一种带匹配路径约束的最长公共子序列长度算法

王前东*
(中国电子科技集团公司第十研究所 成都 610036)

摘 要：在带约束的最长公共子序列问题中提出一种特殊的新问题：假设有两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$, $\boldsymbol{Q}$ 中指定的匹配位置序列 $\boldsymbol{I}$，计算两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ 的最长公共子序列，且这个最长公共子序列的匹配路径必须经过位置序列 $\boldsymbol{I}$。针对此问题，该文提出一种带匹配路径约束的最长公共子序列算法。首先定义带匹配路径约束的最长公共子序列模型，其次推出该序列的性质，最后求出带匹配路径约束的最长公共子序列长度的基础算法和快速算法。基础算法和快速算法时间复杂度分别为 $O(mnt)$和 $O(mn)$, m, n, t 分别为序列 $\boldsymbol{Q}$, $\boldsymbol{C}$, $\boldsymbol{I}$ 的长度。

关键词：最长公共子序列；匹配路径约束；带约束的最长公共子序列；带匹配路径约束的最长公共子序列
中图分类号： TP301 **文献标识码**：A **文章编号**：1009-5896(2017)11-2615-05
DOI: 10.11999/JEIT170092

A Matching Path Constrained Longest Common Subsequence Length Algorithm

WANG Qiandong
(*No.*10 *Research Institute of China Electronics Technology Group Corporation*, *Chengdu* 610036, *China*)

Abstract: A special new problem is proposed in the constrained longest common subsequence problem. Given sequences $\boldsymbol{Q}$, $\boldsymbol{C}$ and the specific positions sequence $\boldsymbol{I}$ in $\boldsymbol{Q}$, the matching path constrained longest common subsequence problem for $\boldsymbol{Q}$ and $\boldsymbol{C}$ with respect to $\boldsymbol{I}$ is to find a Longest Common Subsequence (LCS) of $\boldsymbol{Q}$ and $\boldsymbol{C}$ such that the positions $\boldsymbol{I}$ in $\boldsymbol{Q}$ are in matching path of this LCS. A matching path constrained longest common subsequence algorithm is proposed for this problem. Firstly, a new model is defined for matching path constrained longest common subsequence. Secondly, the property of the subsequence is given. Lastly, a common method with $O(mnt)$ time and a fast method with $O(mn)$ time are respectively analyzed, where n, m and t are lengths of $\boldsymbol{Q}$, $\boldsymbol{C}$, and $\boldsymbol{I}$ respectively.

Key words: Longest Common Subsequence (LCS); Matching Path Constrained (MPC); A Constrained Longest Common Subsequence (CLCS); Matching Path Constrained Longest Common Subsequence (MPCLCS)

1 引言

最长公共子序列(Longest Common Subsequence, LCS)算法最早由Wagner等人[1]提出。LCS算法具有很好的抗干扰属性，在相似文献检索、基因序列比较、关键词查询、轨迹相似比较等中得到了广泛的应用[2,3]。

目前，基于LCS算法衍生了很多不同的算法，如带约束的最长公共子序列(Constrained Longest Common Subsequence, CLCS)算法。

CLCS算法：除长度为 m , n 的两序列外，增加一个长度为 t 的约束序列P，两序列的LCS必须是约束序列P的超序列。CLCS算法最早由Tsai[4]于2003年提出，是一个基于动态规划的算法，该算法的时间复杂度为 $O(tm^2n^2)$ ，算法耗时较多。Gotthilf等人[5]证明CLCS算法是NP难度的算法。后来Chin等人[6]和Arslan等人[7]等对CLCS算法进行了改进，相继于2004年和2005年独立提出时间复杂度为 $O(tmn)$ 的CLCS算法。迄今为止，各种对CLCS改进算法[8–11]相继被提出，但精确的CLCS改进算法的时间复杂度仍为 $O(tmn)$ 。

针对CLCS中一种特殊情形，本文提出一种新的CLCS子问题：假设有两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ ，以及 $\boldsymbol{Q}$ 中指定的位置序列 $\boldsymbol{I}$ ，计算两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ 的LCS，且 $\boldsymbol{Q}$ 中指定的位置 $\boldsymbol{I}$ 必须在这个LCS的匹配路径中。

