 23-34.

[2] 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. 2003, pp.181–192.

[3] 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), 2009, pp.321-334.

[4] 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. 1993, pp.69-84.

[5] 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. 1990. pp.322-331.

[6] 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. 2001, pp. 263-272.

[7] 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.

[8] A. Gao, H. Siegelmann, "Time-Warped Longest Common Subsequence Algorithm for Music Retrieval". 5th International Conference on Music Information Retrieval (ISMIR'04),Oct.2004.

[9] 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. 2002, pp.406–417.

[10] 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. 2005, pp: 68 – 75.

[11] H. Wang, "All Common Subsequences", Proc. 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07), Jan. 2007, pp.635-640.

[12] J. Ma & P. Hayes, "Primitive Intervals Vs Point-Based Intervals: Rivals Or Allies?", the Computer Journal, 49(1), 2006, pp.32-41.

[13] J. Allen. "Towards a General Theory of Action and Time", Artificial Intelligence, 23, 1984, pp.123-154.

[14] 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. 2008, pp.325-329.

[15] 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), 2008, pp639 – 682.