J Comb Optim (2011) 21: 383–392
DOI 10.1007/s10878-009-9262-5

# On the generalized constrained longest common subsequence problems

**Yi-Ching Chen · Kun-Mao Chao**

Published online: 27 August 2009


**Abstract** We investigate four variants of the longest common subsequence problem. Given two sequences $X$, $Y$ and a constrained pattern $P$ of lengths $m$, $n$, and $\rho$, respectively, the generalized constrained longest common subsequence (GC-LCS) problems are to find a longest common subsequence of $X$ and $Y$ including (or excluding) $P$ as a subsequence (or substring). We propose new dynamic programming algorithms for solving the GC-LCS problems in $O(mn\rho)$ time. We also consider the case where the number of constrained patterns is arbitrary.

**Keywords** Algorithms · Longest common subsequence · Dynamic programming

## 1 Introduction

A sequence is a string of characters over an alphabet $\Sigma$. 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 $X$. Given a sequence $X$ of length $m$, let $X[i]$ denote the $i$th character of $X$ for any $i = 1, \ldots, m$. We also let $X[i..j]$ denote the subsequence of consecutive characters in $X$ from position $i$ to position $j$ if $1 \le i \le j \le m$, and an empty string otherwise. For example, if $X = \texttt{algorithm}$, then $X[7] = \texttt{t}$ and $X[3..5] = \texttt{gor}$.

This research was supported in part by NSC grants NSC 95-2221-E-002-126-MY3 and NSC 97-2221-E-002-097-MY3 from the National Science Council, Taiwan.

Y.-C. Chen
Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

K.-M. Chao (✉)
Department of Computer Science and Information Engineering, Graduate Institute of Biomedical Electronics and Bioinformatics, Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 106, Taiwan
e-mail: kmchao@csie.ntu.edu.tw

A common subsequence of two sequences is a subsequence that appears in both sequences. A longest common subsequence (LCS) of two sequences is a maximum-length common subsequence of the sequences. The LCS problem is to find an LCS of two given sequences, which is a fundamental problem in computer science. Wagner and Fischer (1974) proposed the LCS problem and solved it by computing the edit distance between the sequences in quadratic time and space. Hirschberg (1975) presented a quadratic-time and linear-space algorithm for the LCS problem. The LCS problem has been studied intensively to improve the time complexity for decades (Aho et al. 1976; Apostolico and Guerra 1987; Bergroth et al. 2000; Bonizzoni et al. 2007; Cormen et al. 2001; Gusfield 1997; Hirschberg 1977; Hunt and Szymanski 1977; Masek and Paterson 1980; Pevzner 2000; Rahman and Iliopoulos 2007). The LCS problem with an arbitrary number of sequences, even on a binary alphabet, is NP-hard (Maier 1978).

The LCS problem is a special case of the sequence alignment problem (Chao and Zhang 2009; Gusfield 1997), and has many applications in molecular biology and pattern recognition. One major application is to measure the similarity of sequences. In the evolutionary molecular biology, a significant segment of the DNA sequence might have been conserved in different species. To take a common specific segment into account for the similarity measurement, the LCS problem with an inclusive constraint, named the inclusion-constrained longest common subsequence (IC-LCS) problem, is considered. We address two IC-LCS problems as follows.

**Problem 1** (SEQ-IC-LCS) (Arslan and Eğecioğlu 2005; Chin et al. 2004; Iliopoulos and Rahman 2008; Tsai 2003) Given two sequences $X$, $Y$ and a constrained pattern $P$ of lengths $m$, $n$, and $\rho$, respectively, the SEQ-IC-LCS problem is to find an LCS of $X$ and $Y$ including $P$ as a subsequence.

**Problem 2** (STR-IC-LCS) Given two sequences $X$, $Y$ and a constrained pattern $P$ of lengths $m$, $n$, and $\rho$, respectively, the STR-IC-LCS problem is to find an LCS of $X$ and $Y$ including $P$ as a substring.

For example, if $X = \texttt{AATGCCTAGGC}$, $Y = \texttt{CGATCTGGAC}$, and $P = \texttt{GTAC}$, an LCS of $X$ and $Y$ is `ATCTGGC`, and outputs of the SEQ-IC-LCS and STR-IC-LCS problems are `GCTAC` and `GTAC`, respectively.

