An Efficient Dynamic Programming Algorithm for a New Generalized LCS Problem

Daxin Zhu, Lei Wang, Jun Tian* and Xiaodong Wang*

***Abstract*—In this paper, we consider a generalized longest common subsequence problem, in which a constraining sequence of length s must be included as a substring and the other constraining sequence of length t must be included as a subsequence of two main sequences and the length of the result must be maximal. For the two input sequences X and Y of lengths n and m, and the given two constraining sequences of length s and t, we present an $O(nmst)$ time dynamic programming algorithm for solving the new generalized longest common subsequence problem. The time complexity can be reduced further to cubic time in a more detailed analysis. The correctness of the new algorithm is proved.**

***Index Terms*—generalized longest common subsequence, NP-hard problems, dynamic programming, time complexity**

I. Introduction

The longest common subsequence (LCS) problem is a well-known measurement for computing the similarity of two strings. It can be broadly applied in diverse areas, such as file comparison, pattern matching and computational biology [3], [4], [8]–[12], [14], [15].

Given two sequences X and Y, the longest common subsequence (LCS) problem is to find a subsequence of X and Y whose length is the longest among all common subsequences of the two given sequences.

For some biological applications some constraints must be applied to the LCS problem. These kinds of variants of the LCS problem are called the constrained LCS (CLCS) problem. Recently, Chen and Chao [1] proposed the more generalized forms of the CLCS problem, the generalized constrained longest common subsequence (GC-LCS) problem. For the two input sequences X and Y of lengths n and m,respectively, and a constraint string P of length r, the GC-LCS problem is a set of four problems which are to find the LCS of X and Y including/excluding P as a subsequence/substring, respectively.

In this paper, we consider a more general constrained longest common subsequence problem called SEQ-IC-STR-IC-LCS, in which a constraining sequence of length s must be included as a substring and the other constraining sequence of length t must be included as a subsequence of two main sequences and the length of the result must be maximal. We will present the first efficient dynamic programming algorithm for solving this problem.

The organization of the paper is reproduced below.

In the following 4 sections, we describe our presented dynamic programming algorithm for the SEQ-IC-STR-IC-LCS problem.

In Section 2 the preliminary knowledge for presenting our algorithm for the SEQ-IC-STR-IC-LCS problem is discussed. In Section 3 we give a new dynamic programming solution for the SEQ-IC-STR-IC-LCS problem with time complexity $O(nmst)$, where n and m are the lengths of the two given input strings, and s and t the lengths of the two constraining sequences. In Section 4 the time complexity is further improved to $O(nms)$. Some concluding remarks are in Section 5.

II. Characterization of the generalized LCS problem

A sequence is a string of characters over an alphabet $\sum$. A subsequence of a sequence X is obtained by deleting zero or more characters from X (not necessarily contiguous). A substring of a sequence X is a subsequence of successive characters within X.

For a given sequence $X = x_1x_2 \cdots x_n$ of length n, the ith character of X is denoted as $x_i \in \sum$ for any $i = 1, \cdots, n$. A substring of X from position i to j can be denoted as $X[i : j] = x_ix_{i+1} \cdots x_j$. If $i \neq 1$ or $j \neq n$, then the substring $X[i : j] = x_ix_{i+1} \cdots x_j$ is called a proper substring of X. A substring $X[i : j] = x_ix_{i+1} \cdots x_j$ is called a prefix or a suffix of X if $i = 1$ or $j = n$, respectively.

An appearance of sequence $X = x_1x_2 \cdots x_n$ in sequence $Y = y_1y_2 \cdots y_m$, for any X and Y, starting at position j is a sequence of strictly increasing indexes $i_1, i_2, \cdots, i_n$ such that $i_1 = j$, and $X = y_{i_1}, y_{i_2}, \cdots, y_{i_n}$. A compact

Manuscript received December 9, 2015; revised February 22, 2016.

This work was supported in part by the Quanzhou Foundation of Science and Technology under Grant No.2013Z38, Fujian Provincial Key Laboratory of Data-Intensive Computing and Fujian University Laboratory of Intelligent Computing and Information Processing.

Daxin Zhu is with Quanzhou Normal University, Quanzhou, China.(email:dex@qztc.edu.cn)

Lei Wang is with Facebook, 1 Hacker Way, Menlo Park, CA 94052, USA.

Jun Tian is with Fujian Medical University, Fuzhou, China.

Xiaodong Wang is with Fujian University of Technology, Fuzhou, China.

*Corresponding author.

appearance of X in Y starting at position j is the appearance of the smallest last index i_n. A match for sequences X and Y is a pair (i, j) such that $x_i = y_j$. The total number of matches for X and Y is denoted by δ. It is obvious that $\delta \le nm$.

For the two input sequences $X = x_1x_2\cdots x_n$ and $Y = y_1y_2\cdots y_m$ of lengths n and m, respectively, and two constrained sequences $P = p_1p_2\cdots p_s$ and $Q = q_1q_2\cdots q_t$ of lengths s and t, the SEQ-IC-STR-IC-LCS problem is to find a constrained LCS of X and Y including P as a subsequence and including Q as a substring.

