Theoretical Computer Science 796 (2019) 272–285

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Comparing incomplete sequences via longest common subsequence ☆

Mauro Castelli [a], Riccardo Dondi [b,*], Giancarlo Mauri [c], Italo Zoppis [c]

[a] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312, Lisboa, Portugal
[b] Dipartimento di Scienze umane e sociali, Università degli Studi di Bergamo, Italy
[c] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Milano, Italy

ARTICLE INFO

*Article history:*
Received 17 October 2018
Received in revised form 26 July 2019
Accepted 14 September 2019
Available online 19 September 2019
Communicated by T. Calamoneri

*Keywords:*
Longest common subsequence
String algorithms
Approximation algorithms
Computational complexity
Fixed-parameter algorithms

ABSTRACT

Inspired by scaffold filling, a recent approach for genome reconstruction from incomplete data, we consider a variant of the well-known longest common subsequence problem for the comparison of two sequences. The new problem, called Longest Filled Common Subsequence, aims to compare a complete sequence with an incomplete one, i.e. with some missing elements. Longest Filled Common Subsequence (LFCS), given a complete sequence $A$, an incomplete sequence $B$, and a multiset $\mathcal{M}$ of symbols missing in $B$, asks for a sequence $B^*$ obtained by inserting the symbols of $\mathcal{M}$ into $B$ so that $B^*$ induces a common subsequence with $A$ of maximum length.

We investigate the computational and approximation complexity of the problem and we show that it is NP-hard and APX-hard when $A$ contains at most two occurrences of each symbol, and we give a polynomial time algorithm when the input sequences are over a constant-size alphabet. We give a $\frac{3}{5}-$approximation algorithm for the Longest Filled Common Subsequence problem. Finally, we present a fixed-parameter algorithm for the problem, when it is parameterized by the number of symbols inserted in $B$ that "match" symbols of $A$.



## 1. Introduction

Longest Common Subsequence (LCS) is a well-known approach to compare sequences and it has been applied in several contexts where the goal is to retrieve the maximum number of elements that appear in the same order in two or more sequences. Well-known fields of application of LCS include scheduling, data compression and computational biology. LCS has been widely applied to compare molecular sequences in bioinformatics [2,3]. For example, the comparison of biological sequences provides a measure of their similarities and differences, aiming at understanding whether they encode similar/different functionalities.

Different variants of LCS have been considered in the last years for the comparison of two genomes, for example the constrained longest common subsequence [4–8] or the repetition-free longest common subsequence and variants thereof [9–12]. These approaches assume that the input sequences are complete, that is there are no missing data in the considered

☆ A preliminary version of this paper appeared in CPM 2017 [1].

* Corresponding author.
*E-mail addresses:* mcastelli@novaims.unl.pt (M. Castelli), riccardo.dondi@unibg.it (R. Dondi), mauri@disco.unimib.it (G. Mauri), zoppis@disco.unimib.it (I. Zoppis).

https://doi.org/10.1016/j.tcs.2019.09.022
0304-3975/

sequences. However, there are cases where the considered sequences are not complete. For example, Next Generation Sequencing technologies produce a huge amount of DNA/RNA fragments, called *scaffold*, while released genomes obtained by assembling these fragments, are often incomplete [13].

An approach for the reconstruction of complete genomes is to complete scaffolds with missing genes, based on the comparison of a scaffold with a reference genome [14–16]. Given an incomplete genome $B$, a multiset of missing genes (symbols) $\mathcal{M}$ and a reference genome $A$, the goal is to insert the missing symbols in $B$ so that the number of common adjacencies between the resulting genome $B^*$ and $A$ is maximized. We have a common adjacency when two genes $a$, $b$ are consecutive both in $A$ and $B^*$, independently from the order. Notice that other measures to compare a genome and a scaffold have been introduced in [17,18]. We mention briefly that there is also a variant of the scaffold filling approach that compares two incomplete genomes [15,16].

