Computing SEQ-IC-LCS of Labeled Graphs

Yuki Yonemoto[1], Yuto Nakashima[2], and Shunsuke Inenaga[2]

[1] Department of Information Science and Technology, Kyushu University
yonemoto.yuuki.240@s.kyushu-u.ac.jp
[2] Department of Informatics, Kyushu University
{nakashima.yuto.003, inenaga.shunsuke.380}@m.kyushu-u.ac.jp

Abstract. We consider labeled directed graphs where each vertex is labeled with a non-empty string. Such labeled graphs are also known as non-linear texts in the literature. In this paper, we introduce a new problem of comparing two given labeled graphs, called the SEQ-IC-LCS problem on labeled graphs. The goal of SEQ-IC-LCS is to compute the the length of the longest common subsequence (LCS) Z of two target labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ that includes some string in the constraint labeled graph $G_3 = (V_3, E_3)$ as its subsequence. Firstly, we consider the case where G_1, G_2 and G_3 are all acyclic, and present algorithms for computing their SEQ-IC-LCS in $O(|E_1||E_2||E_3|)$ time and $O(|V_1||V_2||V_3|)$ space. Secondly, we consider the case where G_1 and G_2 can be cyclic and G_3 is acyclic, and present algorithms for computing their SEQ-IC-LCS in $O(|E_1||E_2||E_3| + |V_1||V_2||V_3| \log |\Sigma|)$ time and $O(|V_1||V_2||V_3|)$ space, where Σ is the alphabet.

1 Introduction

We consider *labeled (directed) graphs* where each vertex is labeled with a non-empty string. Such labeled graphs are also known as *non-linear texts* or *hypertexts* in the literature. Labeled graphs are a natural generalization of usual (unary-path) strings, which can also be regarded as a compact representation of a set of strings. After introduced by the Database community [13], labeled graphs were then considered by the string matching community [21,23,2,22,16,17,10]. Recently, graph representations of large-scale string sets appear in the real-world applications including graph databases [3] and pan-genomics [14]. For instance, *elastic degenerate strings* [18,4,8,19,7], which recently gain attention with bioinformatics background, can be regarded as a special case of labeled graphs. In the best case, a single labeled graph can represent exponentially many strings. Thus, efficient string algorithms that directly work on labeled graphs without expansion are of significance both in theory and in practice.

Shimohira et al. [24] introduced the problem of computing the *longest common subsequence* (*LCS*) of two given labeled graphs, which, to our knowledge, the first and the only known similarity measure of labeled graphs. Since we can easily convert any labeled graph with string labels to an equivalent labeled graph with single character labels (see Figure 1), in what follows, we evaluate the size of a labeled graph by the number of vertices and edges in the (converted) graph. Given two labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, Shimohira et al. [24] showed how to solve the LCS problem on labeled graphs in $O(|E_1||E_2|)$ time and $O(|V_1||V_2|)$ space when both G_1 and G_2 are acyclic, and in $O(|E_1||E_2| + |V_1||V_2| \log |\Sigma|)$ time and $O(|V_1||V_2|)$ space when G_1 and G_2 can be cyclic, where Σ is the alphabet. It is noteworthy that their solution is almost optimal since the quadratic $O((|A||B|)^{1-\epsilon})$-time conditional lower bound [1,9] with any constant $\epsilon > 0$ for the LCS problem on two strings A, B also applies to the LCS problem on labeled graphs.

Yuki Yonemoto, Yuto Nakashima, Shunsuke Inenaga: Computing SEQ-IC-LCS of Labeled Graphs, pp. 3–17.
Proceedings of PSC 2023, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-07206-6

The *constrained LCS problems* on strings, which were first proposed by Tsai [25] and then extensively studied in the literature [25,12,6,11,15,27,28], use a third input string P which introduces a-priori knowledge of the user to the solution string Z to output. The task here is to compute the longest common subsequence Z of two target strings A and B that meets the condition w.r.t. P, such that

STR-IC-LCS: Z includes (contains) P as substring;
STR-EC-LCS: Z excludes (does not contain) P as substring;
SEQ-IC-LCS: Z includes (contains) P as subsequence;
SEQ-EC-LCS: Z excludes (does not contain) P as subsequence.

While STR-IC-LCS can be solved in $O(|A||B|)$ time [15], the state-of-the-art solutions to STR-EC-LCS and SEQ-IC/EC-LCS run in $O(|A||B||P|)$ time [12,6,11,27].

In this paper, we consider the SEQ-IC-LCS problems on labeled graphs, where the inputs are two target labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, and a constraint text $G_3 = (V_3, E_3)$, and the output is (the length of) a longest common subsequence of G_1 and G_2 such that Z includes as subsequence some string that is represented by G_3. Firstly, we consider the case where G_1, G_2 and G_3 are all acyclic, and present algorithms for computing their SEQ-IC-LCS in $O(|E_1||E_2||E_3|)$ time and $O(|V_1||V_2||V_3|)$ space. Secondly, we consider the case where G_1 and G_2 can be cyclic and G_3 is acyclic, and present algorithms for computing their SEQ-IC-LCS in $O(|E_1||E_2||E_3| + |V_1||V_2||V_3| \log |\Sigma|)$ time and $O(|V_1||V_2||V_3|)$ space, where Σ is the alphabet. The time complexities of our algorithms and related work are summarized in Table 1. Our algorithms for solving SEQ-IC-LCS on labeled graphs are based on the solutions to SEQ-IC-LCS of usual strings proposed by Chin et al. [12]. We emphasize that a faster $o(|E_1||E_2||E_3|)$-time solution to the SEQ-IC-LCS problems implies a major improvement over the SEQ-IC-LCS problems for strings whose best known solutions require cubic time.

A related work is the *regular language constrained sequence alignment* (*RLCSA*) problem [5] for two input strings A and B in which the constraint is given as an NFA. It is known that this problem can be solved in $O(|A||B||V|^3/\log |V|)$ time [20], where $|V|$ denotes the number of states in the NFA.

problem	text-1	text-2	text-3	time complexity	
LCS	string	string	-	$O(\|E_1\|\|E_2\|)$	[26]
	DAG	DAG	-	$O(\|E_1\|\|E_2\|)$	[24]
	graph	graph	-	$O(\|E_1\|\|E_2\| + \|V_1\|\|V_2\| \log \|\Sigma\|)$	[24]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$	[12,6]
	DAG	DAG	DAG	$O(\|E_1\|\|E_2\|\|E_3\|)$	[this work]
	graph	graph	DAG	$O(\|E_1\|\|E_2\|\|E_3\| + \|V_1\|\|V_2\|\|V_3\| \log \|\Sigma\|)$	[this work]
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$	[11]
STR-IC-LCS	string	string	-	$O(\|E_1\|\|E_2\|)$	[15]
STR-EC-LCS	string	string	-	$O(\|E_1\|\|E_2\|)$	[27]
RLCSA	string	string	NFA	$O(\|E_1\|\|E_2\|\|V_3\|^3/\log \|V_3\|)$	[20]

Table 1. Time complexities of algorithms for labeled graph/usual string comparisons, for inputs text-1 $G_1 = (V_1, E_1)$, text-2 $G_2 = (V_2, E_2)$, and text-3 $G_3 = (V_3, E_3)$. Here, a string input of length n is regarded as a unary path graph $G = (V, E)$ with $|E| = n$.

2 Preliminaries

2.1 Strings and Graphs

Let Σ be an alphabet. An element of Σ^* is called a *string*. The *length* of a string w is denoted by $|w|$. The *empty string*, denoted by ε, is a string of length 0. Let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For a string $w = xyz$ with $x, y, z \in \Sigma^*$, strings x, y, and z are called a *prefix*, *substring*, and *suffix* of string w, respectively. The ith character of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$, and the substring of w that begins at position i and ends at position j is denoted by $w[i..j]$ for $1 \leq i \leq j \leq |w|$. For convenience, let $w[i..j] = \varepsilon$ for $i > j$. A string u is a *subsequence* of another string w if $u = \varepsilon$ or there exists a sequence of integers $i_1, \ldots, i_{|u|}$ such that $1 \leq i_1 < \cdots < i_{|u|} \leq |w|$ and $u = w[i_1] \cdots w[i_{|u|}]$.