The SEQ-IC-LCS problem was proposed and solved in $O(m^2n^2\rho)$ time by Tsai (2003). Later, Chin et al. (2004) and Arslan and Eğecioğlu (2005) independently presented two improved algorithms with $O(mn\rho)$ time. Chin et al. (2004) also showed that this problem is equivalent to a special case of the constrained multiple sequence alignment problem (Chin et al. 2005; Tang et al. 2003). Without loss of generality, assume that $m \le n$. Recently, Iliopoulos and Rahman (2008) proposed an $O(\rho r \log\log n + n)$-time algorithm by employing the van Emde Boas tree (van Emde Boas 1977; van Emde Boas et al. 1977), where $r$ is the total number of ordered pairs of positions at which $X$ and $Y$ match. It should be noted, however, that the preprocessing time of the van Emde Boas tree was not evaluated when analyzing the time complexity.

| Problem | Input | Output |
|---|---|---|
| SEQ-IC-LCS | | An LCS of $X$ and $Y$ including $P$ as a subsequence |
| STR-IC-LCS | $X$, $Y$, and $P$ | An LCS of $X$ and $Y$ including $P$ as a substring |
| SEQ-EC-LCS | | An LCS of $X$ and $Y$ excluding $P$ as a subsequence |
| STR-EC-LCS | | An LCS of $X$ and $Y$ excluding $P$ as a substring |

**Fig. 1** The GC-LCS problems

To take more common flexible structures into account for the similarity measurement, we extend the definition of the IC-LCS problem to be the LCS problem with an exclusive constraint, named the exclusion-constrained longest common subsequence (EC-LCS) problem. We address two EC-LCS problems as follows.

**Problem 3** (SEQ-EC-LCS) Given two sequences $X$, $Y$ and a constrained pattern $P$ of lengths $m$, $n$, and $\rho$, respectively, the SEQ-EC-LCS problem is to find an LCS of $X$ and $Y$ excluding $P$ as a subsequence.

**Problem 4** (STR-EC-LCS) Given two sequences $X$, $Y$ and a constrained pattern $P$ of lengths $m$, $n$, and $\rho$, respectively, the STR-EC-LCS problem is to find an LCS of $X$ and $Y$ excluding $P$ as a substring.

For example, suppose $X = \texttt{AATGCCTAGGC}$ and $Y = \texttt{CGATCTGGAC}$. If $P = \texttt{TGC}$, an output of the SEQ-EC-LCS problem is `ATCTGG`. If $P = \texttt{TG}$, an output of the STR-EC-LCS problem is `ATCGGC`.

The four variants of the LCS problem are named as the generalized constrained longest common subsequence (GC-LCS) problems and summarized in Fig. 1. To our knowledge, the STR-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems are discussed for the first time.

In Sect. 2, we present three $O(mn\rho)$-time algorithms for solving the STR-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems, respectively. We also consider the GC-LCS problems with an arbitrary number of constrained patterns in Sect. 3. Finally, concluding remarks are given in Sect. 4.

## 2 The algorithms

In the following, we propose three dynamic programming algorithms for solving the STR-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems, respectively.

### 2.1 An $O(mn\rho)$-time algorithm for the STR-IC-LCS problem

The STR-IC-LCS problem is to find an LCS of two sequences $X$ and $Y$ *including* a constrained pattern $P$ as a *substring*. Property 1 shows the characterization of the structure of a solution to the STR-IC-LCS problem.

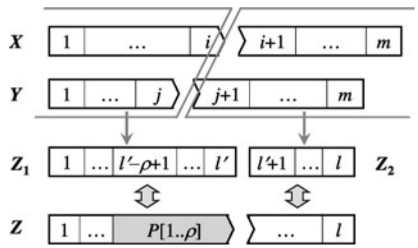


**Fig. 2** An illustration of the problem decomposition on the STR-IC-LCS problem. $Z$ is an LCS of $X$ and $Y$ including $P$ as a substring, and is also a concatenation of the substrings $Z_1$ and $Z_2$, where $Z_1$ is an LCS of $X[1..i]$ and $Y[1..j]$ including $P$ as the suffix $Z[l' - \rho + 1..l']$, and $Z_2$ is an LCS of $X[i + 1..m]$ and $Y[j + 1..n]$, for some $0 \le i \le m$, $0 \le j \le n$, and $\rho \le l' \le l$

*Property 1* If $Z[1..l]$ is an LCS of $X[1..m]$ and $Y[1..n]$ including $P$ as the substring $Z[l' - \rho + 1..l']$ for some $\rho \le l' \le l$, then $Z[1..l]$ is a concatenation of the following two substrings, for some $0 \le i \le m$ and $0 \le j \le n$:

1. The prefix $Z[1..l']$: $Z[1..l']$ is an LCS $Z_1$ of $X[1..i]$ and $Y[1..j]$ including $P$ as the suffix $Z[l' - \rho + 1..l']$, and
2. The suffix $Z[l' + 1..l]$: $Z[l' + 1..l]$ is an LCS $Z_2$ of $X[i + 1..m]$ and $Y[j + 1..n]$.