Inspired by methods for genome comparison based on LCS and by the scaffold filling approach, we introduce a new variant of the LCS problem, called the *Longest Filled Common Subsequence* problem, for the comparison of a complete sequence $A$ and an incomplete sequence $B$. The goal is to find the maximum number of symbols that appear in the same order in $A$ and in a filling $B^*$ of $B$, that is of a sequence obtained from $B$ by inserting the symbols of a multiset $\mathcal{M}$ of missing symbols into $B$. Notice that while the scaffold filling problem aims to reconstruct a complete genome from an incomplete one by maximizing the number of common adjacencies, here we aim to infer only those elements that appear in the same order in the complete sequence $A$ and in the filling $B^*$.

The approach we introduce can be of interest when comparing two orders of the same set of data. An example is the comparison of two similar, but different, schedules of a set of activities. Suppose we have a complete schedule $A$ and an incomplete schedule $B$ and we would like to reconstruct the complete schedule $B^*$ starting from $B$. A possible approach is to consider the insertion of missing activities (i.e. symbols) into $B$ such that we obtain similar complete orders of activities $A$ and $B^*$.

In this paper, we investigate different algorithmic and complexity aspects of the Longest Filled Common Subsequence problem. We prove that the problem is NP-hard and APX-hard, even when the sequence $A$ contains at most two occurrences of each symbol of the alphabet. Notice that bounding the maximum number of occurrences of symbols in a sequence is relevant in this case, as usually the number of copies of a gene inside a genome is bounded. On the positive side, we show that if $A$ and $B$ are defined over an alphabet of constant size, then there is a polynomial-time algorithm for Longest Filled Common Subsequence.

We consider two possible algorithmic directions to deal with the problem: approximation algorithms and fixed-parameter algorithms. For the first direction, we present a polynomial-time approximation algorithm of factor $\frac{3}{5}$. For the second direction, we present a fixed-parameter algorithm, where the parameter is the number of inserted symbols that lead to a "match" with symbols of $A$. Such a parameter can be of interest when the number of missing elements, and in particular those that lead to a "match" with symbols of $A$, is moderate, as the complexity of the algorithm depends exponentially only on this parameter.

The rest of the paper is organized as follows. In Section 2, we introduce some basic definitions and we formally define the Longest Filled Common Subsequence problem. Then, in Section 3 we investigate the computational and approximation complexity of the problem, showing that it is APX-hard (hence NP-hard) when each symbol in $A$ has at most two occurrences. In Section 4 we present a polynomial-time approximation algorithm of factor $\frac{3}{5}$. In Section 5, we present a fixed-parameter algorithm where the parameter is the number of inserted symbols that induce a "match" with symbols of sequence $A$ and a polynomial-time algorithm when the size of the alphabet over which $A$ and $B$ are defined is a constant. Finally, we conclude the paper with some open problems.

## 2. Preliminaries

In this section we introduce some basic definitions that will be useful in the rest of the paper and we give the formal definition of the Longest Filled Common Subsequence problem. Let $S$ be a sequence over an alphabet $\Sigma$, we denote by $|S|$ the length of $S$. Given a position $i$, with $1 \leq i \leq |S|$, we denote by $S[i]$ the symbol in position $i$ of $S$. Given two positions $i, j$ in $S$, with $1 \leq i \leq j \leq |S|$, we denote by $S[i, j]$ the substring of $S$ that starts at position $i$ and ends at position $j$. Given two sequences $S$ and $T$, we denote by $S \cdot T$ the sequence that results by concatenating $S$ and $T$.

A subsequence of $S$ is a sequence $S'$ that is obtained from $S$ by deleting some symbols (possibly none). A common subsequence $S$ of two sequences $A$ and $B$ is a subsequence of both $A$ and $B$. A longest common subsequence of $A$ and $B$ is a common subsequence of $A$ and $B$ having maximum length.