如何解决这个问题，本文提出一种带匹配路径约束的最长公共子序列(Matching Path Constrained Longest Common Subsequence, MPCLCS)的基础算法和一种MPCLCS快速算法。快速算法时间复杂度为 $O(mn)$, m 和 n 分别为两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ 的长度。

收稿日期：2017-01-23；改回日期：2017-08-19；网络出版：2017-09-13
*通信作者：王前东 wangqiandong@sohu.com

本文第 2 节介绍目前已有的最长公共子序列算法及带约束的最长公共子序列算法。这个是本文算法基础。第 3 节给出了带匹配路径约束的最长公共子序列算法定义及相关性质。第 4 节给出了带匹配路径约束的最长公共子序列长度计算的基础算法和快速算法。最后总结全文，并给出了未来需要研究的方向。

2 基础知识[3−8,12]

2.1 最长公共子序列算法

定义 1 最长公共子序列(LCS)： 设两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$ 与 $\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 的公共子序列为 $\boldsymbol{Z}_r=(z_1,z_2,\cdots,z_r)$，则 $\boldsymbol{Z}_r$ 必须满足条件：

$$\left.\begin{aligned} &z_s=q_{i_s}=c_{j_s},\ 1\le s\le r\\ &1\le i_1\le i_2\le\cdots\le i_r\le m\\ &1\le j_1\le j_2\le\cdots\le j_r\le n\end{aligned}\right\} \tag{1}$$

$\boldsymbol{Z}_r$ 为两序列的最长公共子序列，当且仅当 $\boldsymbol{Z}_r$ 为满足式(1)条件中最长的序列。

根据最长公共子序列定义，文献[1]给出了如下的求最长公共子序列长度的算法。

算法 1 LCS长度算法

设 $\boldsymbol{Q}_i=(q_1,q_2,\cdots,q_i)$ 为 $\boldsymbol{Q}_m$ 的子序列，$\boldsymbol{C}_j=(c_1,c_2,\cdots,c_j)$ 为 $\boldsymbol{C}_n$ 的子序列，令 $\boldsymbol{L}(i,j)$ 表示序列 $\boldsymbol{Q}_i$ 与序列 $\boldsymbol{C}_j$ 的最长公共子序列长度，则 $\boldsymbol{L}(i,j)$ 有式(2)的递推计算公式：

$$\boldsymbol{L}(i,j)=\begin{cases}\max\{\boldsymbol{L}(i-1,j),\boldsymbol{L}(i,j-1)\},\\ \qquad i,j>0,\ q_i\ne c_j\\ \boldsymbol{L}(i-1,j-1)+1,\\ \qquad i,j>0,\ q_i=c_j\\ 0,\qquad i=0\ \text{或}\ j=0\end{cases} \tag{2}$$

式(2)为一般的最长公共子序列长度算法，$\boldsymbol{L}(m,n)$ 计算的时空复杂度为 $O(mn)$。

2.2 带约束的最长公共子序列算法

定义 2 带约束的最长公共子序列(CLCS)：设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及约束序列 $\boldsymbol{P}_t=(p_1,p_2,\cdots,p_t)$，则 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{P}_t$ 的CLCS为 $\boldsymbol{Z}_r=(z_1,z_2,\cdots,z_r)$，则 $\boldsymbol{Z}_r$ 必须满足条件：

$$\left.\begin{aligned} &z_{s_w}=q_{i_w}=c_{j_w}=p_w,\ 1\le w\le t\\ &1\le i_1\le i_2\le\cdots\le i_t\le m\\ &1\le j_1\le j_2\le\cdots\le j_t\le n\\ &1\le s_1\le s_2\le\cdots\le s_t\le r\end{aligned}\right\} \tag{3}$$

$\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{P}_t$ 的CLCS，当且仅当 $\boldsymbol{Z}_r$ 为满足式(3)的LCS。