Definiton 1: Let $Z(i, j, k, r)$ denote the set of all LCSs of $X[1 : i]$ and $Y[1 : j]$ such that for each $z \in Z(i, j, k, r)$, z includes $P[1 : k]$ as a subsequence, and includes $Q[1 : r]$ as a substring, where $1 \le i \le n, 1 \le j \le m, 0 \le k \le s$, and $0 \le r \le t$. The length of an LCS in $Z(i, j, k, r)$ is denoted as $g(i, j, k, r)$.

Definiton 2: Let $W(i, j, k, r)$ denote the set of all LCSs of $X[1 : i]$ and $Y[1 : j]$ such that for each $w \in W(i, j, k, r)$, w includes $P[1 : k]$ as a subsequence, and includes $Q[1 : r]$ as a suffix, where $1 \le i \le n, 1 \le j \le m, 0 \le k \le s$, and $0 \le r \le t$. The length of an LCS in $W(i, j, k, r)$ is denoted as $f(i, j, k, r)$.

Definiton 3: Let $U(i, j, k)$ denote the set of all LCSs of $X[i : n]$ and $Y[j : m]$ such that for each $u \in U(i, j, k)$, u includes $P[k : s]$ as a subsequence, where $1 \le i \le n, 1 \le j \le m, 0 \le k \le s$. The length of an LCS in $U(i, j, k)$ is denoted as $h(i, j, k)$.

Definiton 4: Let $V(i, j, k)$ denote the set of all LCSs of $X[1 : i]$ and $Y[1 : j]$ such that for each $v \in V(i, j, k)$, v includes $P[1 : k]$ as a subsequence, where $1 \le i \le n, 1 \le j \le m, 0 \le k \le s$. The length of an LCS in $V(i, j, k)$ is denoted as $v(i, j, k)$.

A problem decomposition on the STR-IC-LCS problem was pointed out in [5]. A similar property is also valid for the SEQ-IC-STR-IC-LCS problem.

Property 1: If $Z[1 : l] = z_1, z_2, \cdots, z_l \in Z(n, m, s, t)$, and for some $t \le l' \le l$, $Z[l' - t + 1 : l'] = Q[1 : t]$, then $Z[1 : l]$ is a concatenation of the following two substrings, for some $1 \le i \le n$ and $1 \le j \le m$:

1) The prefix $Z[1 : l']$: $Z[1 : l']$ is an LCS Z_1 of $X[1 : i]$ and $Y[1 : j]$ including $P[1 : k]$ as a subsequence, and including Q as the suffix, i.e. $Z[1 : l'] \in W(i, j, k, t)$.
2) The suffix $Z[l' + 1 : l]$: $Z[l' + 1 : l]$ is an LCS Z_2 of $X[i+1 : n]$ and $Y[j+1 : m]$ including $P[k+1 : s]$ as a subsequence, i.e. $Z[l'+1 : l] \in V(i+1, j+1, k+1)$.

The following theorem characterizes the structure of an optimal solution based on optimal solutions to subproblems, for computing the LCSs in $W(i, j, k, r)$, for any $1 \le i \le n, 1 \le j \le m, 0 \le k \le s$, and $0 \le r \le t$.

Theorem 1: If $Z[1 : l] = z_1, z_2, \cdots, z_l \in W(i, j, k, r)$, then the following conditions hold:

1) If $r = 0$, $x_i = y_j$ and $k > 0, x_i = p_k$, then $z_l = x_i = y_j = p_k$ and $Z[1 : l - 1] \in W(i - 1, j - 1, k - 1, r)$.
2) If $r = 0$, $x_i = y_j$ and $k = 0$ or $k > 0, x_i \ne p_k$, then $z_l = x_i = y_j$ and $Z[1 : l - 1] \in W(i - 1, j - 1, k, r)$.
3) If $r > 0$, $x_i = y_j = q_r$ and $k > 0, x_i = p_k$, then $z_l = x_i = y_j = p_k = q_r$ and $Z[1 : l - 1] \in W(i - 1, j - 1, k - 1, r - 1)$.
4) If $r > 0$, $x_i = y_j = q_r$ and $k = 0$ or $k > 0, x_i \ne p_k$, then $z_l = x_i = y_j = q_r$ and $Z[1 : l - 1] \in W(i - 1, j - 1, k, r - 1)$.
5) If $r > 0$, $x_i = y_j$ and $x_i \ne q_r$, then $z_l \ne x_i$ and $Z[1 : l] \in W(i - 1, j - 1, k, r)$.
6) If $x_i \ne y_j$, then $z_l \ne x_i$ implies $Z[1 : l] \in W(i - 1, j, k, r)$.
7) If $x_i \ne y_j$, then $z_l \ne y_j$ implies $Z[1 : l] \in W(i, j - 1, k, r)$.