Given two sequences $A$ and $B$, a common subsequence can be represented by its associated *threading schema* (see Fig. 1): $A$ and $B$ are written on different lines, with the leftmost symbols aligned and positions of $A$ and $B$ containing an identical symbol are connected with a line (at most one for each position), such that there is no pair of crossing lines. Given a threading schema for sequences $A$, $B$, a line that connects two symbols in $A$ and $B$ is called a *match* and the two positions incident in a line are said to be *matched*.

Given a sequence $S$ and a multiset of symbols $\mathcal{M}$, we define a *filling* of $S$ with $\mathcal{M}$ as a sequence $S'$ obtained by inserting a subset $\mathcal{M}'$ of symbols of $\mathcal{M}$ into $S$. Notice that in a filling of $S$ with $\mathcal{M}$ not all the symbols of $\mathcal{M}$ have to be inserted in $S$. Informally, we may not insert those symbols that do not induce matches, to simplify the algorithms we describe in Section 4 and in Section 5.

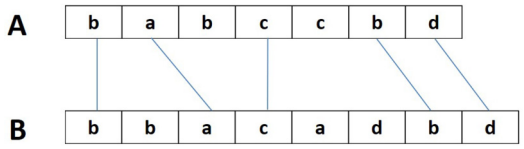


**Fig. 1.** The threading schema of two sequences $A$ and $B$: lines connect matched positions of $A$ and $B$.

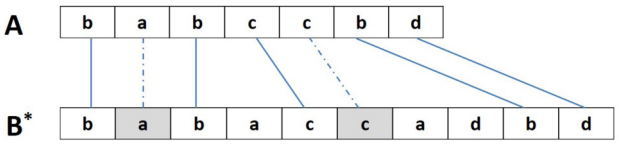


**Fig. 2.** A filling $B^*$ of sequence $B$ in Fig. 1, where the symbol in position 2 (symbol $a$) and the symbol in position 6 (symbol $c$), both in gray, have been inserted. A subsequence of $A$ and $B^*$ is induced by the threading schema of $A$ and $B^*$, where straight lines represent matches by alignment, dashed lines represent matches by insertion.

**Algorithm 1:** Algorithm that computes a subsequence of $A$ that matches by insertion the maximum number of symbols of $\mathcal{M}$.

**Data**: $A$, $\mathcal{M}$
**Result**: a subsequence $A'$ of $A$ that matches the maximum number of symbols in $\mathcal{M}$
1 $d_i :=$ number of occurrences of $\alpha_i \in \Sigma$ in $A$;
2 $m_i :=$ number of occurrences of $\alpha_i \in \Sigma$ in $\mathcal{M}$;
3 **for** *each* $\alpha_i \in \Sigma$ **do**
4  **if** $d_i \leq m_i$ **then**
5  Define a matching by insertion for all the $d_i$ occurrences of $\alpha_i$ in $A$
6  **else**
7  Define a matching by insertion for the $m_i$ rightmost occurrences of $\alpha_i$ in $A$
8 Define $A'$ as the subsequence of $A$ induced by the positions matched by insertion.

Now, we present the formal definition of the Longest Filled Common Subsequence problem.

**Problem 1.** Longest Filled Common Subsequence ($\mathcal{LFCS}$)
**Instance:** two sequences $A$ and $B$ over an alphabet $\Sigma$, and a multiset $\mathcal{M}$ over $\Sigma$.
**Solution:** a filling $B^*$ of $B$ with $\mathcal{M}$.
**Measure:** the length of a longest common subsequence of $A$ and $B^*$ (to be maximized).

Given two sequences $A$, $B$ and a multiset $\mathcal{M}$ over $\Sigma$, let $B^*$ be a filling of $B$ with $\mathcal{M}$. Consider a common subsequence of $A$ and $B^*$, and their corresponding threading schema. We can distinguish two types of matches of a matched position of $A$ (see Fig. 2): (1) a *match by insertion*, if it is a match with a position of $B^*$ that contains a symbol of $\mathcal{M}$ inserted in $B$, or (2) a *match by alignment*, if it is a match with a position of $B^*$ that have not been inserted into $B$.