根据带约束的最长公共子序列定义，文献[13,14]给出了如下的带约束的最长公共子序列长度算法。

算法 2 CLCS长度算法

设 $\boldsymbol{Q}_i$ 为 $\boldsymbol{Q}_m$ 的子序列，$\boldsymbol{C}_j$ 为 $\boldsymbol{C}_n$ 的子序列，$\boldsymbol{P}_k$ 为 $\boldsymbol{P}_t$ 的子序列，令 $\boldsymbol{L}(i,j,k)$ 表示序列 $\boldsymbol{Q}_i$ 与序列 $\boldsymbol{C}_j$ 关于 $\boldsymbol{P}_k$ 的最长公共子序列长度，则当 $i>0$，$j>0$，$k>0$ 时，$\boldsymbol{L}(i,j,k)$ 有式(4)的递推计算公式：

$$\boldsymbol{L}(i,j,k)=\begin{cases}\boldsymbol{L}(i-1,j-1,k-1)+1, & q_i=c_j=p_k\\ \boldsymbol{L}(i-1,j-1,k)+1, & q_i=c_j\ne p_k\\ \max\{\boldsymbol{L}(i,j-1,k),\boldsymbol{L}(i-1,j,k)\}, & q_i\ne c_j\end{cases} \tag{4}$$

其中，递推计算的初始值为

$$\boldsymbol{L}(i,j,k)=-\infty,\ k>0 \tag{5}$$

$$\boldsymbol{L}(i,j,0)=\begin{cases}\max\{\boldsymbol{L}(i,j-1,0),\ \boldsymbol{L}(i-1,j,0)\},\\ \qquad i,j>0,\ q_i\ne c_j\\ \boldsymbol{L}(i-1,j-1,0)+1,\ i,j>0,\ q_i=c_j\\ 0,\qquad i=0\ \text{或}\ j=0\end{cases} \tag{6}$$

上述公式为带约束最长公共子序列算法，$\boldsymbol{L}(m,n,t)$ 计算的时间复杂度为 $O(mnt)$，空间复杂度为 $O(mn)$。

3 带匹配路径约束的最长公共子序列定义及性质

3.1 带匹配路径约束的最长公共子序列的定义

定义3 带匹配路径约束的最长公共子序列(MPCLCS)： 设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$，其中 $1\le i_1<i_2<\cdots<i_t\le m$，$i_1$，$i_2$，$\cdots$，$i_t$ 为 $\boldsymbol{Q}_m$ 中元素的位置，则 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS为 $\boldsymbol{Z}_r=(z_1,z_2,\cdots,z_r)$，则 $\boldsymbol{Z}_r$ 必须满足条件：

$$\left.\begin{aligned} &z_{s_w}=q_{i_w}=c_{j_w},\ 1\le w\le t\\ &1\le s_1<s_2<\cdots<s_t\le r\\ &1\le j_1<j_2<\cdots<j_t\le n\end{aligned}\right\} \tag{7}$$

$\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS，当且仅当 $\boldsymbol{Z}_r$ 为满足式(7)的LCS。

例 1 有两序列 $\boldsymbol{Q}_8=(q_1,q_2,\cdots,q_8)=(A,B,D,C,A,B,E,H)$，$\boldsymbol{C}_9=(c_1,c_2,\cdots,c_9)=(A,B,F,D,E,A,B,H,G)$ 及约束序列 $\boldsymbol{P}_3=(p_1,p_2,p_3)=(A,B,E)$，$\boldsymbol{Q}$ 的约束位置序列 $\boldsymbol{I}_3=(5,6,7)$，则匹配结果如图1所示。从图1可看出：$\boldsymbol{Q}_8$ 与 $\boldsymbol{C}_9$ 的LCS为 $\boldsymbol{Z}_6=(A,B,D,A,B,H)$。