A *directed graph* G is an ordered pair (V, E) of the set V of *vertices* and the set $E \subseteq V \times V$ of *edges*. The *in-degree* of a vertex v is denoted by $\mathsf{in\_deg}(v) = |\{u \mid (u, v) \in E\}|$. A *path* in a (directed) graph $G = (V, E)$ is a sequence $v_0, \ldots, v_k$ of vertices such that $(v_{i-1}, v_i) \in E$ for every $i = 1, \ldots, k$. A path $\pi = v_0, \ldots, v_k$ in graph G is said to be *left-maximal* if its left-end vertex v_0 has no in-coming edges, and π is said to be *right-maximal* if its right-end vertex v_k has no out-going edges. A path π is said to be *maximal* if π is both left-maximal and right-maximal. For any vertex $v \in V$, let $\mathsf{P}(v)$ denote the set of all paths ending at vertex v, and $\mathsf{LMP}(v)$ denote the set of left-maximal paths ending at v. The set of all paths in $G = (V, E)$ is denoted by $\mathsf{P}(G) = \{\mathsf{P}(v) \mid v \in V\}$. Let $\mathsf{MP}(G)$ denote the set of maximal paths in G.

2.2 Longest Common Subsequence (LCS) of Strings

The *longest common subsequence* (LCS) problem for two given strings A and B is to compute (the length of) the longest string Z that is a subsequences of both A and B. It is well-known that LCS can be solved in $O(|A||B|)$ time by using the following recurrence [26]:

$$C_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0; \\ 1 + C_{i-1,j-1} & \text{if } i, j > 0 \text{ and } x[i] = y[j]; \\ \max(C_{i-1,j}, C_{i,j-1}) & \text{if } i, j > 0 \text{ and } x[i] \neq y[j], \end{cases}$$

where $C_{i,j}$ is the LCS length of $A[1..i]$ and $B[1..j]$.

2.3 SEQ-IC-LCS of Strings

Let A, B, and P be strings. A string Z is said to be an *SEQ-IC-LCS* of two target strings A and B *including* the pattern P if Z is a longest string such that P is a subsequence of Z and that Z is a common subsequence of A and B. Chin et al. [12] solved this problem in $O(|A||B||P|)$ time by using the following recurrence:

$$C_{i,j,k} = \begin{cases} 0 & \text{if } k = 0 \text{ and } (i = 0 \text{ or } j = 0); \\ -\infty & \text{if } k \neq 0 \text{ and } (i = 0 \text{ or } j = 0); \\ C_{i-1,j-1,k-1} + 1 & \text{if } i, j, k > 0 \text{ and } A[i] = B[j] = P[k]; \\ C_{i-1,j-1,k} + 1 & \text{if } i, j > 0 \text{ and } A[i] = B[j] \neq P[k]; \\ \max(C_{i-1,j,k}, C_{i,j-1,k}) & \text{if } i, j > 0 \text{ and } A[i] \neq B[j], \end{cases} \quad (1)$$

where $C_{i,j,k}$ is the SEQ-IC-LCS length of $A[1..i]$, $B[1..j]$, and $P[1..k]$.

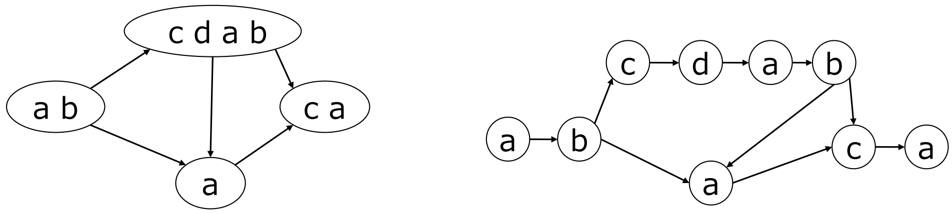

Figure 1. A labeled graph $G = (V, E, L)$ with $L : V \to \Sigma^+$ and its corresponding atomic labeled graph $G' = (V', E', L')$ with $L' : V' \to \Sigma$.

2.4 Labeled Graphs

A *labeled graph* is a directed graph with vertices labeled by strings, namely, it is a directed graph $G = (V, E, L)$ where V is the set of vertices, E is the set of edges, and $L : V \to \Sigma^+$ is a labeling function that maps nodes $v \in V$ to non-empty strings $L(v) \in \Sigma^+$. For a path $\pi = v_0, \ldots, v_k \in \mathsf{P}(G)$, let $L(\pi)$ denote the string spelled out by w, namely $L(\pi) = L(v_0) \cdots L(v_k)$. The size $|G|$ of a labeled graph $G = (V, E, L)$ is $|V| + |E| + \sum_{v \in V} |L(v)|$. Let $\mathsf{Subseq}(G) = \{\mathsf{Subseq}(L(\pi)) \mid \pi \in \mathsf{P}(G)\}$ denote the set of subsequences of a labeled graph $G = (V, E, L)$. For a set $P \in \mathsf{P}(G)$ of paths in G, let $L(P) = \{L(\pi) \mid \pi \in P\}$ denote the set of string labels for the paths in P.

For a labeled graph $G = (V, E, L)$, consider an "atomic" labeled graph $G' = (V', E', L')$ such that $L' : V' \to \Sigma$,

$$V' = \{v_{i,j} \mid L'(v_{i,j}) = L(v_i)[j], v_i \in V, 1 \leq j \leq |L(v_i)|\}, \text{ and}$$
$$E' = \{(v_{i,|L(v_i)|}, v_{k,1}) \mid (v_i, v_k) \in E\} \cup \{(v_{i,j}, v_{i,j+1}) \mid v_i \in V, 1 \leq j < |L(v_i)|\},$$

that is, G' is a labeled graph with each vertex being labeled by a single character, which represents the same set of strings as G. An example is shown in Figure 1. Since $|V'| = \sum_{v \in V} |L(v)|$, $|E'| = |E| + \sum_{v \in V}(|L(v)| - 1)$, and $\sum_{v' \in V'} |L(v')| = \sum_{v \in V} |L(v)|$, we have $|G'| = O(|G|)$. We remark that given G, we can easily construct G' in $O(|G|)$ time. Observe that $\mathsf{Subseq}(G) = \mathsf{Subseq}(G')$ also holds.

In the sequel we only consider atomic labeled graphs where each vertex is labeled with a single character.

2.5 LCS of Acyclic Labeled Graphs

The problem of computing the length of longest common subsequence of two input acyclic labeled graphs is formalized by Shimohira et al. [24] as follows.

Problem 1 (Longest common subsequence problem for acyclic labeled graphs).

Input: Labeled graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$.
Output: The length of a longest string in $\mathsf{Subseq}(G_1) \cap \mathsf{Subseq}(G_2)$.

This problem can be solved in $O(|E_1||E_2|)$ time and $O(|V_1||V_2|)$ space by sorting G_1 and G_2 topologically and using the following recurrence:

$$C'_{i,j} = \begin{cases} 1 + \max(\{C'_{k,\ell} \mid (v_{1,k}, v_{1,i}) \in E_1, (v_{2,\ell}, v_{2,j}) \in E_2\} \cup \{0\}) & \text{if } L_1(v_{1,i}) = L_2(v_{2,j}); \\ \max\left(\begin{array}{l}\{C'_{k,j} \mid (v_{1,k}, v_{1,i}) \in E_1\} \cup \\ \{C'_{i,\ell} \mid (v_{2,\ell}, v_{2,j}) \in E_2\} \cup \{0\}\end{array}\right) & \text{otherwise,} \end{cases} \quad (2)$$

where $v_{1,i}$ and $v_{2,j}$ are respectively the ith and jth vertices of G_1 and in G_2 in topological order, for $1 \leq i \leq |V_1|$ and $1 \leq j \leq |V_2|$, and $C'_{i,j}$ is the length of a longest string in $\mathsf{Subseq}(L_1(\mathsf{P}(v_{1,i}))) \cap \mathsf{Subseq}(L_2(\mathsf{P}(v_{2,j})))$.

2.6 LCS of Cyclic Labeled Graphs

Here we consider a generalized version of Problem 1 where the input labeled graphs G_1 and/or G_2 can be cyclic. In this problem, the output is ∞ if there is a string $s \in \mathsf{Subseq}(G_1) \cap \mathsf{Subseq}(G_2)$ such that $|s| = \infty$, and that is the length of a longest string in $\mathsf{Subseq}(G_1) \cap \mathsf{Subseq}(G_2)$. Shimohira et al. [24] proposed an $O(|E_1||E_2| + |V_1||V_2| \log |\Sigma|)$ time and $O(|V_1||V_2|)$ space algorithm solving this problem. Their algorithm judges whether the output is ∞ by using a balanced tree, and computes the length of the solution by using Equation (2) and the balanced tree if the output is not ∞.