We can easily compute in polynomial-time two upper bounds on the number of positions of $A$ that can be matched by alignment and by insertion, that will be useful in Section 4. The first upper bound is related to a longest common subsequence $L$ of $A$ and $B$, which can be computed in polynomial time. In fact, the maximum number of positions of $A$ (and of a filling $B^*$ of $B$ with $\mathcal{M}$) that are matched by alignment is at most the length of $L$.

Next we show how to compute in polynomial-time an upper bound on the number of position of a sequence $A$ that can be matched by insertion.

Next, we prove the correctness of Algorithm 1.

**Lemma 1.** *Given a sequence $A$, a multiset $\mathcal{M}$ on $\Sigma$, Algorithm 1 computes a subsequence of $A$ that matches by insertion the maximum number of symbols of $\mathcal{M}$.*

**Proof.** Let $d_i$ be the number of occurrences of $\alpha_i \in \Sigma$ in $A$, and $m_i$ be the number of occurrences of $\alpha_i \in \Sigma$ in $\mathcal{M}$. The subsequence $A'$ of $A$ returned by Algorithm 1, for each $\alpha_i \in \Sigma$, matches by insertion $\min(d_i, m_i)$ positions of $A$ containing $\alpha_i$. Since any subsequence of $A$ can match by insertion at most $\min(d_i, m_i)$ positions of $A$ containing $\alpha_i$, the lemma follows. □

## 3. Complexity of $\mathcal{LFCS}$

In this section, we investigate the computational and approximation complexity of the $\mathcal{LFCS}$ problem, and we prove that it is APX-hard when $A$ contains at most two occurrences of each symbol in $\Sigma$ (we denote this restriction of $\mathcal{LFCS}$ by 2-$\mathcal{LFCS}$). We prove that 2-$\mathcal{LFCS}$ is APX-hard, by giving an L-reduction from the Maximum Independent Set problem

on Cubic Graphs (Max-ISC), which is known to be APX-hard [19] (see [20] for details on L-reduction). Given a cubic graph $G = (V, E)$,[1] Max-ISC asks for a maximum cardinality subset $V' \subseteq V$ such that, given $v_i, v_j \in V'$, it holds $\{v_i, v_j\} \notin E$.

Given a cubic graph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ and $|E| = m$, in the following we show how to construct an instance $(A, B, \mathcal{M})$ of 2-$\mathcal{LFCS}$. Define an order on the edges incident on a vertex $v_i \in V$ assuming $\{v_i, v_j\} < \{v_i, v_h\}$ if $j < h$. Given a vertex $v_i$, and the edges $\{v_i, v_j\}, \{v_i, v_h\}, \{v_i, v_z\} \in E$, with $j < h < z$, we say that $\{v_i, v_j\}$ ($\{v_i, v_h\}$, $\{v_i, v_z\}$, respectively) is the first (second, third, respectively) edge incident on $v_i$.

First, we define the alphabet $\Sigma$:

$$\Sigma = \{x_{i,p} : v_i \in V, 1 \leq p \leq 3\} \cup \{y_{i,p} : v_i \in V, 1 \leq p \leq 2\} \cup$$

$$\{z_{i,p} : 1 \leq i \leq n + m - 1, 1 \leq p \leq 4\}$$

The input sequences $A$ and $B$ are built by concatenating several substrings.