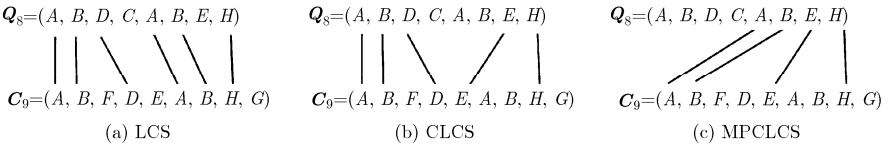

图 1 例1中的LCS与CLCS及MPCLCS

由于要求公共序列中必须包含序列 $\boldsymbol{P}_3=(A,B,E)$，故 $\boldsymbol{Q}_8$ 与 $\boldsymbol{C}_9$ 关于 $\boldsymbol{P}_3$ 的CLCS为 $\boldsymbol{Z}_5=(A,B,D,E,H)$。

由于要求 $\boldsymbol{Q}_8=(A,B,D,C,A,B,E,H)$ 中第5、6、7个元素匹配成功，故 $\boldsymbol{Q}_8$ 与 $\boldsymbol{C}_9$ 关于 $\boldsymbol{I}_3$ 的MPCLCS为 $\boldsymbol{Z}_4=(A,B,E,H)$。

CLCS与MPCLCS都是带约束的序列，故CLCS长度与MPCLC长度都不大于LCS长度。

虽然MPCLCS的 $\boldsymbol{I}_3$ 约束的序列与CLCS的 $\boldsymbol{P}_3$ 相同，但是MPCLCS还要求约束序列的元素必须是 $\boldsymbol{I}_3$ 指定位置的元素，故此例中MPCLCS序列是CLCS序列的子序列。

例 2　有两序列 $\boldsymbol{Q}_8=(q_1,q_2,\cdots,q_8)=(E,B,D,C,A,B,E,H)$，$\boldsymbol{C}_9=(c_1,c_2,\cdots,c_9)=(A,B,F,D,E,A,B,H,G)$ 及约束序列 $\boldsymbol{P}_3=(p_1,p_2,p_3)=(A,B,E)$，$\boldsymbol{Q}$ 的约束位置序列 $\boldsymbol{I}_3=(5,6,7)$，则匹配结果如图2所示。从图2看出：$\boldsymbol{Q}_8$ 与 $\boldsymbol{C}_9$ 关于 $\boldsymbol{P}_3$ 的CLCS为 $\boldsymbol{Z}_5=(A,B,E,H)$。$\boldsymbol{Q}_8$ 与 $\boldsymbol{C}_9$ 关于 $\boldsymbol{I}_3$ 的MPCLCS为 $\boldsymbol{Z}_4=(A,B,E,H)$。

CLCS与MPCLCS都是带约束的序列，约束序列表示的子序列都是 (A,B,E)，且约束序列在两序列中有且仅有一个子序列，故此例中MPCLCS序列等于CLCS序列。

3.2 带匹配路径约束的最长公共子序列性质

设 $\boldsymbol{Z}_r=(z_1,z_2,\cdots,z_r)$ 为两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$ 与 $\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 关于 $\boldsymbol{Q}_m$ 的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$ 的MPCLCS，则MPCLCS有如下性质：

(1)若 $m=i_t$，$q_m=c_n$，则有 $z_r=q_m=c_n$，且 $\boldsymbol{Z}_{r-1}$ 为 $\boldsymbol{Q}_{m-1}$ 与 $\boldsymbol{C}_{n-1}$ 关于 $\boldsymbol{I}_{t-1}$ 的MPCLCS。

(2)若 $m=i_t$，$q_m\neq c_n$，则有 $z_r=q_m$，且 $\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_{n-1}$ 关于 $\boldsymbol{I}_t$ 的MPCLCS。

(3)若 $m>i_t$，$q_m=c_n$，则 $\boldsymbol{Z}_{r-1}$ 为 $\boldsymbol{Q}_{m-1}$ 与 $\boldsymbol{C}_{n-1}$ 关于 $\boldsymbol{I}_t$ 的MPCLCS或 $\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_{m-1}$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS。

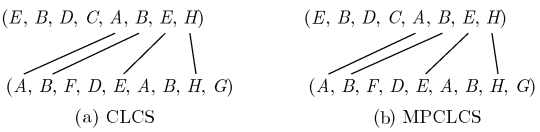

图 2　MPCLCS 与 CLCS 相同

(4)若 $m>i_t$，$q_m\neq c_n$，则 $\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_{m-1}$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS或 $\boldsymbol{Z}_r$ 为 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_{n-1}$ 关于 $\boldsymbol{I}_t$ 的MPCLCS。