Figure 2 illustrates the concept of the problem decomposition shown in Property 1. Based on solutions of the STR-IC-LCS problem to subproblems, we solve it by first computing an LCS of $X[1..i]$ and $Y[1..j]$ including $P$ as a suffix and an LCS of $X[i..m]$ and $Y[j..n]$ for all $1 \le i \le m$ and $1 \le j \le n$. The solutions to the two subproblems are then merged to determine a longest concatenation. A quadratic-time algorithm for computing an LCS of $X[i..m]$ and $Y[j..n]$ (Cormen et al. 2001) is employed. For obtaining an LCS of $X[1..i]$ and $Y[1..j]$ including $P$ as a suffix, Theorem 1 decomposes the structure of an optimal solution based on the solutions to its smaller subproblems.

**Theorem 1** *Let $S_{i,j,k}$ denote the set of all LCSs of $X[1..i]$ and $Y[1..j]$ including $P[1..k]$ as a suffix. If $Z[1..l] \in S_{i,j,k}$, the following conditions hold*:

(1) *If $X[i] = Y[j] = P[k]$ and $k > 0$, then $Z[l] = X[i] = Y[j] = P[k]$ and $Z[1..l - 1] \in S_{i-1,j-1,k-1}$.*
(2) *If $X[i] = Y[j]$, $X[i] \ne P[k]$, and $k > 0$, then $Z[l] \ne X[i]$ and $Z[1..l] \in S_{i-1,j-1,k}$.*
(3) *If $X[i] = Y[j]$ and $k = 0$, then $Z[l] = X[i] = Y[j]$ and $Z[1..l - 1] \in S_{i-1,j-1,k}$.*
(4) *If $X[i] \ne Y[j]$, then $Z[l] \ne X[i]$ implies $Z[1..l] \in S_{i-1,j,k}$.*
(5) *If $X[i] \ne Y[j]$, then $Z[l] \ne Y[j]$ implies $Z[1..l] \in S_{i,j-1,k}$.*

*Proof* We prove this theorem case by case. (1) Since $P[1..k]$ is a suffix of $Z[1..l]$, we have $Z[l] = P[k]$. If $Z[l] \ne X[i]$, we could append $X[i] = Y[j] = P[k]$ to $Z[1..l-1]$ obtain a common subsequence of length $l$, and the resulting sequence also includes $P[1..k]$ as a suffix. Thus, $Z[1..l - 1]$ is a common subsequence of $X[1..i - 1]$ and $Y[1..j - 1]$ including $P[1..k - 1]$ as the suffix $Z[l - k + 1..l - 1]$. Assume by contradiction that there exists a common subsequence $Z'[1..l]$ of $X[1..i-1]$ and $Y[1..j-1]$ including $P[1..k-1]$ as the suffix $Z'[l-k+2..l]$. We could append $X[i] = Y[j] = P[k]$ to $Z'[1..l]$ to obtain a common subsequence of $X[1..i]$ and $Y[1..j]$ of length greater than $l$ such that $P[1..k]$ is the suffix $S[l-k+2..l+1]$, contradicting the hypothesis of $Z[1..l] \in S_{i,j,k}$.

(2) If $Z[l] = X[i]$, then $Z[l] \neq P[k]$ and $P[1..k]$ is not a suffix of $Z[1..l]$. Thus, we have $Z[l] \neq X[i]$ and $Z[1..l] \in S_{i-1,j-1,k}$. Assume by contradiction that there exists a common subsequence $Z'[1..l+1]$ of $X[1..i-1]$ and $Y[1..j-1]$ containing $P[1..k]$ as a suffix. Obviously, $Z'[1..l+1]$ is also a common subsequence of $X[1..i]$ and $Y[1..j]$ of length greater than $l$ such that $P[1..k]$ is a suffix. This contradicts the hypothesis of $Z[1..l] \in S_{i,j,k}$.

(3) This case is equivalent to the case of $X[i] = Y[j]$ in the LCS problem. Hence, it is obvious that $Z[l] = X[i] = Y[j]$ and $Z[1..l-1] \in S_{i-1,j-1,k}$.