For each $v_i \in V$, we define the following substrings of the input sequences $A$, $B$:

$$A(v_i) = y_{i,1} y_{i,2} x_{i,1} x_{i,2} x_{i,3} \qquad B(v_i) = x_{i,1} x_{i,2} x_{i,3} y_{i,1} y_{i,2}$$

For each $\{v_i, v_j\} \in E$, with $i < j$ (which is the $p$-th edge, $1 \leq p \leq 3$, incident on $v_i$ and the $q$-th edge, $1 \leq q \leq 3$, incident on $v_j$), define the following substrings of $A$, $B$:

$$A(\{v_i, v_j\}) = x_{i,p} x_{j,q} \qquad B(\{v_i, v_j\}) = x_{j,q} x_{i,p}$$

Finally, define $2(n+m-1)$ additional substrings $S_{A,1}, S_{A,2}, \ldots, S_{A,m+n-1}$, $S_{B,1}, S_{B,2}, \ldots, S_{B,m+n-1}$ where $S_{A,i}$, $S_{B,i}$, with $1 \leq i \leq m+n-1$, are defined as follows: $S_{A,i} = S_{B,i} = z_{i,1} z_{i,2} z_{i,3} z_{i,4}$.

Now, we are able to define the input sequences $A$ and $B$, by concatenating the substrings previously defined, where substrings associated with edges of $G$ are concatenated assuming some edge ordering (we assume that $\{v_1, v_w\}$ is the first edge, while $\{v_r, v_t\}$ is the last edge according to the ordering):

$$A = A(v_1) \cdot S_{A,1} \cdot A(v_2) \cdot \cdots \cdot S_{A,n-1} \cdot A(v_n) \cdot S_{A,n} \cdot A(\{v_1, v_w\}) \cdot \cdots \cdot S_{A,n+m-1} \cdot A(\{v_r, v_t\})$$

$$B = B(v_1) \cdot S_{B,1} \cdot B(v_2) \cdot \cdots \cdot S_{B,n-1} \cdot B(v_n) \cdot S_{B,n} B(\{v_1, v_w\}) \cdot \cdots \cdot S_{B,n+m-1} \cdot B(\{v_r, v_t\})$$

Notice that each substring associated with an edge $\{v_i, v_j\}$ appears exactly once in both $A$ and $B$.

The multiset $\mathcal{M}$ is defined as follows: $\mathcal{M} = \{x_{i,t} : v_i \in V, 1 \leq t \leq 3\}$.

First, we prove that $(A, B, \mathcal{M})$ is an instance of 2-$\mathcal{LFCS}$, that is we prove that each symbol has at most two occurrences in $A$.

**Lemma 2.** *Each symbol of $\Sigma$ occurs at most twice in $A$.*

**Proof.** Notice that each symbol that appears in a substring $S_{A,i}$, $1 \leq i \leq m+n-1$, does not appear in any other subsequence of $A$. Now, consider a symbol $y_{i,t}$, with $1 \leq i \leq n$ and $1 \leq t \leq 2$, that occurs in substring $A(v_i)$; $y_{i,t}$ does not appear in any other substring of $A$. Finally, consider a symbol $x_{i,t}$, with $1 \leq i \leq n$ and $1 \leq t \leq 3$; $x_{i,t}$ has one occurrence in exactly two subsequences of $A$: subsequence $A(v_i)$ and subsequence $A(\{v_i, v_j\})$ (where $\{v_i, v_j\}$ is the $t$-th edges incident on $v_i$). □

Now, we present an outline of the reduction. First, we prove that a solution of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$ matches by alignment each position of $S_{A,i}$ with a position of $S_{B,i}$ (Lemma 3). Then, we show that the possible matches of each subsequence $A(v_i)$ can be essentially of two kinds: an *I-configuration* (related to a vertex in an independent set of the graph $G$) and a *C-configuration* (related to a vertex not in an independent set of the graph $G$).

Let $B^*$ be a solution of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$. We denote by $S_{B^*,i}$ ($B^*(v_i)$, $B^*(\{v_i, v_j\})$, respectively), the substring of a solution $B^*$ corresponding (possibly after some insertion) to the substring $S_{B,i}$ ($B(v_i)$, $B(\{v_i, v_j\})$, respectively), of $B$.