3 The SEQ-IC-LCS Problem for Labeled Graphs

In this paper, we tackle the problem of computing the SEQ-IC-LCS length of three labeled graphs, which formalized as follows:

Problem 2 (SEQ-IC-LCS problem for labeled graphs).

Input: Labeled graphs $G_1 = (V_1, E_1, L_1)$, $G_2 = (V_2, E_2, L_2)$, and $G_3 = (V_3, E_3, L_3)$.
Output: The length of a longest string in the set
$\{z \mid \exists\, q \in L_3(\mathsf{MP}(G_3))$ such that $q \in \mathsf{Subseq}(z)$ and $z \in \mathsf{Subseq}(G_1) \cap \mathsf{Subseq}(G_2)\}$.

Intuitively, Problem 2 asks to compute a longest string z such that z is a subsequence occurring in both G_1 and G_2 and that there exists a string q which corresponds to a maximal path of G_3 and is a subsequence of z.

For a concrete example, see the labeled graphs G_1, G_2 and G_3 of Figure 2. String `cdba` is a common subsequence of G_1 and G_2 and that contains an element `ba` of a maximal path string in $L_3(\mathsf{MP}(G_3))$. Since `cdba` is such a longest string, we ouput the SEQ-IC-LCS length $|\mathtt{cdba}| = 4$ as the solution to this instance.

In the sequel, Section 4 presents our solution to the case where the all input labeled graphs are acyclic, and Section 5 presents our solutions case where G_1 and/or G_2 can be cyclic and G_3 is acyclic.

4 Computing SEQ-IC-LCS of Acyclic Labeled Graphs

In this section, we present our algorithm which solves Problem 2 in the case where all of G_1, G_2 and G_3 are acyclic. The following is our result:

Theorem 3. *Problem 2 with acyclic labeled graphs G_1, G_2 and G_3 can be solved in $O(|E_1||E_2||E_3|)$ time and $O(|V_1||V_2||V_3|)$ space.*

Proof. We perform topological sort to the vertices of G_1, G_2, and G_3 in $O(|E_1| + |E_2| + |E_3|)$ time and $O(|V_1| + |V_2| + |V_3|)$ space. For $1 \leq i \leq |V_1|$, $1 \leq j \leq |V_2|$, and $1 \leq k \leq |V_3|$, let $v_{1,i}$, $v_{2,j}$, $v_{3,k}$ denote the ith, jth, and kth vertices in G_1, G_2, and G_3 in topological order, respectively. Let

$$\mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k}) = \left\{ z \;\middle|\; \begin{array}{l} \exists q \in L_3(\mathsf{LMP}(v_{3,k})) \text{ such that } q \in \mathsf{Subseq}(z) \\ \text{and } z \in \mathsf{Subseq}(L_1(\mathsf{P}(v_{1,i}))) \cap \mathsf{Subseq}(L_2(\mathsf{P}(v_{2,j}))) \end{array} \right\}$$

be the set of candidates of SEQ-IC-LCS strings for the maximal induced graphs of G_1, G_2, and G_3 whose sinks are $v_{1,i}$, $v_{2,j}$, and $v_{3,k}$, respectively. Let $D_{i,j,k}$ denote the length of a longest string in $\mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$. The solution to Problem 2 (the SEQ-IC-LCS length) is the maximum value of $D_{i,j,k}$ for which $v_{3,k}$ does not have out-going edges (i.e. $v_{3,k}$ is the end of a maximal path in G_3).

When $k = 0$, then the problem is equivalent to Problem 1 of computing SEQ-IC-LCS of strings. In that follows, we show how to compute $D_{i,j,k}$ for $k > 0$:

1. If $L_1(v_{1,i}) = L_2(v_{2,j}) = L_3(v_{3,k})$, there are three cases to consider:
   (a) If $v_{1,i}$ does not have in-coming edges or $v_{2,j}$ does not have in-coming edges, and if $v_{3,k}$ does not have in-coming edges (i.e., $\mathsf{in\_deg}(v_{1,i}) = \mathsf{in\_deg}(v_{3,k}) = 0$, or $\mathsf{in\_deg}(v_{2,j}) = \mathsf{in\_deg}(v_{3,k}) = 0$), then clearly $D_{i,j,k} = 1$.
   (b) If $v_{1,i}$ does not have in-coming edges or $v_{2,j}$ does not have in-coming edges, and if $v_{3,k}$ has some in-coming edge(s) (i.e., $\mathsf{in\_deg}(v_{1,i}) = 0$ and $\mathsf{in\_deg}(v_{3,k}) \geq 1$, or $\mathsf{in\_deg}(v_{2,j}) = 0$ and $\mathsf{in\_deg}(v_{3,k}) \geq 1$), then clearly $D_{i,j,k} = -\infty$.
   (c) If both $v_{1,i}$ and $v_{2,j}$ have some in-coming edge(s) and $v_{3,k}$ does not have in-coming edges (i.e., $\mathsf{in\_deg}(v_{1,i}) \geq 1$, $\mathsf{in\_deg}(v_{2,j}) \geq 1$, and $\mathsf{in\_deg}(v_{3,k}) = 0$), then let $v_{1,x}$ and $v_{2,y}$ be any nodes s.t. $(v_{1,x}, v_{1,i}) \in E_1$, and $(v_{2,y}, v_{2,j}) \in E_2$, respectively. Let s be a longest string in $\mathsf{Subseq}(L_1(\mathsf{P}(v_{1,i}))) \cap \mathsf{Subseq}(L_2(\mathsf{P}(v_{2,j})))$. Assume on the contrary that there exists a string $t \in \mathsf{Subseq}(L_1(\mathsf{P}(v_{1,x}))) \cap \mathsf{Subseq}(L_2(\mathsf{P}(v_{2,y})))$ such that $|t| > |s| - 1$. This contradicts that s is a longest common subsequence of $L_1(\mathsf{P}(v_{1,i}))$ and $L_2(\mathsf{P}(v_{2,j}))$, since $L_1(v_{1,i}) = L_2(v_{2,j})$. Hence $|t| \leq |s| - 1$. If $v_{1,x}$ and $v_{2,y}$ are vertices satisfying $C'_{x,y,0} = |s| - 1$, then $C'_{i,j,k} = C'_{x,y,0} + 1$. Note that such nodes $v_{1,x}$ and $v_{2,y}$ always exist.
   (d) Otherwise (all $v_{1,i}$, $v_{2,j}$, and $v_{3,k}$ have some in-coming edge(s)), let $v_{1,x}$, $v_{2,y}$ and $v_{3,z}$ be any nodes s.t. $(v_{1,x}, v_{1,i}) \in E_1$, $(v_{2,y}, v_{2,j}) \in E_2$ and $(v_{3,z}, v_{3,k}) \in E_3$, respectively. Let s be a longest string in $\mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$. Assume on the contrary that there exists a string $t \in \mathsf{S}_{\mathrm{IC}}(v_{1,x}, v_{2,y}, v_{3,z})$ such that $|t| > |s| - 1$. This contradicts that s is a SEQ-IC-LCS of $L_1(\mathsf{P}(v_{1,i}))$, $L_2(\mathsf{P}(v_{2,j}))$ and $L_3(\mathsf{LMP}(v_{3,k}))$, since $L_1(v_{1,i}) = L_2(v_{2,j}) = L_3(v_{3,k})$. Hence $|t| \leq |s| - 1$. If $v_{1,x}$, $v_{2,y}$ and $v_{3,z}$ are vertices satisfying $D_{x,y,z} = |s| - 1$, then $D_{i,j,k} = D_{x,y,z} + 1$. Note that such nodes $v_{1,x}$, $v_{2,y}$ and $v_{3,z}$ always exist.
2. If $L_1(v_{1,i}) = L_2(v_{2,j}) \neq L_3(v_{3,k})$, there are two cases to consider:
   (a) If $v_{1,i}$ does not have in-coming edges or $v_{2,j}$ does not have-incoming edges (i.e., $\mathsf{in\_deg}(v_{1,i}) = 0$ or $\mathsf{in\_deg}(v_{2,j}) = 0$), then clearly $D_{i,j,k}$ does not exist and let $D_{i,j,k} = -\infty$.
   (b) Otherwise (both $v_{1,i}$ and $v_{2,j}$ have in-coming edge(s)), let $v_{1,x}$ and $v_{2,y}$ be any nodes s.t. $(v_{1,x}, v_{1,i}) \in E_1$ and $(v_{2,y}, v_{2,j}) \in E_2$, respectively. Let s be a longest string in $\mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$. Assume on the contrary that there exists a string $t \in \mathsf{S}_{\mathrm{IC}}(v_{1,x}, v_{2,y}, v_{3,k})$ such that $|t| > |s| - 1$. This contradicts that s is a SEQ-IC-LCS of $L_1(\mathsf{P}(v_{1,i}))$, $L_2(\mathsf{P}(v_{2,j}))$ and $L_3(\mathsf{LMP}(v_{3,k}))$, since $L_1(v_{1,i}) = L_2(v_{2,j})$. Hence $|t| \leq |s| - 1$. If $v_{1,x}$, $v_{2,y}$ and $v_{3,k}$ are vertices satisfying $D_{x,y,k} = |s| - 1$, then $D_{i,j,k} = D_{x,y,k} + 1$. Note that such nodes $v_{1,x}$, $v_{2,y}$ and $v_{3,k}$ always exist.
3. If $L_1(v_{1,i}) \neq L_2(v_{2,j})$, there are two cases to consider:
   (a) If $v_{1,i}$ does not have in-coming edges and $v_{2,j}$ does not have in-coming edges (i.e., $\mathsf{in\_deg}(v_{1,i}) = \mathsf{in\_deg}(v_{2,j}) = 0$), then clearly $D_{i,j,k}$ does not exist and let $D_{i,j,k} = -\infty$.
   (b) Otherwise ($v_{1,i}$ has some in-coming edge(s) or $v_{2,j}$ has some in-coming edge(s)), let $v_{1,x}$ and $v_{2,y}$ be any nodes such that $(v_{1,x}, v_{1,i}) \in E_1$ and $(v_{2,y}, v_{2,j}) \in E_2$, respectively. Let s be a $\mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$. Assume on the contrary that there