Proof.

1. In this case, we do not have any constraints on Q, due to $r = 0$. Since $x_i = y_j = p_k = z_l$, $Z[1 : l-1]$ is a common subsequence of $X[1 : i - 1]$ and $Y[1 : j - 1]$ including $P[1 : k-1]$ as a subsequence. We can show that $Z[1 : l-1]$ is an LCS of $X[1 : i-1]$ and $Y[1 : j-1]$ including $P[1 : k-1]$ as a subsequence. Assume by contradiction that there exists a common subsequence a of $X[1 : i - 1]$ and $Y[1 : j - 1]$ including $P[1 : k - 1]$ as a subsequence, whose length is greater than $l - 1$. Then the concatenation of a and z_l will result in a common subsequence of $X[1 : i]$ and $Y[1 : j]$ including $P[1 : k]$ as a subsequence, whose length is greater than l. This is a contradiction.

2. In this case, since $x_i = y_j \ne p_k$, we have $x_i = y_j = z_l$ and $z_l \ne p_k$. Therefore, $Z[1 : l-1]$ is a common subsequence of $X[1 : i - 1]$ and $Y[1 : j - 1]$ including $P[1 : k]$ as a subsequence. We can show that $Z[1 : l - 1]$ is an LCS of $X[1 : i - 1]$ and $Y[1 : j - 1]$ including $P[1 : k]$ as a subsequence. Assume by contradiction that there exists a common subsequence a of $X[1 : i - 1]$ and $Y[1 : j - 1]$ including $P[1 : k]$ as a subsequence, whose length is greater than $l - 1$. Then the concatenation of a and z_l will result in a common subsequence of $X[1 : i]$ and $Y[1 : j]$ including $P[1 : k]$ as a subsequence, whose length is greater than l. This is a contradiction.

3. Since $x_i = y_j = p_k = q_r$, we have $x_i = y_j = z_l$ and $Z[1 : l-1]$ is a common subsequence of $X[1 : i-1]$ and $Y[1 : j - 1]$ including $P[1 : k - 1]$ as a subsequence and including $Q[1 : r - 1]$ as a suffix. We can show that $Z[1 : l - 1]$ is an LCS of $X[1 : i-1]$ and $Y[1 : j-1]$ including $P[1 : k-1]$ as a subsequence and including $Q[1 : r-1]$ as a suffix. Assume by contradiction that there exists a common subsequence a

of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k-1]$ as a subsequence and including $Q[1:r-1]$ as a suffix, whose length is greater than $l-1$. Then the concatenation of a and z_l will result in a common subsequence of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix, whose length is greater than l. This is a contradiction.

4. Since $x_i = y_j = q_r$ and $x_i \neq p_k$, we have $x_i = y_j = z_l$ and $Z[1:l-1]$ is a common subsequence of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r-1]$ as a suffix. We can show that $Z[1:l-1]$ is an LCS of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r-1]$ as a suffix. Assume by contradiction that there exists a common subsequence a of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r-1]$ as a suffix, whose length is greater than $l-1$. Then the concatenation of a and z_l will result in a common subsequence of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix, whose length is greater than l. This is a contradiction.

5. In this case, if $x_i = y_j = z_l$, then $z_l \neq q_r$, and thus $Q[1:r]$ is not a suffix of $Z[1:l]$. Therefore, we have $x_i = y_j \neq z_l$, and $Z[1:l]$ must be a common subsequence of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix. It is obvious that $Z[1:l]$ is also an LCS of $X[1:i-1]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix.

6. Since $x_i \neq y_j$ and $z_l \neq x_i$, $Z[1:l]$ must be a common subsequence of $X[1:i-1]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix. It is obvious that $Z[1:l]$ is also an LCS of $X[1:i-1]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix.

7. Since $x_i \neq y_j$ and $z_l \neq y_j$, $Z[1:l]$ must be a common subsequence of $X[1:i]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix. It is obvious that $Z[1:l]$ is also an LCS of $X[1:i]$ and $Y[1:j-1]$ including $P[1:k]$ as a subsequence and including $Q[1:r]$ as a suffix.

The proof is completed. □

III. A SIMPLE DYNAMIC PROGRAMMING ALGORITHM

Our new algorithm for solving the SEQ-IC-STR-IC-LCS problem is composed of three main stages. The first stage is to find LCSs in $W(i,j,k,r)$. Let $f(i,j,k,r)$ denote the length of an LCS in $W(i,j,k,r)$. By the optimal substructure properties of the SEQ-IC-STR-IC-LCS problem shown in Theorem 1, we can build the following recursive formula for computing $f(i,j,k,r)$. For any $1 \leq i \leq n, 1 \leq j \leq m, 0 \leq k \leq s$, and $0 \leq r \leq t$, the values of $f(i,j,k,r)$ can be computed by the following recursive formula (1).