Next, we show that we can assume that in a solution $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$, a longest common subsequence of $A$ and $B^*$ matches by alignment a position of a subsequence $S_{A,i}$, $1 \leq i \leq m+n-1$, only with a position of $S_{B^*,i}$, $1 \leq i \leq m+n-1$.

**Lemma 3.** *Given a cubic graph $G$, let $(A, B, \mathcal{M})$ be the corresponding instance of 2-$\mathcal{LFCS}$, and $B^*$ a solution of 2-$\mathcal{LFCS}$ over $(A, B, \mathcal{M})$. Then a longest common subsequence of $A$ and $B^*$ contains each symbol $z_{t,q}$, with $1 \leq t \leq m+n-1$ and $1 \leq q \leq 4$.*

[1] We recall that a cubic graph is an undirected graph where each vertex has degree exactly three.

**Proof.** Consider a solution $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$ and assume that it does not contain a symbol $z_{t,q}$, with $1 \leq t \leq m+n-1$ and $1 \leq q \leq 4$.

First, observe that by construction a longest common subsequence of $B^*$ and $A$ matches by alignment a position of $A(v_i)$ either with a position of $B(v_i)$ or with a position of $B(\{v_i, v_j\})$. We prove that a longest common subsequence between $A$ and $B^*$ matches by alignment a position of $A(v_i)$ only with a position of $B(v_i)$. Assume that $i$ is the minimum value such that a longest common subsequence $S$ of $A$ and $B^*$ matches by alignment a position of $A(v_i)$ and a position of $B^*(\{v_i, v_j\})$, then, by construction of $(A, B, \mathcal{M})$, no position of $S_{A,i}$ can be matched. Now, starting from $S$ we can compute a common subsequence $S'$ of $A$ and $B^*$, with $|S'| > |S|$, by modifying the alignment of $S$ as follows: (i) match by alignment the positions of $A(v_i)$ and the positions of $B^*(v_i)$ containing symbols $y_{i,1}$, $y_{i,2}$; (ii) match by alignment the positions of subsequences $S_{A,i}$ containing symbol $z_{i,q}$, with $1 \leq q \leq 4$, with position of subsequences $S_{B,i}$ containing symbol $z_{i,q}$; (iii) any other match is not modified. It follows that the number of positions in $A(v_i)$ matched by $S'$ with respect to $S$ is decreased by at most three, since eventually positions of $A(v_i)$ containing symbols $x_{i,1}$, $x_{i,2}$, $x_{i,3}$ will not be matched. The number of positions in $S_{A,i}$ matched by $S'$ with respect to $S$ is increased by at least 4, since each position of $S_{A,i}$ is matched by $S'$ and not by $S$. By iterating this procedure, we eventually find a longest common subsequence $S'$ of $A$ and $B^*$, where if a position of $A(v_i)$ is matched by alignment, then it is matched with a position of $B(v_i)$. Moreover, since $|A(\{v_i, v_j\})| = 2$, with $\{v_i, v_j\} \in E$, while $|S_{A,t}| = 4$, we can assume that each position of $A(\{v_i, v_j\})$ is possibly matched by alignment only with positions of $B(\{v_i, v_j\})$.

By the maximality of $S'$, this implies that each position of $A$ containing a symbol $z_{t,q}$, with $1 \leq t \leq m+n-1$ and $1 \leq q \leq 4$, matches a position of $B^*$ containing symbol $z_{t,q}$. □