exists a string $t \in \mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$ such that $|t| > |s|$. This contradicts that s is a SEQ-IC-LCS of $L_1(\mathsf{P}(v_{1,i}))$, $L_2(\mathsf{P}(v_{2,j}))$ and $L_3(\mathsf{LMP}(v_{3,k}))$, since $\mathsf{S}_{\mathrm{IC}}(v_{1,x}, v_{2,y}, v_{3,k}) \subseteq \mathsf{S}_{\mathrm{IC}}(v_{1,i}, v_{2,j}, v_{3,k})$. Hence $|t| \le |s|$. If $v_{1,x}$ is a vertex satisfying $D_{x,j,k} = |z|$, then $D_{i,j,k} = D_{x,j,k}$. Similarly, if $v_{2,y}$ is a vertex satisfying $D_{i,y,k} = |s|$, then $D_{i,j,k} = D_{i,y,k}$. Note that such node $v_{1,x}$ or $v_{2,y}$ always exists.

Consequently we obtain the following recurrence:

$$D_{i,j,k} = \begin{cases} \text{Recurrence in Equation (2)} & \text{if } k = 0; \\ 1 + \max\left(\left\{ D_{x,y,z} \;\middle|\; \begin{array}{l} (v_{1,x}, v_{1,i}) \in E_1, \\ (v_{2,y}, v_{2,j}) \in E_2, \\ (v_{3,z}, v_{3,k}) \in E_3, \\ \text{or } z = 0 \end{array} \right\} \cup \{\gamma\} \right) & \begin{array}{l} \text{if } k > 0 \text{ and} \\ L_1(v_{1,i}) = L_2(v_{2,j}) \\ = L_3(v_{3,k}); \end{array} \\ \max\left(\left\{ 1 + D_{x,y,k} \;\middle|\; \begin{array}{l} (v_{1,x}, v_{1,i}) \in E_1, \\ (v_{2,y}, v_{2,j}) \in E_2 \end{array} \right\} \cup \{-\infty\} \right) & \begin{array}{l} \text{if } k > 0 \text{ and} \\ L_1(v_{1,i}) = L_2(v_{2,j}) \\ \neq L_3(v_{3,k}); \end{array} \\ \max\left(\begin{array}{l} \{D_{x,j,k} \mid (v_{1,x}, v_{1,i}) \in E_1\} \cup \\ \{D_{i,y,k} \mid (v_{2,y}, v_{2,j}) \in E_2\} \cup \{-\infty\} \end{array} \right) & \text{otherwise.} \end{cases} \tag{3}$$

where

$$\gamma = \begin{cases} 0 & \begin{array}{l} \text{if } v_{1,i} \text{ does not have in-coming edges at all or } v_{2,j} \text{ does not have} \\ \text{in-coming edges at all, and } v_{3,k} \text{ does not have in-coming edges;} \end{array} \\ -\infty & \text{otherwise.} \end{cases}$$

We compute $D_{i,j,k}$ for all $1 \le i \le |V_1|$, $1 \le j \le |V_2|$ and $0 \le k \le |V_3|$, using a dynamic programming table of size $O(|V_1||V_2||V_3|)$.

Below we analyze the time complexity for computing $D_{i,j,k}$ with the recurrence:

- The first case with Equation (2) takes $O(|E_1||E_2|)$ time (Section 2.5).
- Second, let us analyze the time cost for computing

$$M_{i,j,k} = \max\{D_{x,y,z} \mid (v_{1,x}, v_{1,i}) \in E_1, (v_{2,y}, v_{2,j}) \in E_2, (v_{3,z}, v_{3,k}) \in E_3, \text{ or } z = 0\}$$

in the second case of the recurrence for all i, j, k. For each fixed pair of $(v_{1,x}, v_{1,i}) \in E_1$ and $(v_{2,y}, v_{2,j}) \in E_2$, we refer the value of $D_{x,y,z}$ for all $1 \le z < k$ such that $(v_{3,z}, v_{3,k}) \in E_3$, in $O(|E_3|)$ time. For each fixed $(v_{1,x}, v_{1,i}) \in E_1$, we refer the value of $D_{x,y,z}$ for all $1 \le y < j$ such that $(v_{2,y}, v_{2,j}) \in E_2$ and all $1 \le z < k$ such that $(v_{3,z}, v_{3,k}) \in E_3$, in $O(|E_2||E_3|)$ time. Therefore, the total time complexity for computing all $M_{i,j,k}$ for all i, j, k is $O(|E_1||E_2||E_3|)$.
- Third, let us analyze the time cost for computing

$$M'_{i,j,k} = \max\{D_{x,y,k} \mid (v_{1,x}, v_{1,i}) \in E_1, (v_{2,y}, v_{2,j}) \in E_2\}$$

in the third case of the recurrence for all i, j, k. For each fixed pair of $(v_{1,x}, v_{1,i}) \in E_1$ and $(v_{2,y}, v_{2,j}) \in E_2$, we refer the value of $D_{x,y,k}$ for all $1 \le k \le |V_3|$, in $O(|V_3|)$ time. For each fixed $(v_{1,x}, v_{1,i}) \in E_1$, we refer the value of $D_{x,y,k}$ for all $1 \le y < j$ such that $(v_{2,y}, v_{2,j}) \in E_2$ and all $1 \le k \le |V_3|$, in $O(|E_2||V_3|)$ time. Therefore, the total time complexity for computing $M'_{i,j,k}$ for all i, j, k is $O(|E_1||E_2||V_3|) \subseteq O(|E_1||E_2||E_3|)$.