上述性质证明略。

4 带匹配路径约束的最长公共子序列长度计算算法

4.1 带匹配路径约束的最长公共子序列长度的基础算法

根据带匹配路径约束的最长公共子序列定义3，给出带匹配路径约束的最长公共子序列长度基础算法3。

算法 3　MPCLCS长度基础算法。

设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$，其中 $1\le i_1<i_2<\cdots<i_t\le m$，$i_1,i_2,\cdots,i_t$ 为 $\boldsymbol{Q}_m$ 中 t 个约束元素的位置。令MPCLCS长度 $\boldsymbol{L}_{\boldsymbol{Q}}^t(m,n)=-\infty$ 表示 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于约束位置序列 $\boldsymbol{I}_t$ 的MPCLCS不存在，则当 $t>0$，$m>0$，$n>0$ 时，$\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS长度基础算法的公式 $\boldsymbol{L}_{\boldsymbol{Q}}^t(m,n)$ 可由式(8)的递推公式计算求得

$$\boldsymbol{L}_{\boldsymbol{Q}}^k(i,j)=\begin{cases}-\infty, & i<i_k\\ \boldsymbol{L}_{\boldsymbol{Q}}^{k-1}(i-1,j-1)+1, & q_i=c_j,i=i_k\\ \boldsymbol{L}_{\boldsymbol{Q}}^k(i,j-1), & q_i\neq c_j,i=i_k\\ \max\left\{\boldsymbol{L}_{\boldsymbol{Q}}^k(i-1,j-1)+1,\ \boldsymbol{L}_{\boldsymbol{Q}}^k(i-1,j)\right\}, & q_i=c_j,i>i_k\\ \max\left\{\boldsymbol{L}_{\boldsymbol{Q}}^k(i,j-1),\ \boldsymbol{L}_{\boldsymbol{Q}}^k(i-1,j)\right\}, & q_i\neq c_j,i>i_k\end{cases}\tag{8}$$

其中，递推计算的初始值为

$$\boldsymbol{L}_{\boldsymbol{Q}}^k(i,0)=-\infty,k>0\tag{9}$$

$$\boldsymbol{L}_{\boldsymbol{Q}}^0(i,j)=\begin{cases}\max\left\{\boldsymbol{L}_{\boldsymbol{Q}}^0(i-1,j),\boldsymbol{L}_{\boldsymbol{Q}}^0(i,j-1)\right\}, & i,j>0,\ q_i\neq c_j\\ \boldsymbol{L}_{\boldsymbol{Q}}^0(i-1,j-1)+1, & i,j>0,\ q_i=c_j\\ 0, & i=0\text{或}j=0\end{cases}\tag{10}$$

式(10)表示的 $\boldsymbol{L}_{\boldsymbol{Q}}^0(i,j)$ 为LCS算法的式(2)。

当 $k>0$ 时，下面分3种情形分别证明式(8)表示的 $\boldsymbol{L}_{\boldsymbol{Q}}^k(i,j)$ 的递推计算公式成立。

(1) $i<i_k$，由于 $i<i_k$，故 $\boldsymbol{Q}_i$ 与序列 $\boldsymbol{C}_j$ 关于 $\boldsymbol{Q}_i$ 的约束位置序列 $\boldsymbol{I}_k$ 的MPCLCS不存在。由此知式(11)成立：

$$\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i,j)=-\infty,\quad i<i_k \tag{11}$$

(2) $i=i_k$，由MPCLCS的性质(1)得式(12)成立：

$$\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i_k,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k-1}(i_k-1,j-1)+1,\ q_i=c_j \tag{12}$$

由MPCLCS的性质(2)得式(13)成立：

$$\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i_k,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i_k,j-1),\quad q_i\neq c_j \tag{13}$$

(3) $i>i_k$，由MPCLCS的性质(3)得式(14)成立：

$$\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i,j)=\max\left\{\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i-1,j-1)+1,\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i-1,j)\right\},\quad q_i=c_j,\quad i>i_k \tag{14}$$