The boundary conditions of this recursive formula are $f(i,0,0,0) = f(0,j,0,0) = 0$ and $f(i,0,k,r) = f(0,j,k,r) = -\infty$ for any $0 \leq i \leq n, 0 \leq j \leq m, 0 \leq k \leq s$, and $0 \leq r \leq t$.

Based on this formula, our algorithm for computing $f(i,j,k,r)$ is a standard dynamic programming algorithm. By the recursive formula (1), the dynamic programming algorithm for computing $f(i,j,k,r)$ can be implemented as the following Algorithm 1.

Algorithm 1 Suffix

Input: Strings $X = x_1 \cdots x_n$, $Y = y_1 \cdots y_m$ of lengths n and m, respectively, and two constrained sequences $P = p_1 p_2 \cdots p_s$ and $Q = q_1 q_2 \cdots q_t$ of lengths s and t

Output: $f(i,j,k,r)$, the length of an LCS of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence, and including $Q[1:r]$ as a suffix, for all $1 \leq i \leq n, 1 \leq j \leq m, 0 \leq k \leq s$, and $0 \leq r \leq t$.

for all i,j,k,r , $0 \leq i \leq n, 0 \leq j \leq m, 0 \leq k \leq s$ and $0 \leq r \leq t$ **do**
  $f(i,0,k,r), f(0,j,k,r) \leftarrow -\infty, f(i,0,0,0), f(0,j,0,0) \leftarrow 0$ {boundary condition}
end for
for all i,j,k,r , $1 \leq i \leq n, 1 \leq j \leq m, 0 \leq k \leq s$ and $0 \leq r \leq t$ **do**
  if $x_i \neq y_j$ **then**
    $f(i,j,k,r) \leftarrow \max\{f(i-1,j,k,r), f(i,j-1,k,r)\}$
  else if $r = 0$ **then**
    if $k > 0$ **and** $x_i = p_k$ **then**
      $f(i,j,k,r) \leftarrow 1 + f(i-1,j-1,k-1,r)$
    else
      $f(i,j,k,r) \leftarrow 1 + f(i-1,j-1,k,r)$
    end if
  else if $x_i = q_r$ **then**
    if $k > 0$ **and** $x_i = p_k$ **then**
      $f(i,j,k,r) \leftarrow 1 + f(i-1,j-1,k-1,r-1)$
    else
      $f(i,j,k,r) \leftarrow 1 + f(i-1,j-1,k,r-1)$
    end if
  else
    $f(i,j,k,r) \leftarrow f(i-1,j-1,k,r)$
  end if
end for

It is obvious that the algorithm requires $O(nmst)$ time and space. For each value of $f(i,j,k,r)$ computed by algorithm

$$f(i,j,k,r)=\begin{cases}\max\{f(i-1,j,k,r),f(i,j-1,k,r)\} & \text{if } x_i\neq y_j\\ 1+f(i-1,j-1,k-1,r) & \text{if } r=0\wedge x_i=y_j\wedge k>0\wedge x_i=p_k\\ 1+f(i-1,j-1,k,r) & \text{if } r=0\wedge x_i=y_j\wedge (k=0\vee x_i\neq p_k)\\ 1+f(i-1,j-1,k-1,r-1) & \text{if } k>0\wedge r>0\wedge x_i=y_j=p_k=q_r\\ 1+f(i-1,j-1,k,r-1) & \text{if } r>0\wedge x_i=y_j=q_r\wedge (k=0\vee x_i\neq p_k)\\ f(i-1,j-1,k,r) & \text{if } r>0\wedge x_i=y_j\wedge x_i\neq q_r\end{cases}\qquad(1)$$

Suffix, the corresponding LCS of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence, and including $Q[1:r]$ as a suffix, can be constructed by backtracking through the computation paths from (i,j,k,r) to $(0,0,0,0)$. The following algorithm $back(i,j,k,r)$ is the backtracking algorithm to obtain the LCS, not only its length. The time complexity of the algorithm $back(i,j,k,r)$ is obviously $O(n+m)$.

Algorithm 2 $back(i,j,k,r)$

Input: Integers i,j,k,r

Output: The LCS of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence and $Q[1:r]$ as a suffix

if $i<1$ **or** $j<1$ **then**
  return
end if
if $x_i\neq y_j$ **then**
  if $f(i-1,j,k,r)>f(i,j-1,k,r)$ **then**
    $back(i-1,j,k,r)$
  else
    $back(i,j-1,k,r)$
  end if
else if $r=0$ **then**
  if $k>0$ **and** $x_i=p_k$ **then**
    $back(i-1,j-1,k-1,r)$
    print x_i
  else
    $back(i-1,j-1,k,r)$
    print x_i
  end if
else if $x_i=q_r$ **then**
  if $k>0$ **and** $x_i=p_k$ **then**
    $back(i-1,j-1,k-1,r-1)$
    print x_i
  else
    $back(i-1,j-1,k,r-1)$
    print x_i
  end if
else
  $back(i-1,j-1,k,r)$
end if

The second stage of our algorithm is to find LCSs in $U(i,j,k)$. The length of an LCS in $U(i,j,k)$ is denoted as $h(i,j,k)$. Chin et al. [5] presented a dynamic programming algorithm with $O(nms)$ time and space. A reverse version of the dynamic programming algorithm for computing $h(i,j,k)$ can be described as follows.