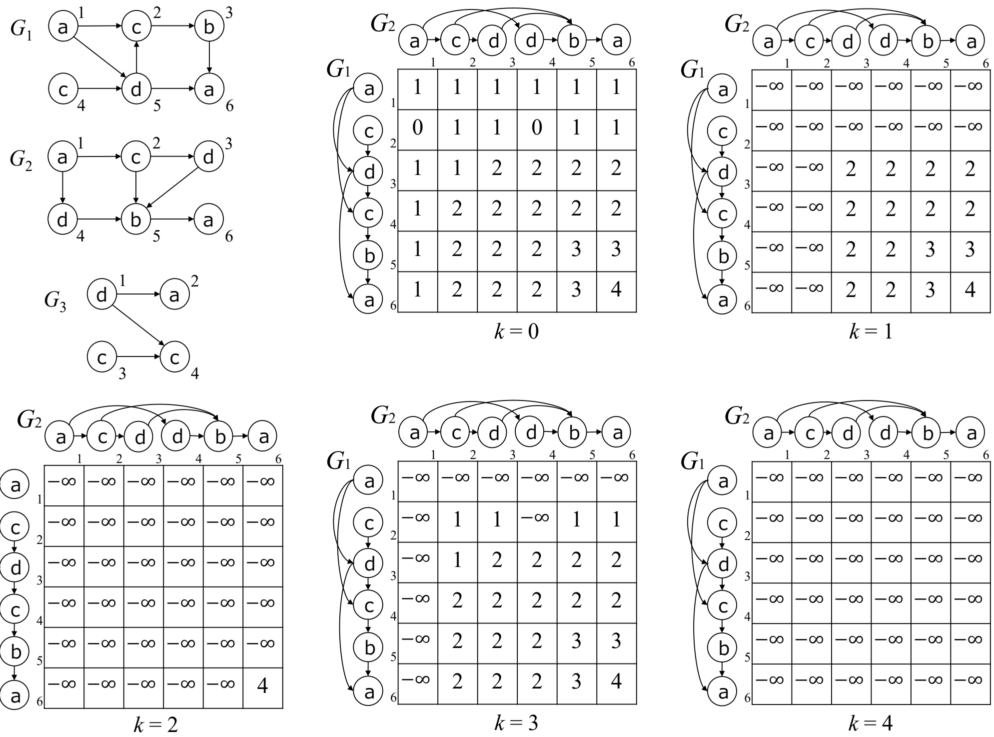

Figure 2. Example of dynamic programming table D for computing the SEQ-IC-LCS length of acyclic labeled graphs G_1, G_2 and G_3. Each vertex is annotated with its topological order. In this example, $v_{3,2}$ and $v_{3,4}$ with $k \in \{2,4\}$ in G_3 are vertices with no out-going edges. The maximum value of $D_{i,j,k}$ with $k \in \{2,4\}$ is $D_{6,6,2} = 4$, and the corresponding SEQ-IC-LCS is `cdba` of length 4.

– Fourth, let us analyze the time cost for computing

$$M''_{i,j,k} = \max\{D_{x,j,k}, D_{i,y,k} \mid (v_{1,x}, v_{1,i}) \in E_1, (v_{2,y}, v_{2,j}) \in E_2\}$$

in the fourth case of the recurrence for all i, j, k. For each fixed $(v_{1,x}, v_{1,i}) \in E_1$, we refer the value of $D_{x,j,k}$ for all $1 \le j \le |V_2|$ and all $1 \le k \le |V_3|$ in $O(|V_2||V_3|)$ time. Similarly, for each fixed $(v_{2,y}, v_{2,j}) \in E_2$, we refer the value of $D_{i,y,k}$ for all $1 \le i \le |V_1|$ and all $1 \le k \le |V_3|$ in $O(|V_1||V_3|)$ time. Therefore, the total time cost for computing $M''_{i,j,k}$ for all i, j, k is $O(|V_3|(|V_2||E_1| + |V_1||E_2|)) \subseteq O(|E_1||E_2||E_3|)$.

Thus the total time complexity is $O(|E_1||E_2||E_3|)$. □

An example of computing $D_{i,j,k}$ using dynamic programming is show in Figure 2. We remark that the recurrence in Equation (3) is a natural generalization of the recurrence in Equation (1) for computing the SEQ-IC-LCS length of given two strings.

5 Computing SEQ-IC-LCS of Cyclic Labeled Graphs

In this section, we present an algorithm to solve Problem 2 in case where G_1 and/or G_2 can be cyclic and G_3 is acyclic. We output ∞ if the set of output candidates in Problem 2 contains a string of infinite length, and outputs the (finite) SEQ-IC-LCS length otherwise.

To deal with cyclic graphs, we follow the approach by Shimohira et al. [24] which transforms a cyclic labeled graph $G = (V, E, L)$ into an acyclic labeled graph $\hat{G} = (\hat{V}, \hat{E}, \hat{L})$ based on the strongly connected components.

For each vertex $v \in V$, let $[v]$ denote the set of vertices that belong to the same strongly connected component. Formally, $\hat{G} = (\hat{V}, \hat{E}, \hat{L})$ is defined by

$$\hat{V} = \{[v] \mid v \in V\},$$
$$\hat{E} = \{([v],[u]) \mid [v] \neq [u], (\hat{v}, \hat{u}) \in E \text{ for some } \hat{v} \in [v],\ \hat{u} \in [u]\} \cup \{(v,v) \mid |[v]| \geq 2\},$$

and $\hat{L}([v]) = \{L(v) \mid v \in [v]\} \subseteq \Sigma$. We regard each $[v]$ as a single vertex that is contracted from vertices in $[v]$. Observe that $\mathsf{Subseq}(\hat{G}) = \mathsf{Subseq}(G)$. An example of transformed acyclic labeled graphs is shown in Figure 3.

It is possible that a vertex $\hat{v} \in \hat{V}$ in the transformed graph $\hat{G}$ has a self-loop. We regard that a self-loop $(\hat{v}, \hat{v})$ is also an in-coming edge of vertex $\hat{v}$. We say that vertex $\hat{v}$ does not have in-coming edges *at all*, if $\hat{v}$ does not have in-coming edges from *any* vertex in $\hat{V}$ (including $\hat{v}$).

Our main result of this section follows:

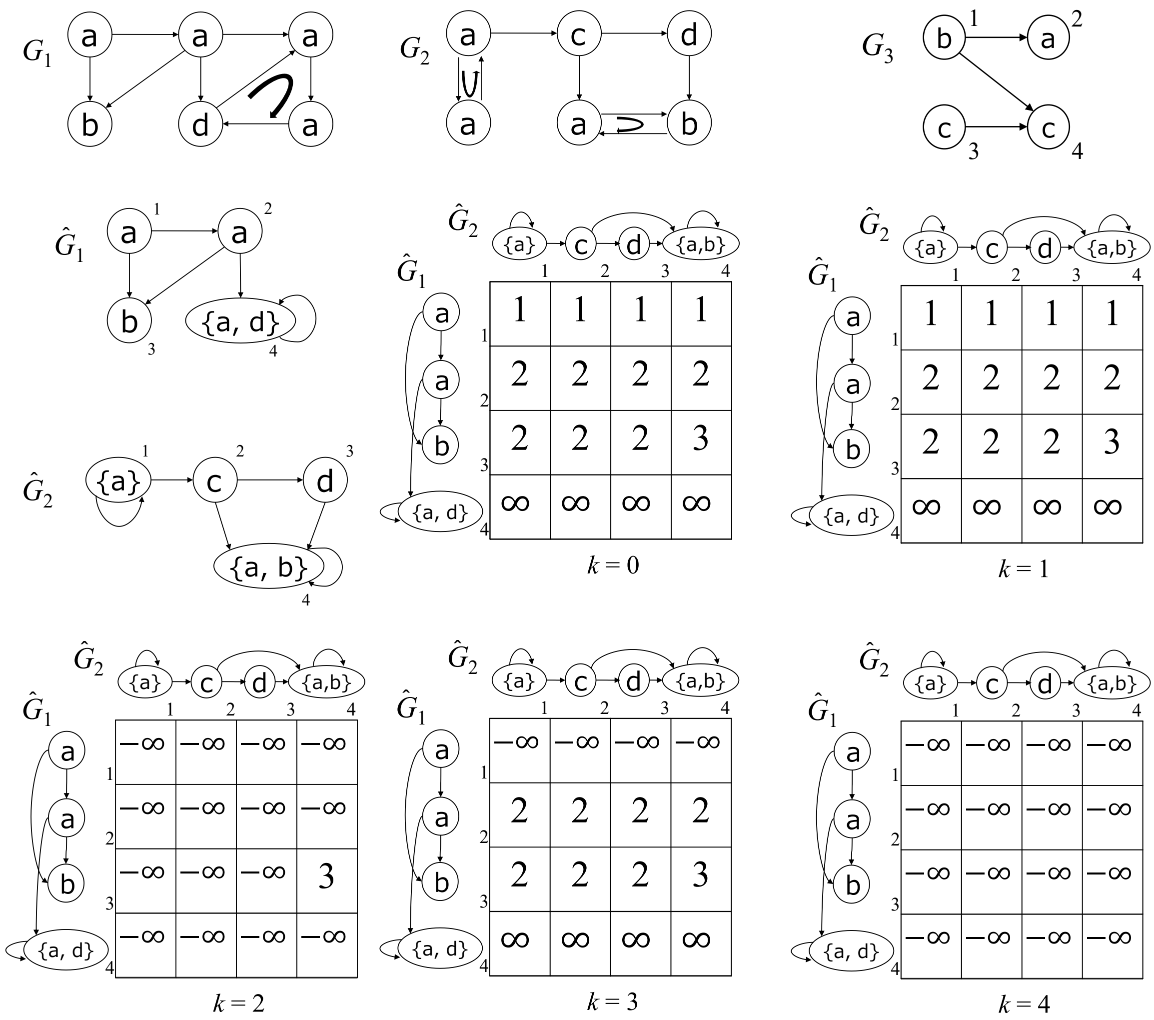

Figure 3. Example of dynamic programming table $\hat{D}$ for computing the SEQ-IC-LCS length of cyclic labeled graphs G_1 and G_2, and acyclic labeled graph G_3. $\hat{G}_1$ and $\hat{G}_2$ are the labeled graphs which are transformed from G_1 and G_2 by grouping vertices into strongly connected components. Each vertex is annotated with its topological order. In this example, $v_{3,2}$ and $v_{3,4}$ with $k \in \{2,4\}$ in G_3 are vertices with no out-going edges. The maximum value of $\hat{D}_{i,j,k}$ with $k \in \{2,4\}$ is $\hat{D}_{4,3,2} = 3$, and the corresponding SEQ-IC-LCS is `aab` of length 3.

Theorem 4. *Problem 2 with G_1 and/or G_2 cyclic and G_3 acyclic can be solved in $O(|E_1||E_2||E_3| + |V_1||V_2||V_3| \log |\Sigma|)$ time and $O(|V_1||V_2||V_3|)$ space.*

Proof. We first transform cyclic labeled graphs G_1 and G_2 into corresponding acyclic labeled graphs $\hat{G}_1$ and $\hat{G}_2$, as described previously. For $1 \le i \le |\hat{V}_1|$ and $1 \le j \le |\hat{V}_2|$, let $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ respectively denote the ith and jth vertices in $\hat{G}_1$ and $\hat{G}_2$ in topological order. Let $v_{3,k}$ denote the k-th vertex in topological ordering in G_3 for $1 \le k \le |V_3|$.

Let

$$\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k}) = \left\{ z \;\middle|\; \begin{array}{l} \exists q \in L_3(\mathsf{MP}(v_{3,k})) \text{ such that } q \in \mathsf{Subseq}(z) \\ \text{and } z \in \mathsf{Subseq}(\hat{L}_1(\mathsf{P}(\hat{v}_{1,i}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{P}(\hat{v}_{2,j}))) \end{array} \right\}.$$

Let $\hat{D}_{i,j,k}$ denote the length of a longest string in $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$. For convenience, we let $\hat{D}_{i,j,k} = -\infty$ if $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k}) = \emptyset$. The solution to Problem 2 (the SEQ-IC-LCS length) is the maximum value of $\hat{D}_{i,j,k}$ for which $v_{3,k}$ has no out-going edges (i.e. $v_{3,k}$ is the end of a maximal path in G_3).

$\hat{D}_{i,j,k}$ can be computed as follows:

1. If both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ are cyclic vertices (i.e. $|[\hat{v}_{1,i}]| \ge 2$ and $|[\hat{v}_{2,j}]| \ge 2$), then remark that both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ have some self-loop(s). There are four cases to consider:
   (a) If $k = 0$, there are two cases to consider:
      i. If $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \ne \emptyset$, then clearly $\hat{D}_{i,j,k} = \infty$.
      ii. Otherwise, there are two cases to consider:
         A. If the in-coming edges of $\hat{v}_{1,i}$ are $\hat{v}_{2,j}$ only self-loops, then clearly $\hat{D}_{i,j,k} = 0$.
         B. Otherwise ($\hat{v}_{1,i}$ has some in-coming edge(s) other than self-loops, or $\hat{v}_{2,j}$ has some in-coming edge(s) other than self-loops), let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. Let s be a longest string in the set $\mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j})))$. Assume on the contrary that there is a string $t \in \mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,x}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j})))$ such that $|t| > |s|$. This contradicts that s is a longest common subsequence of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$ and $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$, since $\mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,x}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))) \subseteq \mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j})))$. Hence $|t| \le |s|$. If $\hat{v}_{1,x}$ is a vertex satisfying $\hat{D}_{x,j,k} = |s|$, then $\hat{D}_{i,j,k} = \hat{D}_{x,j,k}$. Similarly, if $\hat{v}_{2,y}$ is a vertex satisfying $\hat{D}_{i,y,k} = |s|$, then $\hat{D}_{i,j,k} = \hat{D}_{i,y,k}$. Note that such $\hat{v}_{1,x}$ or $\hat{v}_{2,y}$ always exists.
   (b) If $k > 0$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \ne \emptyset$, there are two cases to consider:
      i. If $v_{3,k}$ has no in-coming edges, let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively (these edges may be self-loops). If $\hat{D}_{x,y,0} = -\infty$ for all $1 \le x < i$ and $1 \le y < j$, then clearly $\hat{D}_{i,j,k} = -\infty$. Otherwise, clearly $\hat{D}_{i,j,k} = \infty$.
      ii. Otherwise ($v_{3,k}$ has some in-coming edge(s)), let $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,z}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$, $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$ and $(v_{3,z}, v_{3,k}) \in E_3$, respectively (the first two edges may be self-loops). If $\hat{D}_{x,y,z} = -\infty$ for all $1 \le x < i$ and $1 \le y < j$, then clearly $\hat{D}_{i,j,k} = -\infty$. Otherwise, $\hat{D}_{i,j,k} = \infty$.
   (c) If $k > 0$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \ne \emptyset$, there are two cases to consider:

i. If the in-coming edges of $\hat{v}_{1,i}$ are $\hat{v}_{2,j}$ only self-loops, then clearly $\hat{D}_{i,j,k} = -\infty$.

ii. Otherwise ($\hat{v}_{1,i}$ has some in-coming edge(s) other than self-loops, or $\hat{v}_{2,j}$ has some in-coming edge(s) other than self-loops), let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. If all $\hat{D}_{x,y,k} = -\infty$, then clearly $\hat{D}_{i,j,k} = -\infty$. Otherwise, clearly $\hat{D}_{i,j,k} = \infty$.

(d) If $k > 0$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset$, there are two cases to consider:

i. If the in-coming edges of $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ are only self-loops, then clearly $\hat{D}_{i,j,k} = -\infty$.

ii. Otherwise ($\hat{v}_{1,i}$ has some in-coming edge(s) other than self-loops, or $\hat{v}_{2,j}$ has some in-coming edge(s) other than self-loops), let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. Let s be a longest string in $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$. Assume on the contrary that there exists a string $t \in \hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$ such that $|t| > |s|$. This contradicts that s is a SEQ-IC-LCS of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$, $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$ and $L_3(\mathsf{MP}(v_{3,k}))$, since $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,x}, \hat{v}_{2,y}, v_{3,k}) \subseteq \hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$. Hence $|t| \leq |s|$. If $\hat{v}_{1,x}$ is a vertex satisfying $\hat{D}_{x,j,k} = |z|$, then $\hat{D}_{i,j,k} = \hat{D}_{x,j,k}$. Similarly, if $\hat{v}_{2,y}$ is a vertex satisfying $\hat{D}_{i,y,k} = |s|$, then $\hat{D}_{i,j,k} = \hat{D}_{i,y,k}$. Note that such $\hat{v}_{1,x}$ or $\hat{v}_{2,y}$ always exists.

2. Otherwise ($v_{1,i}$ is not a cyclic vertex and/or $v_{2,j}$ is not a cyclic vertex), there are four cases to consider:

(a) If $k = 0$, there are two cases to consider:

i. If $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$, there are two cases to consider:

A. If $\hat{v}_{1,i}$ does not have in-coming edges at all or $\hat{v}_{2,j}$ does not have in-coming edges at all, then clearly $\hat{D}_{i,j,k} = 1$.