由MPCLCS的性质(4)得式(15)成立：

$$\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i,j)=\max\left\{\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i,j-1),\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i-1,j)\right\},\quad q_i\neq c_j,\ i>i_k \tag{15}$$

由式(11)–式(15)可得式(8)在上述3种情形下都成立。

由递推公式(8)，有定理1成立。

定 理 1 设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,\ i_2,\cdots,i_t)$，则 $\boldsymbol{Q}_m$ 与 $\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的MPCLCS长度基础算法 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$ 的时空复杂度为 $O(mnt)$。

证明 类似于参考文献[13,14]中算法2中的证明，利用式(8)，$k=1,2,\cdots,t$，$i=1,2,\cdots,m$，$j=1,2,\cdots,n$，经过三重循环算出 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$，于是MPCLCS长度 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$ 的基础算法的时空复杂度为 $O(mnt)$。 证毕

4.2 带匹配路径约束的最长公共子序列长度的快速算法

根据带匹配路径约束的最长公共子序列长度的基础算法3，推出带匹配路径约束的最长公共子序列长度快速算法4。

算法 4 MPCLCS长度快速算法。

设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$，其中 $1\le i_1<i_2<\cdots<i_t\le m$，$i_1,i_2,\cdots,i_t$ 为 $\boldsymbol{Q}_m$ 中 t 个约束元素的位置。定义如式(16)所示的递推公式 $\boldsymbol{M}(i,j)$：

$$\boldsymbol{M}(i,j)=\begin{cases}\boldsymbol{M}(i-1,j-1)+1, & q_i=c_j,\ i\in\boldsymbol{I}_t\\ \boldsymbol{M}(i,j-1), & q_i\neq c_j,\ i\in\boldsymbol{I}_t\\ \max\left\{\boldsymbol{M}(i-1,j-1)+1,\boldsymbol{M}(i-1,j)\right\}, & q_i=c_j,\ i\notin\boldsymbol{I}_t\\ \max\left\{\boldsymbol{M}(i,j-1),\boldsymbol{M}(i-1,j)\right\}, & q_i\neq c_j,\ i\notin\boldsymbol{I}_t\end{cases} \tag{16}$$

其中，递推计算的初始值由式(17)求得：

$$\boldsymbol{M}(i,j)=\begin{cases}0, & i<i_1\\ -\infty, & i\ge i_1\end{cases} \tag{17}$$

其中，初始值式(17)中 i_1 为 $\boldsymbol{I}_t$ 中第1个约束位置，若 $t=0$，即 $\boldsymbol{I}_t=\boldsymbol{I}_0$ 为空集，则 $i_1=m+1$。

下面证明由递推公式(16)和初始值式(17)求出的 $\boldsymbol{M}(m,n)$ 与MPCLCS基础算法公式求出的 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$ 相同，即有定理2成立。

定 理 2 设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$，其中 $1\le i_1<i_2<\cdots<i_t\le m$，$i_1,i_2,\cdots,i_t$ 为 $\boldsymbol{Q}_m$ 中 t 个约束元素的位置。令 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$ 为MPCLCS长度基础算法求出的序列 $\boldsymbol{Q}_m$，序列 $\boldsymbol{C}_n$ 关于序列 $\boldsymbol{I}_t$ 的MPCLCS长度，$\boldsymbol{M}(m,n)$ 为递推公式(16)和初始值式(17)所求出的值，则有 $\boldsymbol{M}(m,n)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$。

证明该定理之前先给出几个引理，其证明略。

引理 1 当 $i<i_1$ 时，$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{0}(i,j)$ 成立。

引理 2 假设任意 $j\le n$，都有 $\boldsymbol{M}(i_k-1,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k-1}(i_k-1,j)$ 成立，则对任意的 $j\le n$，$\boldsymbol{M}(i_k,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i_k,j)$ 成立。

引理 3 假设存在 $i_k<i<i_{k+1}$，对任意 $j\le n$，都有 $\boldsymbol{M}(i-1,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i-1,j)$ 成立，则对任意的 $j\le n$，$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{k}(i,j)$ 成立。