(4) Since $Z[l] \neq X[i]$, $Z[1..l]$ is a common subsequence of $X[1..i-1]$ and $Y[1..j]$ including $P[1..k]$ as the suffix $Z[l-k+1..l]$. Similar to proof of 2, we have $Z[1..l] \in S_{i-1,j,k}$. The proof of Case (5) is similar to the proof of this case. □

Let $\mathcal{L}(i,j,k)$ denote the length of an LCS of $X[1..i]$ and $Y[1..j]$ including $P[1..k]$ as a suffix. By the optimal-substructure properties of the STR-IC-LCS problem shown in Theorem 1, we have the following recursive formula. For $0 < i \le m$, $0 < i \le n$, and $0 \le k \le \rho$,

$$\mathcal{L}(i,j,k) = \begin{cases} 1 + \mathcal{L}(i-1,j-1,k-1) & \text{if } k > 0 \text{ and } X[i] = Y[j] = P[k], \\ \mathcal{L}(i-1,j-1,k) & \text{if } k > 0,\ X[i] = Y[j] \text{ and } X[i] \neq P[k], \\ 1 + \mathcal{L}(i-1,j-1,k) & \text{if } k = 0 \text{ and } X[i] = Y[j], \\ \max\{\mathcal{L}(i-1,j,k), & \\ \quad \mathcal{L}(i,j-1,k)\} & \text{if } X[i] \neq Y[j]. \end{cases} \tag{1}$$

The boundary conditions of this recursive formula are $\mathcal{L}(i,0,0) = \mathcal{L}(0,j,0) = 0$ and $\mathcal{L}(0,j,k) = \mathcal{L}(i,0,k) = -\infty$ for any $0 \le i \le m$, $0 \le j \le n$, and $1 \le k \le \rho$. Based on (1), $\mathcal{L}$ is computed.

Let $\mathcal{C}(i,j)$ denote the length of an LCS of $X[i..m]$ and $Y[j..n]$ for $1 \le i \le m$ and $1 \le j \le n$. If $i = m+1$ or $j = n+1$, we set $\mathcal{C}(i,j) = 0$. $\mathcal{C}$ is computed by employing an $O(mn)$-time algorithm for the LCS problem (Cormen et al. 2001).

Let $Z$ be an LCS of $X$ and $Y$ including $P$ as a substring, and initially be an empty sequence. We define $\mathcal{T}(i,j) = \mathcal{L}(i,j,\rho) + \mathcal{C}(i+1,j+1)$ for $1 \le i \le m$ and $1 \le j \le n$. According to Property 1, the length of $Z$ is given by the maximum value of $\mathcal{T}$. Suppose that the maximum value of $\mathcal{T}$ is supplied from the entry $\mathcal{T}(i^*, j^*)$ for some $1 \le i^* \le m$ and $1 \le j^* \le n$. Let $Z_1$ be an LCS of $X[1..i^*]$ and $Y[1..j^*]$ containing $P$ as a suffix, and $Z_2$ be an LCS of $X[i^*+1..m]$ and $Y[j^*+1..n]$. We construct $Z_1$ and $Z_2$ by backtracking through the computation paths from $\mathcal{L}(i^*, j^*, \rho)$ to $\mathcal{L}(0,0,0)$ and from $\mathcal{C}(i^*+1, j^*+1)$ to $\mathcal{C}(m+1, n+1)$, respectively. Figure 3 illustrates the concept of constructing $Z_1$ and $Z_2$. Finally, we concatenate $Z_1$ and $Z_2$ to obtain $Z$.

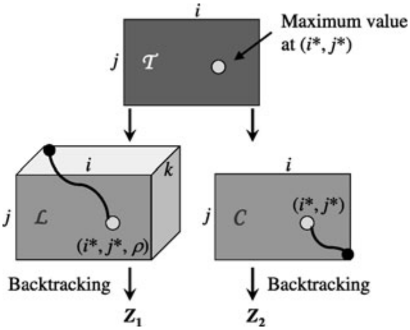


**Fig. 3** An illustration of finding solutions to two subproblems of the STR-IC-LCS problem. The maximum value $\mathcal{T}(i^*, j^*)$ is the length of an LCS of $X$ and $Y$ including $P$ as a substring. We construct an LCS $Z_1$ of $X[1..i^*]$ and $Y[1..j^*]$ including $P$ as a suffix by backtracking through the computation path from $\mathcal{L}(i^*, j^*, \rho)$ to $\mathcal{L}(0, 0, 0)$. An LCS $Z_2$ of $X[i^*+1..m]$ and $Y[j^*+1..n]$ is constructed by backtracking through the computation path from $\mathcal{C}(i^*+1, j^*+1)$ to $\mathcal{C}(m, n)$

Recovering the computation paths of $Z_1$ and $Z_2$ take $O(m+n+\rho)$ and $O(m+n)$ steps, respectively. Consequently, we solve the STR-IC-LCS problem in $O(mn\rho)$ time and space.

### 2.2 An $O(mn\rho)$-time algorithm for the SEQ-EC-LCS problem

The SEQ-EC-LCS problem is to find an LCS of two sequences $X$ and $Y$ *excluding* a constrained pattern $P$ as a *subsequence*. Theorem 2 decomposes the structure of an optimal solution based on the solutions to its smaller subproblems.