B. Otherwise (both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ have some in-coming edge(s) including self-loops), let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. Let s be a longest string in the set $\mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j})))$. Assume on the contrary that there is a string $t \in \mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,x}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,y})))$ such that $|t| > |s| - 1$. This contradicts that s is a longest common subsequence of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$ and $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$, since $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$. Hence $|t| \leq |s| - 1$. If $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ are vertices satisfying $\hat{D}_{x,y,k} = |s| - 1$, then $\hat{D}_{i,j,k} = \hat{D}_{x,y,k} + 1$. Note that such $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ always exist.

ii. Otherwise, then this case is the same as Case 1(a)ii.

(b) If $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \neq \emptyset$, there are three cases to consider:

i. If $\hat{v}_{1,i}$ does not have in-coming edges at all or $\hat{v}_{2,j}$ does not have in-coming edges at all, and if $v_{3,k}$ does not have in-coming edges, then clearly $\hat{D}_{i,j,k} = 1$.

ii. If $\hat{v}_{1,i}$ does not have in-coming edges at all or $\hat{v}_{2,j}$ does not have in-coming edge at all, and if $v_{3,k}$ has some in-coming edge(s), then clearly $\hat{D}_{i,j,k} = -\infty$.

iii. If both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ have some in-coming edge(s) including self-loops and $v_{3,k}$ does not have in-coming edges, let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. Let s be a longest string in the set $\mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j})))$. Assume on the contrary that there exists a string $t \in \mathsf{Subseq}(\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,x}))) \cap \mathsf{Subseq}(\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,y})))$ such that $|t| > |s| - 1$. This contradicts that s is a longest common subsequence of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$ and $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$, since $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$. Hence $|t| \leq |s| - 1$. If $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ are vertices satisfying $\hat{D}_{x,y,0} = |s| - 1$, then $\hat{D}_{i,j,k} = \hat{D}_{x,y,0} + 1$. Note that such $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ always exist.

iv. Otherwise (all $\hat{v}_{1,i}$, $\hat{v}_{2,j}$, and $\hat{v}_{3,k}$ have some in-coming edge(s) including self-loops), let $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,z}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$, $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, and $(v_{3,z}, v_{3,k}) \in E_3$, respectively. Let s be a longest string in $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$. Assume on the contrary that there exists a string $t \in \hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,x}, \hat{v}_{2,y}, v_{3,z})$ such that $|t| > |s| - 1$. This contradicts that s is a SEQ-IC-LCS of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$, $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$ and $L_3(\mathsf{MP}(v_{3,k}))$, since $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap L_3(v_{3,k}) \neq \emptyset$. Hence $|t| \leq |s| - 1$. If $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,z}$ are vertices satisfying $\hat{D}_{x,y,z} = |s| - 1$, then $\hat{D}_{i,j,k} = \hat{D}_{x,y,z} + 1$. Note that such $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,z}$ always exist.

(c) If $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$, there are two cases to consider:

i. If $\hat{v}_{1,i}$ does not have in-coming edges at all or $\hat{v}_{2,j}$ does not have in-coming edges at all, then clearly $\hat{D}_{i,j,k} = -\infty$.

ii. Otherwise (both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ have some in-coming edges including self-loops), let $\hat{v}_{1,x}$ and $\hat{v}_{2,y}$ be any nodes such that $(\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1$ and $(\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2$, respectively. Let s be a longest string in $\hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,i}, \hat{v}_{2,j}, v_{3,k})$. Assume on the contrary that there exists a string $t \in \hat{\mathsf{S}}_{\mathrm{IC}}(\hat{v}_{1,x}, \hat{v}_{2,y}, v_{3,k})$ such that $|t| > |s| - 1$. This contradicts that s is a SEQ-IC-LCS of $\hat{L}_1(\mathsf{LMP}(\hat{v}_{1,i}))$, $\hat{L}_2(\mathsf{LMP}(\hat{v}_{2,j}))$ and $L_3(\mathsf{MP}(v_{3,k}))$, since $\hat{L}_1(\hat{v}_{1,i}] \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$. Hence $|t| \leq |s| - 1$. If $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,k}$ are vertices satisfying $\hat{D}_{x,y,k} = |s| - 1$, then $\hat{D}_{i,j,k} = \hat{D}_{x,y,k} + 1$. Note that such $\hat{v}_{1,x}$, $\hat{v}_{2,y}$ and $v_{3,k}$ always exist.

(d) If $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset$, then this case is the same as Case 1d.

The above arguments lead us to the following recurrence:

$$
\hat{D}_{i,j,k} =
\begin{cases}
\delta + \max\left(\left\{ \hat{D}_{x,y,k} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \end{array} \right\} \cup \{0\}\right) & \begin{array}{l}\text{if } k = 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset;\end{array} \\
\max\left(\begin{array}{l} \{\hat{D}_{x,j,k} \mid (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1\} \cup \\ \{\hat{D}_{i,y,k} \mid (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2\} \cup \{0\} \end{array}\right) & \begin{array}{l}\text{if } k = 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset;\end{array} \\
\delta + \max\left(\left\{ \hat{D}_{x,y,z} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2, \\ (v_{3,z}, v_{3,k}) \in E_3 \\ \text{or } z = 0 \end{array} \right\} \cup \{\gamma\}\right) & \begin{array}{l}\text{if } k > 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3([v_{3,k}])\} \\ \neq \emptyset;\end{array} \\
\max\left(\left\{ \delta + \hat{D}_{x,y,k} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \end{array} \right\} \cup \{-\infty\}\right) & \begin{array}{l}\text{if } k > 0, \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \\ = \emptyset, \text{ and } \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset;\end{array} \\
\max\left(\begin{array}{l} \{\hat{D}_{x,j,k} \mid (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1\} \cup \\ \{\hat{D}_{i,y,k} \mid (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2\} \cup \{-\infty\} \end{array}\right) & \text{otherwise,}
\end{cases}
$$

where

$$\delta = \begin{cases} \infty & \text{if both } \hat{L}_1(\hat{v}_{1,i}) \text{ and } \hat{L}_2(\hat{v}_{2,j}) \text{ are cyclic vertices;} \\ 1 & \text{otherwise,} \end{cases}$$

$$\gamma = \begin{cases} 0 & \begin{array}{l}\text{if } \hat{v}_{1,i} \text{ does not have in-coming edges at all or } \hat{v}_{2,j} \text{ does not have} \\ \text{in-coming edges at all, and } v_{3,k} \text{ does not have in-coming edges;}\end{array} \\ -\infty & \text{otherwise.} \end{cases}$$

In the above recurrence, we use a convention that $\infty + (-\infty) = -\infty$.

We perform preprocessing which transforms G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ in $O(|E_1| + |E_2|)$ time with $O(|V_1| + |V_2|)$ space, based on strongly connected components.

To examine the conditions in the above recurrence, we explicitly construct the intersection of the character labels of the given vertices $\hat{v}_{1,i} \in \hat{V}_1$, $\hat{v}_{2,j} \in \hat{V}_2$, and $\hat{v}_{3,k} \in V_3$ by using balanced trees, as follows:

- Checking whether $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset$ or $\neq \emptyset$: Let Σ_1 and Σ_2 be the sets of characters that appear in G_1 and G_2, respectively. For every node $\hat{v}_{1,i} \in \hat{V}_1$ of the transformed graph $\hat{G}_1$, we build a balanced tree $\mathcal{T}_i$ which consists of the characters in $\hat{L}_1(\hat{v}_i)$. Since the total number of characters in the original graph $G_1 = (V_1, E_1)$ is equal to $|V_1|$, we can build the balanced trees $\mathcal{T}_i$ for all i in a total of $O(|V_1| \log |\Sigma_1|)$ time and $O(|V_1|)$ space. Then, for each fixed $\hat{L}_1(\hat{v}_{1,i}) \in \hat{V}_1$, by using its balanced tree, the intersection $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j})$ can be computed in $O(|V_2| \log |\Sigma_1|)$ time for all $\hat{L}_2(\hat{v}_{2,j}) \in V_2$. Therefore, $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j})$ for all $1 \leq i \leq |\hat{V}_1|$ and $1 \leq j \leq |\hat{V}_2|$ can be computed in $O(|V_1||V_2| \log |\Sigma_1|)$ total time.
- Checking whether $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap L_3(v_{3,k}) = \emptyset$ or $\neq \emptyset$: While computing $\Sigma_{i,j} = \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j})$ in the above, we also build another balanced tree $\mathcal{T}_{i,j}$ which consists of the characters in $\Sigma_{i,j}$ for every $1 \leq i \leq |\hat{V}_1|$ and $1 \leq j \leq |\hat{V}_2|$. This can be done in $O(|V_1||V_2| \log |\Sigma_1|)$ total time and $O(|V_1||V_2|)$ space. Then, for each fixed $1 \leq i \leq |\hat{V}_1|$ and $1 \leq j \leq |\hat{V}_2|$, $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap L_3(v_{3,k})$ can be computed in a total of $O(|V_3| \log |\Sigma_{i,j}|)$ time. Therefore, $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap L_3(v_{3,k})$ for all $1 \leq i \leq |\hat{V}_1|$, $1 \leq j \leq |\hat{V}_2|$ and, $1 \leq k \leq |V_3|$ can be computed in $O(|V_1||V_2||V_3| \log |\Sigma|)$ time.