引理 4 假设存在 $i>i_t$ 任意 $j\le n$，都有 $\boldsymbol{M}(i-1,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(i-1,j)$ 成立，则对任意的 $j\le n$，$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(i,j)$ 成立。

定理2的证明：

由引理1可知式(18)结论成立：

$$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{0}(i,j),\quad i<i_1 \tag{18}$$

利用式(18)知 $i=i_1-1$ 时也成立，则由引理2递推可得 $i=i_1$ 时也成立，即

$$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{1}(i,j),\quad i=i_1 \tag{19}$$

利用式(19)知 $i=i_1$ 时成立，由引理3递推可得式(20)成立：

$$\boldsymbol{M}(i,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{1}(i,j),\quad i_1<i<i_2 \tag{20}$$

同式(19)的证明，利用式(20)知 $i=i_2-1$ 时成立，则由引理2递推可得 $i=i_2$ 时也成立。

同式(20)的证明，由 $i=i_2$ 时成立，则由引理3递推可得 $i_2<i<i_3$ 时也成立。

如此反复利用引理2和引理3，递推可得 $i=i_t$ 时成立，即：$\boldsymbol{M}(i_t,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(i_t,j)$。由 $\boldsymbol{M}(i_t,j)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(i_t,j)$，利用引理4递推可得 $i>i_t$ 时也成立，即

$$\boldsymbol{M}(i,j)\ =\boldsymbol{L}_{\boldsymbol{Q}}^{t}(i,j),\quad i>i_t \tag{21}$$

由式(21)可知：$\boldsymbol{M}(m,n)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$，故定理2

成立。　　证毕

由递推公式(16)，有定理3成立。

定理 3　设有两序列 $\boldsymbol{Q}_m=(q_1,q_2,\cdots,q_m)$，$\boldsymbol{C}_n=(c_1,c_2,\cdots,c_n)$ 及 $\boldsymbol{Q}_m$ 中指定的约束位置序列 $\boldsymbol{I}_t=(i_1,i_2,\cdots,i_t)$，则 $\boldsymbol{Q}_m$，$\boldsymbol{C}_n$ 关于 $\boldsymbol{I}_t$ 的递推公式 $\boldsymbol{M}(m,n)$ 的时空复杂度为 $O(mn)$。

证明　利用公式(16)，$i=1,2,\cdots,m$，$j=1,2,\cdots,n$，经过两重循环算出 $\boldsymbol{M}(m,n)$，于是 $\boldsymbol{M}(m,n)$ 的时空复杂度为 $O(mn)$。　　证毕

由定理2，$\boldsymbol{M}(m,n)=\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$，即 $\boldsymbol{M}(m,n)$ 也是计算MPCLCS长度的公式。由定理3，计算 $\boldsymbol{M}(m,n)$ 的时空复杂度均为 $O(mn)$，比 $\boldsymbol{L}_{\boldsymbol{Q}}^{t}(m,n)$ 的复杂度低，故 $\boldsymbol{M}(m,n)$ 为MPCLCS的快速算法公式。

5　结束语

本文提出了一种特殊的新的CLCS问题：假设有两序列 $\boldsymbol{Q}$，$\boldsymbol{C}$，以及 $\boldsymbol{Q}$ 中指定的约束位置序列 $\boldsymbol{I}$，计算两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ 的最长公共子序列，且这个最长公共子序列的匹配路径必须经过约束位置序列 $\boldsymbol{I}$。针对此特定的CLCS问题，给出了一种计算最长公共子序列长度的 $O(mn)$ 时间复杂度的快速MPCLCS算法。快速MPCLCS算法比时间复杂度为 $O(mnt)$ 的CLCS算法的时间复杂度低。m 和 n 分别为两序列 $\boldsymbol{Q}$ 和 $\boldsymbol{C}$ 的长度，t 为约束序列的长度。

在实际应用中，本文的算法可应用于航迹的规律分析及航迹的识别等应用中，后续会根据工程需要，将该算法进行工程化改进，应用于相似航迹的查询识别和目标活动规律分析中。

参 考 文 献

[1] WANGGER R A and FISCHER M J. The string-to-string correction problem[J]. *Journal of the Association for Computing Machinery*, 1974, 21(1): 168–173.