**Theorem 2** *Let $S_{i,j,k}$ denote the set of all LCSs of $X[1..i]$ and $Y[1..j]$ excluding $P[1..k]$ as a subsequence. If $Z[1..l] \in S_{i,j,k}$, the following conditions hold*:

(1) *If $X[i] = Y[j] = P[k]$ and $k = 1$, then $Z[l] \neq X[i]$ and $Z[1..l] \in S_{i-1,j-1,k}$.*
(2) *If $X[i] = Y[j] = P[k]$ and $k \geq 2$, then $Z[l] = X[i] = Y[j] = P[k]$ implies $Z[1..l-1] \in S_{i-1,j-1,k-1}$.*
(3) *If $X[i] = Y[j] = P[k]$ and $k \geq 2$, then $Z[l] \neq X[i]$ implies $Z[1..l] \in S_{i-1,j-1,k}$.*
(4) *If $X[i] = Y[j]$ and ($X[i] \neq P[k]$ and $k > 0$, or $k = 0$), then $Z[l] = X[i] = Y[j]$ and $Z[1..l-1] \in S_{i-1,j-1,k}$.*
(5) *If $X[i] \neq Y[j]$, then $Z[l] \neq X[i]$ implies $Z[1..l] \in S_{i-1,j,k}$.*
(6) *If $X[i] \neq Y[j]$, then $Z[l] \neq Y[j]$ implies $Z[1..l] \in S_{i,j-1,k}$.*

*Proof* The proof is similar to Theorem 1. □

Let $\mathcal{L}(i, j, k)$ denote the length of an LCS of $X[1..i]$ and $Y[1..j]$ excluding $P[1..k]$ as a subsequence. By the optimal-substructure properties of the SEQ-EC-LCS problem shown in Theorem 2, we have the following recursive formula. For any $0 < i \leq m$, $0 < j \leq n$, and $0 \leq k \leq \rho$,

$$
\mathcal{L}(i,j,k) = \begin{cases} \mathcal{L}(i-1,j-1,k) & \text{if } k=1 \text{ and } X[i]=Y[j]=P[k], \\ \max\{\mathcal{L}(i-1,j-1,k), \\ \quad 1+\mathcal{L}(i-1,j-1,k-1)\} & \text{if } k>2 \text{ and } X[i]=Y[j]=P[k], \\ 1+\mathcal{L}(i-1,j-1,k) & \text{if } X[i]=Y[j] \text{ and} \\ & \quad (k=0, \text{ or } k>0 \text{ and } X[i]\neq P[k]), \\ \max\{\mathcal{L}(i-1,j,k), \\ \quad \mathcal{L}(i,j-1,k)\} & \text{if } X[i]\neq Y[j]. \end{cases} \tag{2}
$$

The boundary conditions of this recursive formula are $\mathcal{L}(i,0,k)=\mathcal{L}(0,j,k)=0$ for any $0\le i\le m$, $0\le j\le n$, and $0\le k\le \rho$. Based on (2), $\mathcal{L}$ is computed.

Let $Z$ be an LCS of $X$ and $Y$ excluding $P$ as a subsequence, and initially be an empty sequence. The length of $Z$ is given by $\mathcal{L}(m,n,\rho)$. Thus, $Z$ is constructed by backtracking through the computation path from $\mathcal{L}(m,n,\rho)$ to $\mathcal{L}(0,0,0)$. Recovering the computation path of an LCS takes $O(m+n+\rho)$ steps. Consequently, we solve the SEQ-EC-LCS problem in $O(mn\rho)$ time and space.

### 2.3 An $O(mn\rho)$-time algorithm for the STR-EC-LCS problem

The STR-EC-LCS problem is to find an LCS of two sequences $X$ and $Y$ *excluding* a constrained pattern $P$ as a *substring*. Theorem 3 decomposes the structure of an optimal solution based on the solutions to its smaller subproblems.

**Theorem 3** *Let $S_{i,j,k}$ denote the set of all LCSs of $X[1..i]$ and $Y[1..j]$ excluding $P[1..k]$ as a substring. If $Z[1..l]\in S_{i,j,k}$, the following conditions hold:*