Consider a vertex $v_i \in V$ and the corresponding substrings $A(v_i)$, $B(v_i)$ of $A$ and $B$. Moreover, let $\{v_i, v_j\}, \{v_i, v_h\}, \{v_i, v_z\} \in E$ be the three edges of $G$ incident on $v_i$ and consider the corresponding substrings $A(\{v_i, v_j\})$, $A(\{v_i, v_h\})$, $A(\{v_i, v_z\})$ ($B(\{v_i, v_j\})$, $B(\{v_i, v_h\})$, $B(\{v_i, v_z\})$, respectively), of $A$ (of $B$, respectively). Informally, the reduction shows that there are essentially two possible configurations (called *I-configuration* and *C-configuration*) of the substring $B^*(v_i)$ (and possibly $B^*(\{v_i, v_j\})$, $B^*(\{v_i, v_h\})$ and $B^*(\{v_i, v_z\})$) of a filling $B^*$ of $B$. A substring $B^*(v_i)$ having an I-configuration is related to the vertex $v_i$ in an independent set of $G$, while a substring $B^*(v_i)$ having a C-configuration is related to the vertex $v_i$ in a vertex cover of $G$. We define now the two possible configurations. An *I-configuration* for the substrings $B^*(v_i)$, $B^*(\{v_i, v_j\})$, $B^*(\{v_i, v_h\})$ and $B^*(\{v_i, v_z\})$ is defined as follows:

- $B^*(v_i) = B(v_i)$ (hence there is no insertion in $B(v_i)$).
- For each $\{v_i, v_t\}$, with $t \in \{j, h, z\}$, where $\{v_i, v_t\}$ is the $p$-th edge incident on $v_i$, $1 \leq p \leq 3$, and the $q$-th edge incident on $v_t$, $1 \leq q \leq 3$, $B^*(\{v_i, v_t\}) = x_{i,p} x_{j,q} x_{i,p}$ (hence $x_{i,p}$ is inserted in $B(v_i)$).

If $B^*(v_i)$, $B^*(\{v_i, v_j\})$, $B^*(\{v_i, v_h\})$, $B^*(\{v_i, v_z\})$ have an *I-configuration*, a longest common subsequence of $B^*(v_i)$ and $A(v_i)$ has length three (it matches the positions containing $x_{i,1}$, $x_{i,2}$, $x_{i,3}$), and a longest common subsequence of $A(\{v_i, v_t\})$ and $B^*(\{v_i, v_t\})$, with $t \in \{j, h, z\}$, has length two (it matches the positions containing $x_{i,p}$, $x_{j,q}$).

A *C-configuration* for the substring $B^*(v_i)$ is defined as follows:

- $B^*(v_i) = x_{i,1} x_{i,2} x_{i,3} y_{i,1} y_{i,2} x_{i,1} x_{i,2} x_{i,3}$ (hence $B^*(v_i) = B(v_i) \cdot x_{i,1} x_{i,2} x_{i,3}$).

If $B^*(v_i)$ has a *C-configuration*, a longest common subsequence of $B^*(v_i)$ and $A(v_i)$ has length five, it matches the positions containing $y_{i,1}$, $y_{i,2}$, $x_{i,1}$, $x_{i,2}$, $x_{i,3}$.

Next, we present the main lemmata of this section.

**Lemma 4.** *Let $G$ be a cubic graph, instance of Max-ISC, and let $(A, B, \mathcal{M})$ be the corresponding instance of 2-$\mathcal{LFCS}$. Then, given an independent set $I$ of $G$ of size $k$, we can compute in polynomial time a solution $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$ inducing a longest common subsequence with $A$ of length $4(m+n-1) + 6|I| + 5|V \setminus I| + |E|$.*