[2] WANG Haoxin, ZHONG Jingdong, and ZHANG Defu. A duplicate code checking algorithm for the programming experiment[C]. 2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), Sliema, 2015: 39–42. doi: 10.1109/MCSI.2015.12.

[3] 赵建军, 陈滨, 杨利斌, 等. 一种基于字符串模型的轨迹相似度计算[J]. 科学技术与工程, 2013, 13(1): 80–84.
ZHAO Jianjun, CHEN Bin, YANG Libin, *et al*. Measure similarity between trajectories based on alphabetic string model[J]. *Science Technology and Engineering*, 2013, 13(1): 80–84.

[4] TSAI Y T. The constrained longest common subsequence problem[J]. *Information Processing Letters*, 2003, 88(4): 173–176. doi: 10.1016/j.ipl.2003.07.001.

[5] GOTTHILF Z, HTSAI YERMELIN D, and LEWENSTEIN M. Constrained LCS: Hardness and approximation[C]. Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching, Berlin, 2008: 255–262. doi: 10.1007/978-3-540-69068-9_24.

[6] CHIN F, SANTIS A, FERRARA A, *et al*. A simple algorithm for the constrained sequence problems[J]. *Information Processing Letters*, 2004, 90(4): 175–179. doi: 10.1016/j.ipl.2004.02.008.

[7] ARSLAN A and EGECIOGLU O. Algorithms for the constrained longest common subsequence problems[J]. *International Journal of Foundations of Computer Science*, 2005, 16(6): 1099–1109. doi: 10.1142/S0129054105003674.

[8] BECERRA D, SOTO W, NINO L, *et al*. An algorithm for constrained LCS[C]. IEEE/ACS International Conference on Computer Systems and Applications, Hammamet Tunisia, 2010: 1–7. doi: 10.1109/AICCSA.2010.5586937.

[9] BONIZZONI P, VEDOVA G, DONDI R, *et al*. Variants of constrained longest common subsequence[J]. *Information Processing Letters*, 2010, 110(20): 877–881. doi: 10.1016/j.ipl.2010.07.015.

[10] 业宁, 朱大铭, 张倩倩, 等. 带约束最长公共子序列快速算法[J]. 南京大学学报(自然科学版), 2009, 45(5): 576–584.
YE Ning, ZHU Daming, ZHANG Qianqian, *et al*. Fast algorithm of the longest common subsequence with constraints[J]. *Journal of Nanjing Universitv*, 2009, 45(5): 576–584.

[11] TSENG C T, YANG C B, and ANN H Y. Efficient algorithms for the longest common subsequence problem with sequential substring constraints[J]. *Journal of Complexity*, 2013, 29(1): 44–52. doi: 10.1109/BIBE.2011.34.

[12] 魏龙翔, 何小海, 滕奇志, 等. 结合Hausdorff距离和最长公共子序列的轨迹分类[J]. 电子与信息学报, 2013, 35(4): 784–790. doi: 10.3724/SP.J.1146.2012.01078.
WEI Longxiang, HE Xiaohai, TENG Qizhi, *et al*. Trajectory classification based on Hausdorff distance and longest common subsequence[J]. *Journal of Electronics & Information Technology*, 2013, 35(4): 784–790. doi: 10.3724/SP.J.1146.2012.01078.

[13] BEAL R, AFRIN T, FARHEEN A, *et al*. A new algorithm for "the LCS problem" with application in compressing genome resequencing data[C]. 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Washington, DC, 2015: 69–74. doi: 10.1109/BIBM.2015.7359657.

[14] LIU Richen, GUO Hanqi, ZHANG Jiang, *et al*. Comparative visualization of vector field ensembles based on longest common subsequence[C]. 2016 IEEE Pacific Visualization Symposium (PacificVis), Taipei, 2016: 96–103. doi: 10.1109/PACIFICVIS.2016.7465256.

王前东：　男，1977 年生，高级工程师，数据信息处理及应用.