(1) *If $X[i]=Y[j]=P[k]$ and $k=1$, then $Z[l]\neq X[i]$ and $Z[1..l]\in S_{i-1,j-1,k}$.*
(2) *If $X[i]=Y[j]=P[k]$ and $k\ge 2$, then $Z[l]=X[i]=Y[j]=P[k]$ and $Z[l-1]=P[k-1]$ implies $Z[1..l-1]\in S_{i-1,j-1,k-1}$.*
(3) *If $X[i]=Y[j]=P[k]$ and $k\ge 2$, then $Z[l]=X[i]=Y[j]=P[k]$ and $Z[l-1]\neq P[k-1]$ implies $Z[1..l-1]\in S_{i-1,j-1,k}$.*
(4) *If $X[i]=Y[j]=P[k]$ and $k\ge 2$, then $Z[l]\neq X[i]$ implies $Z[1..l]\in S_{i-1,j-1,k}$.*
(5) *If $X[i]=Y[j]$ and $X[i]\neq P[k]$, then $Z[l]=X[i]=Y[j]$ and $Z[1..l-1]\in S_{i-1,j-1,k}$.*
(6) *If $X[i]\neq Y[j]$, then $Z[l]\neq X[i]$ implies $Z[1..l]\in S_{i-1,j,k}$.*
(7) *If $X[i]\neq Y[j]$, then $Z[l]\neq Y[j]$ implies $Z[1..l]\in S_{i,j-1,k}$.*

*Proof* The proof is similar to Theorem 1. □

Let $\mathcal{L}(i,j,k)$ denote the length of an LCS of $X[1..i]$ and $Y[1..j]$ excluding $P[1..k]$ as a substring. By the optimal-substructure properties of the STR-EC-LCS problem shown in Theorem 3, we have the following recursive formula. For any $0<i\le m$, $0<j\le n$, and $0\le k\le \rho$,

$$\mathcal{L}(i,j,k) = \begin{cases} \mathcal{L}(i-1,j-1,k) & \text{if } k=1 \text{ and } X[i]=Y[j]=P[k],\\ \max\{1+\mathcal{L}(i-1,j-1,k-1)), & \\ \quad 1+\mathcal{L}(i-1,j-1,k)\} & \text{if } k\geq 2 \text{ and } X[i]=Y[j]=P[k],\\ 1+\mathcal{L}(i-1,j-1,k) & \text{if } X[i]=Y[j] \text{ and}\\ & \quad (k=0, \text{ or } k>0 \text{ and } X[i]\neq P[k]),\\ \max\{\mathcal{L}(i-1,j,k), & \\ \quad \mathcal{L}(i,j-1,k)\} & \text{if } X[i]\neq Y[j]. \end{cases} \tag{3}$$

The boundary conditions of this recursive formula are $\mathcal{L}(i,0,k)=\mathcal{L}(0,j,k)=0$ for any $0\leq i\leq m$, $0\leq j\leq n$, and $0\leq k\leq\rho$. Based on (3), $\mathcal{L}$ is computed.

Let $Z$ be an LCS of $X$ and $Y$ excluding $P$ as a substring, and initially be an empty sequence. The length of $Z$ is given by $\mathcal{L}(m,n,\rho)$. Therefore, $Z$ is constructed by backtracking through the computation path from $\mathcal{L}(m,n,\rho)$ to $\mathcal{L}(0,0,0)$. Recovering the computation path of an LCS takes $O(m+n+\rho)$ steps. Consequently, we solve the STR-EC-LCS problem in $O(mn\rho)$ time and space.

## 3 The GC-LCS problems with an arbitrary number of constrained patterns

In this section, we consider the GC-LCS problems whose inputs are two sequences $X$, $Y$ and $w$ constrained patterns $P_1,\ldots$, and $P_w$ of lengths $m$, $n$, $\rho_1,\ldots$, and $\rho_w$, respectively. Gotthilf et al. (2008) showed that the SEQ-IC-LCS problem with multiple constrained patterns is NP-hard and does not have a polynomial-time approximation scheme (PTAS). In fact, one can further show that the STR-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems with multiple constrained patterns are also NP-hard.

We solve the SEQ-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems with an arbitrary number of constrained patterns by the following approach, which is similar to the methods for the problems with single constrained pattern. An optimal-substructure property for the problem is first given, and a recurrence formula is derived based on this property. We then apply a tabular method to compute the length of an LCS of $X$ and $Y$ including each pattern in $O(mn\times\prod_{k=1}^{w}\rho_k)$ time, and construct an LCS by backtracking through the computation path in $O(m+n+\sum_{k=1}^{w}\rho_k)$ steps. Therefore, the procedure for obtaining an LCS takes $O(mn\times\prod_{k=1}^{w}\rho_k)$ time and space.

The approach to the STR-IC-LCS problem with more than one constrained pattern, however, is quite different from the method for the problem with single constrained pattern. Here we only investigate the STR-IC-LCS problem with two constrained patterns.

Property 2 gives the characterization of the structure of a solution for the STR-IC-LCS problem with two constrained patterns.