Algorithm 3 SEQ-IC-R

Input: Strings $X=x_1\cdots x_n$, $Y=y_1\cdots y_m$ of lengths n and m, respectively, and a constrained sequence $P=p_1p_2\cdots p_s$ of lengths s

Output: $h(i,j,k)$, the length of an LCS of $X[i:n]$ and $Y[j:m]$ including $P[k:s]$ as a subsequence, for all $1\leq i\leq n, 1\leq j\leq m, 0\leq k\leq s$.

for all i,j,k , $0\leq i\leq n, 0\leq j\leq m, 1\leq k\leq s$ **do**
  $h(i,m+1,k), h(n+1,j,k)\leftarrow -\infty$ {boundary condition}
end for
for $i=n$ down to 1 **do**
  for $j=m$ down to 1 **do**
    for $k=s+1$ down to 1 **do**
      if $x_i\neq y_j$ **then**
        $h(i,j,k)\leftarrow\max\{h(i+1,j,k),h(i,j+1,k)\}$
      else
        if $k>s$ **or** $k\leq s$ **and** $x_i\neq p_k$ **then**
          $h(i,j,k)\leftarrow 1+h(i+1,j+1,k)$
        else if $x_i=p_k$ **then**
          $h(i,j,k)\leftarrow 1+h(i+1,j+1,k+1)$
        end if
      end if
    end for
  end for
end for

For each value of $h(i,j,k)$ computed by algorithm *SEQ-IC-R*, the corresponding LCS of $X[i:n]$ and $Y[j:m]$ including $P[k:s]$ as a subsequence, can be constructed by backtracking through the computation paths from (i,j,k) to $(0,0,0)$. The following algorithm $backr(i,j,k)$ is the backtracking algorithm to obtain the corresponding LCS, not only its length. The time complexity of the algorithm $backr(i,j,k)$ is obviously $O(n+m)$.

By Property 1 for the SEQ-IC-STR-IC-LCS problem, the dynamic programming matrices $f(i,j,k,r)$ and $h(i,j,k)$

Algorithm 4 $backr(i, j, k)$

Input: Integers i, j, k

Output: The LCS of $X[i : n]$ and $Y[j : m]$ including $P[k : s]$ as a subsequence

if $i > n$ **or** $j > m$ **then**
  return
end if
if $x_i \neq y_j$ **then**
  if $h(i+1, j, k) > h(i, j+1, k)$ **then**
    $backr(i+1, j, k)$
  else
    $backr(i, j+1, k)$
  end if
else
  if $k > s$ **or** $k \leq s$ **and** $x_i \neq p_k$ **then**
    print x_i
    $backr(i+1, j+1, k)$
  else if $x_i = p_k$ **then**
    print x_i
    $backr(i+1, j+1, k+1)$
  end if
end if

computed by the algorithms *Suffix* and *SEQ-IC-R* can now be combined to obtain the solutions of the SEQ-IC-STR-IC-LCS problem as follows. This is the last stage of our algorithm.

From the 'for' loops of the algorithm, it is readily seen that the algorithm requires $O(nms)$ time. Therefore, the overall time of our algorithm for solving the SEQ-IC-STR-IC-LCS problem is $O(nmst)$.

IV. Improvements of the algorithm

S. Deorowicz [3] proposed the first quadratic-time algorithm for the STR-IC-LCS problem. A similar idea can be exploited to improve the time complexity of our dynamic programming algorithm for solving the SEQ-IC-STR-IC-LCS problem. The improved algorithm is also based on dynamic programming with some preprocessing. To show its correctness it is necessary to prove some more structural properties of the problem.

Let $Z[1 : l] = z_1, z_2, \cdots, z_l \in Z(n, m, s, t)$, be a constrained LCS of X and Y including P as a subsequence and including Q as a substring. Let also $I = (i_1, j_1), (i_2, j_2), \cdots, (i_l, j_l)$ be a sequence of indices of X and Y such that $Z[1 : l] = x_{i_1}, x_{i_2}, \cdots, x_{i_l}$ and $Z[1 : l] = y_{j_1}, y_{j_2}, \cdots, y_{j_l}$. From the problem statement, there must exist an index $d \in [1, l-t+1]$ such that $Q = x_{i_d}, x_{i_{d+1}}, \cdots, x_{i_{d+t-1}}$ and $Q = y_{j_d}, y_{j_{d+1}}, \cdots, y_{j_{d+t-1}}$.