Assuming that the above preprocessing for the conditions in the recurrence are all done, we can compute $\hat{D}_{i,j,k}$ for all $1 \leq i \leq |\hat{V}_1|$, $1 \leq j \leq |\hat{V}_2|$ and $1 \leq k \leq |V_3|$ using dynamic programming of size $O(|\hat{V}_1||\hat{V}_2||V_3|)$ in $O(|\hat{E}_1||\hat{E}_2||E_3|)$ time, in a similar way to the acyclic case for Theorem 3.

Overall, the total time complexity is $O(|E_1| + |E_2| + |E_3| + |\hat{V}_1||\hat{V}_2| \log |\Sigma_1| + |\hat{V}_1||\hat{V}_2||V_3| \log |\Sigma| + |\hat{E}_1||\hat{E}_2||E_3|) \subseteq O(|E_1||E_2||E_3| + |V_1||V_2||V_3| \log |\Sigma|)$.

The total space complexity is $O(|V_1||V_2| + |\hat{V}_1||\hat{V}_2||V_3|) \subseteq O(|V_1||V_2||V_3|)$. □

An example of computing $\hat{D}_{i,j,k}$ using dynamic programming is shown in Figure 3.

6 Conclusions and Open Questions

In this paper, we introduced the new problem of computing the SEQ-IC-LCS on labeled graphs. We showed that when the all the input labeled graphs are acyclic, the problem can be solved in $O(|E_1||E_2|||E_3)$ time and $O(|V_1||V_2||V_3|)$ space by a dynamic programming approach. Furthermore, we extend our algorithm to a more general case where the two target labeled graphs can contain cycles, and presented an efficient algorithm that runs in $O(|E_1||E_2||E_3| + |V_1||V_2||V_3| \log |\Sigma|)$ time and $O(|V_1||V_2||V_3|)$ space.

Interesting open questions are whether one can extend the framework of our methods to the other variants STR-IC/EC-LCS and SEQ-EC-LCS of the constrained LCS problems in the case of labeled graph inputs. We believe that SEQ-EC-LCS for labeled graphs can be solved by similar methods to our SEQ-IC-LCS methods, within the same bounds.

References

1. A. Abboud, A. Backurs, and V. V. Williams: *Tight hardness results for LCS and other sequence similarity measures*, in FOCS 2015, 2015, pp. 59–78.
2. A. Amir, M. Lewenstein, and N. Lewenstein: *Pattern matching in hypertext*, in WADS 1997, vol. 1272 of LNCS, 1997, pp. 160–173.
3. R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc: *Foundations of modern query languages for graph databases.* ACM Comput. Surv., 50(5) 2017, pp. 68:1–68:40.
4. K. Aoyama, Y. Nakashima, T. I, S. Inenaga, H. Bannai, and M. Takeda: *Faster online elastic degenerate string matching*, in CPM 2018, vol. 105 of LIPIcs, 2018, pp. 9:1–9:10.
5. A. N. Arslan: *Regular expression constrained sequence alignment.* Journal of Discrete Algorithms, 5(4) 2007, pp. 647–661, Selected papers from Combinatorial Pattern Matching 2005.
6. A. N. Arslan and Ö. Egecioglu: *Algorithms for the constrained longest common subsequence problems.* Int. J. Found. Comput. Sci., 16(6) 2005, pp. 1099–1109.
7. G. Bernardini, P. Gawrychowski, N. Pisanti, S. P. Pissis, and G. Rosone: *Elastic-degenerate string matching via fast matrix multiplication.* SIAM J. Comput., 51(3) 2022, pp. 549–576.
8. G. Bernardini, N. Pisanti, S. P. Pissis, and G. Rosone: *Approximate pattern matching on elastic-degenerate text.* Theor. Comput. Sci., 812 2020, pp. 109–122.
9. K. Bringmann and M. Künnemann: *Quadratic conditional lower bounds for string problems and dynamic time warping*, in FOCS 2015, 2015, pp. 79–97.
10. M. Cáceres: *Parameterized algorithms for string matching to DAGs: Funnels and beyond*, in CPM 2023, vol. 259 of LIPIcs, 2023, pp. 7:1–7:19.
11. Y.-C. Chen and K.-M. Chao: *On the generalized constrained longest common subsequence problems.* Journal of Combinatorial Optimization, 21(3) Apr 2011, pp. 383–392.
12. F. Y. Chin, A. D. Santis, A. L. Ferrara, N. Ho, and S. Kim: *A simple algorithm for the constrained sequence problems.* Information Processing Letters, 90(4) 2004, pp. 175 – 179.
13. J. Conklin: *Hypertext: An introduction and survey.* IEEE Computer, 20(9) 1987, pp. 17–41.
14. T. C. P.-G. Consortium: *Computational pan-genomics: status, promises and challenges.* Briefings in Bioinformatics, 19(1) 2016, pp. 118–135.
15. S. Deorowicz: *Quadratic-time algorithm for a string constrained lcs problem.* Information Processing Letters, 112(11) 2012, pp. 423 – 426.
16. M. Equi, R. Grossi, V. Mäkinen, and A. I. Tomescu: *On the complexity of string matching for graphs*, in ICALP 2019, vol. 132 of LIPIcs, 2019, pp. 55:1–55:15.
17. M. Equi, V. Mäkinen, and A. I. Tomescu: *Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless SETH fails*, in SOFSEM 2021, vol. 12607 of Lecture Notes in Computer Science, 2021, pp. 608–622.
18. R. Grossi, C. S. Iliopoulos, C. Liu, N. Pisanti, S. P. Pissis, A. Retha, G. Rosone, F. Vayani, and L. Versari: *On-line pattern matching on similar texts*, in CPM 2017, vol. 78 of LIPIcs, 2017, pp. 9:1–9:14.
19. C. S. Iliopoulos, R. Kundu, and S. P. Pissis: *Efficient pattern matching in elastic-degenerate strings.* Inf. Comput., 279 2021, p. 104616.

20. G. Kucherov, T. Pinhas, and M. Ziv-Ukelson: *Regular language constrained sequence alignment revisited*, in IWOCA 2011, 2011, pp. 404–415.
21. U. Manber and S. Wu: *Approximate string matching with arbitrary costs for text and hypertext*, in Proc. IAPR, 1992, pp. 22–33.
22. G. Navarro: *Improved approximate pattern matching on hypertext.* Theoretial Computer Science, 237(1–2) 2000, pp. 455–463.
23. K. Park and D. K. Kim: *String matching in hypertext*, in Proc. CPM'95, 1995, pp. 318–329.
24. K. Shimohira, S. Inenaga, H. Bannai, and M. Takeda: *Computing longest common substring/subsequence of non-linear texts*, in PSC 2011, 2011, pp. 197–208.
25. Y.-T. Tsai: *The constrained longest common subsequence problem.* Information Processing Letters, 88(4) 2003, pp. 173 – 176.
26. R. A. Wagner and M. J. Fischer: *The string-to-string correction problem.* J. ACM, 21(1) Jan. 1974, pp. 168–173.
27. L. Wang, X. Wang, Y. Wu, and D. Zhu: *A dynamic programming solution to a generalized LCS problem.* Inf. Process. Lett., 113(19-21) 2013, pp. 723–728.
28. Y. Yonemoto, Y. Nakashima, S. Inenaga, and H. Bannai: *Space-efficient STR-IC-LCS computation*, in SOFSEM 2023, vol. 13878 of LNCS, 2023, pp. 372–384.