*Property 2* If $Z[1..l]$ is an LCS of $X[1..m]$ and $Y[1..n]$ including $P_1$ and $P_2$ as substrings, and assume that $P_2$ is the latter substring $Z[l'-\rho_2+1..l']$ (the case of $P_1$ being the latter substring is similar) for some $\rho_2\leq l'\leq l$, then $Z[1..l]$ is a concatenation of the following two substrings, for some $1\leq i\leq m$ and $1\leq j\leq n$:

1. The prefix $Z[1..l']$: $Z[1..l']$ is an LCS of $X[1..i]$ and $Y[1..j]$ including $P_1$ and $P_2$ as substrings, where $P_2$ is the suffix $Z[l' - \rho_2 + 1..l']$, and
2. The suffix $Z[l' + 1..l]$: $Z[l' + 1..l]$ is an LCS of $X[i + 1..m]$ and $Y[j + 1..n]$.

Based on Property 2, we first compute an LCS of $X[1..i]$ and $Y[1..j]$ including $P_1$ as a substring and $P_2$ as a suffix, and an LCS of $X[i..m]$ and $Y[j..n]$ for all $1 \le i \le m$ and $1 \le j \le n$. The solutions to the two subproblems are then merged to determine a longest concatenation. The length of an LCS of $X[i..m]$ and $Y[j..n]$ is computed by applying a quadratic-time algorithm (Cormen et al. 2001) for all $1 \le i \le m$ and $1 \le j \le n$.

For obtaining an LCS $Z[1..l']$ of $X[1..i]$ and $Y[1..j]$ including $P_1$ a substring and $P_2$ as the suffix $Z[l' - \rho_2 + 1..l']$ for all $1 \le i \le m$ and $1 \le j \le n$, we need to consider the following two cases:

(1) $P_2$ overlaps $P_1$: We merge $P_1$ and $P_2$ as a new pattern (there are $\min\{\rho_1, \rho_2\}$ concatenations of length at most $\rho_1 + \rho_2 - 1$), and then apply the algorithm for the STR-IC-LCS problem shown in Sect. 2.1 to solve it. For all $1 \le i \le m$ and $1 \le j \le n$, computing the length of an LCS of $X[1..i]$ and $Y[1..j]$ in this case takes $\Sigma_{k=0}^{\min\{\rho_1,\rho_2\}-1} O(mn \times (\max\{\rho_1, \rho_2\} + k))$ $(= O(mn\rho_1\rho_2))$ time and $O(mn \times (\rho_1 + \rho_2 - 1))$ $(= O(mn \times \max\{\rho_1, \rho_2\}))$ space.

(2) $P_2$ does not overlap $P_1$: $Z[1..l']$ is a concatenation of two substrings, which are an LCS $Z[1..l' - \rho_2]$ of $X[1..i']$ and $Y[1..j']$ including $P_1$ as a substring and an LCS $Z[l' - \rho_2 + 1..l'](= P_2)$ of $X[i' + 1..i]$ and $Y[j' + 1..j]$, respectively, for some $1 \le i' < i$ and $1 \le j' < j$. Let $\mathcal{L}_1(i, j)$ denote the length of an LCS of $X[1..i]$ and $Y[1..j]$ including $P_1$ as a substring, and $\mathcal{L}_2(i, j)$ denote the length of an LCS of $X[1..i]$ and $Y[1..j]$ including $P_2$ as a suffix. $\mathcal{L}_1$ and $\mathcal{L}_2$ are computed by the algorithms shown in Sect. 2.1. We use a 2-dimension matrix where $\sigma(i, j)$ records the pair $(i', j')$ such that $P_2$ is exactly an LCS of $X[i' + 1..i]$ and $Y[j' + 1..j]$ for all $0 \le i \le m$ and $0 \le j \le n$. The matrix $\sigma$ is computed in $O(mn\rho_2)$ time. The length of an LCS of $X[1..i]$ and $Y[1..j]$ in this case equals to $\mathcal{L}_1(i', j') + \mathcal{L}_2(i, j) - \mathcal{L}_2(i', j')$. For all $1 \le i \le m$ and $1 \le j \le n$, computing $\mathcal{L}_1$ and $\mathcal{L}_2$ take $O(mn \times \max\{\rho_1, \rho_2\})$ time and space, and calculating the length of an LCS of $X[1..i]$ and $Y[1..j]$ in this case takes $O(mn \times \max\{\rho_1, \rho_2\})$ time and space.