Algorithm 5 SEQ-IC-STR-IC-LCS

Input: Strings $X = x_1 \cdots x_n$, $Y = y_1 \cdots y_m$ of lengths n and m, respectively, and two constrained sequences $P = p_1 p_2 \cdots p_s$ and $Q = q_1 q_2 \cdots q_t$ of lengths s and t

Output: The constrained LCS of X and Y including P as a subsequence, and including Q as a substring.

Suffix {compute $f(i, j, k, r)$}
SEQ-IC-R {compute $h(i, j, k)$}
$i^*, j^*, k^* \leftarrow 0, tmp \leftarrow -\infty$
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    for $k = 1$ to s **do**
      $x \leftarrow f(i, j, k, t) + h(i+1, j+1, k+1)$
      if $tmp < x$ **then**
        $tmp \leftarrow x, i^* \leftarrow i, j^* \leftarrow j, k^* \leftarrow k$
      end if
    end for
  end for
end for
if $tmp > 0$ **then**
  $back(i^*, j^*, k^*, t)$
  $backr(i^*+1, j^*+1, k^*+1)$
end if
return $\max\{0, tmp\}, i^*, j^*, k^*$

Theorem 2: Let $i'_d = i_d$ and for all $e \in [1, t-1]$, i'_{d+e} be the smallest possible, but larger than i'_{d+e-1}, index of X such that $x_{i_{d+e}} = x_{i'_{d+e}}$. The sequence of indices

$$I' = (i_1, j_1), (i_2, j_2), \cdots, (i_{d-1}, j_{d-1}), (i'_d, j_d), (i'_{d+1}, j_{d+1}), \cdots,$$

$$(i'_{d+t-1}, j_{d+t-1}), (i_{d+t}, j_{d+t}), \cdots, (i_l, j_l)$$

defines the same constrained LCS as $Z[1 : l]$.

Proof.

From the definition of indices i'_{d+e}, it is obvious that they form an increasing sequence, since $i'_d = i_d$, and $i'_{d+t-1} \leq i_{d+t-1}$. The sequence $i'_d, \cdots, i'_{d+t-1}$ is of course a compact appearance of Q in X starting at i_d. Therefore, both components of I' pairs form increasing sequences and for any (i'_u, j_u), $x_{i'_u} = y_{j_u}$. Therefore, I' defines the same constrained LCS as $Z[1 : l]$.

The proof is completed. □

The same property is also true for the jth components of the sequence I. Therefore, we can conclude that when finding a constrained LCS in $Z(i, j, k, r)$, instead of checking any common subsequences of X and Y it suffices to check only such common subsequences that contain compact appearances of Q both in X and Y. The number of different compact appearances of Q in X and Y will be denoted by δ_x and δ_y, respectively. It is obvious that $\delta_x \delta_y \leq \delta$, since a pair (i, j) defines a compact appearance of Q in X starting at ith position and compact appearance of Q in Y starting at jth position only for some matches.

Base on Theorem 2, we can reduce the time complexity of our dynamic programming from $O(nmst)$ to $O(nms)$. The improved algorithm consists of also three principal stages. In the first stage, both sequences X and Y are preprocessed to determine two corresponding arrays lx and ly. For each occurrence i of the first character q_1 of Q in X, the index j of the last character q_t of a compact appearance of Q in X is recorded as $lx_i = j$. A similar preprocessing is applicable to the sequence Y.

Algorithm 6 Prep

Input: X, Y

Output: For each $1 \leq i \leq n$, the minimal index $r = lx_i$ such that $X[i : r]$ includes Q as a subsequence
For each $1 \leq j \leq m$, the minimal index $r = ly_j$ such that $Y[j : r]$ includes Q as a subsequence

for $i = 1$ to n **do**
  if $x_i = q_1$ **then**
    $lx_i \leftarrow left(X, n, i)$
  else
    $lx_i \leftarrow 0$
  end if
end for
for $j = 1$ to m **do**
  if $y_j = q_1$ **then**
    $ly_j \leftarrow left(Y, m, j)$
  else
    $ly_j \leftarrow 0$
  end if
end for

In the algorithm Prep, function $left$ is used to find the index lx_i of the last character q_t of a compact appearance of Q.

In the second stage of the improved algorithm, two DP matrices of SEQ-IC-LCS problem are computed: $h(i, j, k)$, the reverse one defined by Definition 3, and $v(i, j, k)$, the forward one defined by Definition 4. Both of the DP matrices can be computed by the SEQ-IC-LCS algorithm of Chin et al. [5].