Finally, we concatenate an LCS of $X[i + 1..m]$ and $Y[j + 1..n]$ and an LCS of $X[1..i]$ and $Y[1..j]$ including $P_1$ as a substring and $P_2$ as a suffix, for all $1 \le i \le m$ and $1 \le j \le n$. The concatenation of maximum length is an LCS of $X[1..m]$ and $Y[1..n]$ including $P_1$ and $P_2$ as substrings. Computing the length of all concatenation takes $O(mn\rho_1\rho_2)$ time and $O(mn \times \max\{\rho_1, \rho_2\})$ space. Constructing an LCS by backtracking and concatenating takes $O(m + n + \rho_1 + \rho_2)$ steps. Consequently, we solve the STR-IC-LCS problem with two constrained patterns in $O(mn\rho_1\rho_2)$ time and $O(mn \times \max\{\rho_1, \rho_2\})$ space.

## 4 Concluding remarks

In this paper, we present three $O(mn\rho)$-time and $O(mn\rho)$-space algorithms for solving the STR-IC-LCS, SEQ-EC-LCS, and STR-EC-LCS problems, where $m$, $n$, and $\rho$ are the lengths of two sequences and a constrained pattern, respectively. In fact, the space requirement can be further reduced to $O(\rho \times (m+n))$ by applying Hirschberg's approach (Hirschberg 1975). We also consider the GC-LCS problems with an arbitrary number of constrained patterns.

## References

Aho AV, Hirschberg DS, Ullman JD (1976) Bounds on the complexity of the longest common subsequence problem. J ACM 23:1–12

Apostolico A, Guerra C (1987) The longest common subsequence problem revisited. Algorithmica 2:315–336

Arslan AN, Eğecioğlu O (2005) Algorithms for the constrained longest common subsequence problems. Int J Found Comput Sci 16(6):1099–1109

Bergroth L, Hakonen H, Raita T (2000) A survey of longest common subsequence algorithms. In: Proceedings of the 7th international symposium on string processing and information retrieval (SPIRE'00), pp 39–48

Bonizzoni P, Vedova GD, Dondi R, Fertin G, Rizzi R, Vialette S (2007) Exemplar longest common subsequence. IEEE Trans Comput Biol Bioinform 4(4):535–543

Chao KM, Zhang L (2009) Sequence comparison: theory and methods. Springer, Berlin

Chin FYL, Santis AD, Ferrara AL, Ho NL, Kim SK (2004) A simple algorithm for the constrained longest common sequence problems. Inf Process Lett 90:175–179

Chin FYL, Ho NL, Lam TW, Wong PWH (2005) Efficient constrained multiple sequence alignment with performance guarantee. J Bioinform Comput Biol 3(1):1–18

Cormen TH, Leiserson CE, Rivest RL, Stein C (2001). In: Introduction to algorithms, 2nd edn. MIT Press/McGraw-Hill, New York, pp 350–355.

Gotthilf Z, Hermelin D, Lewenstein M (2008) Constrained LCS: hardness and approximation. In: Proceedings of the 19th annual symposium on combinatorial pattern matching (CPM'08), pp 255–262

Gusfield D (1997) Algorithms on strings, trees, and sequences. Cambridge University Press, Cambridge

Hirschberg DS (1975) A linear space algorithm for computing maximal common subsequences. Commun ACM 18:341–343

Hirschberg DS (1977) Algorithms for the longest common subsequence problem. J ACM 24:664–675

Hunt JW, Szymanski TG (1977) A fast algorithm for computing longest common subsequence. Commun ACM 20(5):350–353

Iliopoulos CS, Rahman MS (2008) New efficient algorithms for the LCS and constrained LCS problems. Inf Process Lett 106:13–18

Maier D (1978) The complexity of some problems on subsequences and supersequence. J ACM 25:322–336

Masek WJ, Paterson MS (1980) A faster algorithm computing string edit distances. J Comput Syst Sci 20:18–31

Pevzner PA (2000) Computational molecular biology: An algorithmic approach. MIT Press, Cambridge

Rahman MS, Iliopoulos CS (2007) A new efficient algorithm for computing the longest common subsequence. In: Proceedings of the 3rd international conference on algorithmic aspects in information and management (AAIM'07), pp 82–90

Tang CY, Lu CL, Chang MD, Tsai YT, Sun YJ, Chao KM, Chang JM, Chiou YH, Wu CM, Chang HT, Chou WI (2003) Constrained multiple sequence alignment tool development and its application to RNase family alignment. J Bioinform Comput Biol 1(2):267–287

Tsai YT (2003) The constrained longest common subsequence problem. Inf Process Lett 88:173–176

van Emde Boas P (1977) Preserving order in a forest in less than logarithmic time and linear space. Inf Process Lett 6:80–82

van Emde Boas P, Kaas R, Zijlstra E (1977) Design and implementation of an efficient priority queue. Math Syst Theory 10:99–127

Wagner RA, Fischer MJ (1974) The string-to-string correction problem. J ACM 21(1):168–173