In the last stage, two preprocessed arrays lx and ly are used to determine the final results. To this end for each match (i, j) for X and Y the ends (lx_i, ly_i) of compact appearances of Q in X starting at position i and in Y starting at position j are read. The length of an SEQ-IC-STR-IC-

Algorithm 7 $left(X, n, i)$

Input: Integers n, i and $X[1 : n]$

Output: The minimal index r such that $X[i : r]$ includes Q as a subsequence

$a \leftarrow i + 1, b \leftarrow 2$
while $a \leq n$ **and** $b \leq t$ **do**
  if $x_a = q_b$ **then**
    $b \leftarrow b + 1$
  else
    $a \leftarrow a + 1$
  end if
end while
if $b > t$ **then**
  return $a - 1$
else
  return 0
end if

Algorithm 8 SEQ-IC

Input: Strings $X = x_1 \cdots x_n$, $Y = y_1 \cdots y_m$ of lengths n and m, respectively, and a constrained sequence $P = p_1 p_2 \cdots p_s$ of length s

Output: $v(i, j, k)$, the length of an LCS of $X[1 : i]$ and $Y[1 : j]$ including $P[1 : k]$ as a subsequence, for all $1 \leq i \leq n, 1 \leq j \leq m, 0 \leq k \leq s$.

for all i, j, k , $0 \leq i \leq n, 0 \leq j \leq m, 1 \leq k \leq s$ **do**
  $h(i, 0, k), h(0, j, k) \leftarrow -\infty$ {boundary condition}
end for
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    for $k = 0$ to s **do**
      if $x_i \neq y_j$ **then**
        $v(i, j, k) \leftarrow \max\{v(i-1, j, k), v(i, j-1, k)\}$
      else
        if $k = 0$ **or** $k > 0$ **and** $x_i \neq p_k$ **then**
          $v(i, j, k) \leftarrow 1 + v(i-1, j-1, k)$
        else if $x_i = p_k$ **then**
          $v(i, j, k) \leftarrow 1 + v(i-1, j-1, k-1)$
        end if
      end if
    end for
  end for
end for

Algorithm 9 $backf(i, j, k)$

Input: Integers i, j, k

Output: The LCS of $X[1:i]$ and $Y[1:j]$ including $P[1:k]$ as a subsequence

if $i < 1$ **or** $j < 1$ **then**
  return
end if
if $x_i \neq y_j$ **then**
  if $v(i-1, j, k) > v(i, j-1, k)$ **then**
    $backr(i-1, j, k)$
  else
    $backr(i, j-1, k)$
  end if
else
  if $k = 0$ **or** $k > 0$ **and** $x_i \neq p_k$ **then**
    $backr(i-1, j-1, k)$
    print x_i
  else if $x_i = p_k$ **then**
    $backr(i-1, j-1, k-1)$
    print x_i
  end if
end if

LCS, $g(n, m, s, t)$ defined by Definition 1, containing these appearances of Q is determined as a sum of three parts. The first part is, for some indices i, j, k, r, $v(i-1, j-1, k)$, the constrained LCS length of prefixes of X and Y ending at positions $i-1$ and $j-1$, including $P[1:k]$ as a subsequence. The second part is $h(lx_i + 1, ly_j + 1, r + 1)$, the constrained LCS length of suffixes of X and Y starting at positions lx_i+1 and $ly_j + 1$, including $P[r + 1 : t]$ as a subsequence, where the index r can be determined by k. The last part is t, the length of the constrained sequence Q.

For each integer $k, 1 \leq k \leq s$, let $premax(k)$ denote the maximum length l $(0 \leq l \leq t - k + 1)$ such that Q includes $P[k : k + l - 1]$ as a subsequence.

Since the constrained LCS A of the prefixes of X and Y ending at positions $i - 1$ and $j - 1$, includes $P[1 : k]$ as a subsequence, the concatenation of A and Q will include $P[1 : r]$ as a subsequence, where $r = k + premax(k + 1)$.

The constrained LCS B of the suffixes of X and Y starting at positions $lx_i + 1$ and $ly_j + 1$, includes $P[r + 1 : t]$ as a subsequence. Therefore, the concatenation of A,Q and B includes P as a subsequence.

According to the matrices $v(i, j, k)$ and $h(i, j, k)$, backtracking can be used to obtain the optimal subsequence, not only its length.

Theorem 3: The algorithm SEQ-IC-STR-IC-LCS correctly computes a constrained LCS in $Z(n, m, s, t)$. The algorithm requires $O(nms)$ time and to $O(nms)$ space in

Algorithm 10 $premax(k)$

Input: Integers k

Output: The maximum length r $(0 \leq r \leq t - k + 1)$ such that Q includes $P[k : k + r - 1]$ as a subsequence

$a \leftarrow k, b \leftarrow 1, r \leftarrow 0$
while $a \leq s$ **and** $b \leq t$ **do**
  if $p_a = q_b$ **then**
    $a \leftarrow a + 1, r \leftarrow r + 1$
  else
    $b \leftarrow b + 1$
  end if
end while
return r

Algorithm 11 SEQ-IC-STR-IC-LCS

Input: Strings $X = x_1 \cdots x_n$, $Y = y_1 \cdots y_m$ of lengths n and m, respectively, and two constrained sequences $P = p_1 p_2 \cdots p_s$ and $Q = q_1 q_2 \cdots q_t$ of lengths s and t

Output: The length of an LCS of X and Y including P as a subsequence, and including Q as a substring.

SEQ-IC {compute $v(i, j, k)$}
SEQ-IC-R {compute $h(i, j, k)$}
Prep {compute lx, ly}
$i^*, j^*, k^*, r^* \leftarrow 0, tmp \leftarrow 0$
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    if $lx_i = ly_j$ **then**
      for $k = 0$ to s **do**
        $r \leftarrow k + premax(k + 1)$
        $c \leftarrow v(i-1, j-1, k) + h(lx_i + 1, ly_j + 1, r + 1) + t$
        if $tmp < c$ **then**
          $tmp \leftarrow c, i^* \leftarrow i, j^* \leftarrow j, k^* \leftarrow k, r^* \leftarrow r$
        end if
      end for
    end if
  end for
end for
if $tmp > 0$ **then**
  $backf(i^* - 1, j^* - 1, k^*)$
  print Q
  $backr(lx_{i^*} + 1, ly_{j^*} + 1, r^* + 1)$
end if
return $\max\{0, tmp\}, i^*, j^*, k^*, r^*$

the worst case.

Proof. The time and space complexities of the algorithm are dominated by the computation of the two dynamic programming matrices $v(i,j,k)$ and $h(i,j,k)$. It is obvious that they are all $O(nms)$ in the worst case.

The proof is completed. □

V. Concluding remarks

We have suggested a new dynamic programming solution for the new generalized constrained longest common subsequence problem SEQ-IC-STR-IC-LCS. The first dynamic programming algorithm requires $O(nmst)$ in the worst case, where n, m, s, t are the lengths of the four input sequences respectively. The time complexity can be reduced further to cubic time in a more thorough analysis. Many other generalized constrained longest common subsequence (GC-LCS) problems have analogous structures. It is not clear that whether the similar technique of this paper can be applied to these problems to achieve efficient algorithms. We will explore these problems further.

References

[1] Chen Y.C., Chao K.M. On the generalized constrained longest common subsequence problems, *J. Comb. Optim.* 21(3), 2011, pp. 383-392.

[2] Crochemore M.,Hancart C., and Lecroq T., Algorithms on strings, Cambridge University Press, Cambridge, UK, 2007.

[3] Deorowicz S., Quadratic-time algorithm for a string constrained LCS problem, *Inform. Process. Lett.* 112(11), 2012, pp. 423-426.

[4] Deorowicz S., Obstoj J., Constrained longest common subsequence computing algorithms in practice, *Comput. Inform.* 29(3), 2010, pp. 427-445.

[5] Gotthilf Z., Hermelin D., Lewenstein M., Constrained LCS: hardness and approximation. In: *Proceedings of the 19th annual symposium on combinatorial pattern matching, CPM'08*, Pisa, Italy, 2008, pp. 255-262.

[6] Gotthilf Z., Hermelin D., Landau G.M., Lewenstein M., Restricted LCS. In: *Proceedings of the 17th international conference on string processing and information retrieval, SPIRE'10*, Los Cabos, Mexico, 2010, pp. 250-257.

[7] Gusfield, D.,Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, UK, 1997.

[8] Khand M. A. H., Akter S., Rashid M. A., and Yaakob S. B., Velocity Tentative PSO: An Optimal Velocity Implementation based Particle Swarm Optimization to Solve Traveling Salesman Problem, *IAENG International Journal of Computer Science*, vol. 42, no.3, 2015, pp221-232.

[9] Klement V., and Oberhuber T., Multigrid Method for Linear Complementarity Problem and Its Implementation on GPU, *IAENG International Journal of Applied Mathematics*, vol. 45, no.3, 2015, pp193-197.

[10] Mirdehghan S.M., M. Nazaari A., and Vakili J., Relations Among Technical, Cost and Revenue Efficiencies in Data Envelopment Analysis, *IAENG International Journal of Applied Mathematics*, vol. 45, no.4, 2015, pp249-258.

[11] Peng Y.H., Yang C.B., Huang K.S., Tseng K.T., An algorithm and applications to sequence alignment with weighted constraints, *Int. J. Found. Comput. Sci.* 21(1),2010, pp. 51-59.

[12] Tang C.Y., Lu C.L., Constrained multiple sequence alignment tool development and its application to RNase family alignment, *J. Bioinform. Comput. Biol.* 1, 2003, pp. 267-287.

[13] Tseng C.T., Yang C.B., Ann H.Y., Efficient algorithms for the longest common subsequence problem with sequential substring constraints. *J. Complexity* 29, 2013, pp. 44-52.

[14] Yan J., Li M., and Xu J., An Adaptive Strategy Applied to Memetic Algorithms, *IAENG International Journal of Computer Science*, vol. 42, no.2, 2015, pp73-84.

[15] Zhu D., Wang L., Tian J., and Wang X., A Simple Polynomial Time Algorithm for the Generalized LCS Problem with Multiple Substring Exclusive Constraints, *IAENG International Journal of Computer Science*, vol. 42, no.3, 2015, pp214-220.