**Proof.** Consider an independent set $I$ of $G$ and the corresponding instance $(A, B, \mathcal{M})$ of 2-$\mathcal{LFCS}$. Define a solution $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$ as follows. For each $v_i \in I$, where $\{v_i, v_j\}, \{v_i, v_h\}, \{v_i, v_z\} \in E$ are the three edges of $G$ incident on $v_i$, define an *I-configuration* for $B^*(v_i)$, $B^*(\{v_i, v_j\})$, $B^*(\{v_i, v_h\})$, $B^*(\{v_i, v_z\})$. For each $v_i \in V \setminus I$, define a *C-configuration* for $B^*(v_i)$. For each edge $\{v_i, v_j\} \in E$ if $v_i, v_j \in V \setminus I$, then $B^*(\{v_i, v_j\}) = B(\{v_i, v_j\})$; notice that in this case a longest common subsequence of $A(\{v_i, v_j\})$ and $B^*(\{v_i, v_j\})$ has length one, as it matches exactly one position containing either $x_{i,p}$ or $x_{j,q}$. Finally, each position of $A$ in the substring $S_{A,i}$, with $1 \leq i \leq m+n-1$, is matched by alignment with the corresponding position of $S_{B^*,i}$.

Notice that the solution $B^*$ is well-defined, as each $B^*(\{v_i, v_j\})$, with $\{v_i, v_j\} \in E$, can belong to an *I-configuration* of at most one of $B^*(v_i)$ and $B^*(v_j)$, since at most one of $v_i$, $v_j$ belongs to $I$.

Now, consider a longest common subsequence $S$ of $A$ and $B^*$. $S$ matches $4(m+n-1)$ positions in substrings $S_{A,1}$, …, $S_{A,m+n-1}$, since all the positions of these substrings are matched and, by construction, the overall length of $S_{A,1}$, …, $S_{A,m+n-1}$ is $4(m+n-1)$. By definition of *I-configuration* for each $v_i \in I$, $S$ matches 3 positions of $A(v_i)$ and 2 positions of

each $A(\{v_i, v_j\})$, with $\{v_i, v_j\} \in E$. By definition of *C-configuration*, for each $v_i \in V \setminus I$, $S$ matches 5 positions of $A(v_i)$; for each $\{v_i, v_j\} \in E$, with $v_i, v_j \in V \setminus I$, $S$ matches one position of $A(\{v_i, v_j\})$. Hence, $S$ matches $4(m+n-1)+6|I|+5|V \setminus I|+|E|$ positions of $A$ and $B^*$. □

Based on Lemma 3, we can prove the following result.

**Lemma 5.** *Let $G$ be a cubic graph, instance of Max-ISC, and let $(A, B, \mathcal{M})$ be the corresponding instance of 2-$\mathcal{LFCS}$. Then, given a solution $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$ of length $4(m+n-1)+6p+5(|V|-p)+|E|$, we can compute in polynomial time an independent set of $G$ of size at least $p$.*

**Proof.** Given an instance $B^*$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$, by Lemma 3 we can assume that each position of $A$ in the substring $S_{A,i}$ is matched by alignment with a position in the substring $S_{B,i}$ of a solution $B^*$. Hence, it follows that the positions of $A(v_i)$ can be matched only with positions of $B^*(v_i)$ and that the positions of $A(\{v_i, v_j\})$ can be matched only with positions of $B^*(\{v_i, v_j\})$.

Now, assume that there exist two subsequences in $B^*$, say $B^*(v_i)$ and $B^*(v_j)$, that are both equal to $B(v_i)$ and $B(v_j)$ (that is no position of $A(v_i)$ and $A(v_j)$ is matched by insertion), where $\{v_i, v_j\} \in E$. Moreover, let $\{v_i, v_j\}$ be the $p$-th edge incident on $v_i$ and the $q$-th on $v_j$, $1 \leq p, q \leq 3$. By construction, since $A(\{v_i, v_j\})$ has length 2, a longest common subsequence of $A$ and $B^*$ can match at most two positions of $A(\{v_i, v_j\})$ (with positions of $B^*(\{v_i, v_j\})$). We can compute a solution $B'$ of 2-$\mathcal{LFCS}$ over instance $(A, B, \mathcal{M})$, such that $B'$ induces with $A$ a longest common subsequence not shorter than that induced by $B^*$ with the following properties: at least one of $B'(v_i)$, $B'(v_j)$ is associated with a *C-configuration*, assume w.l.o.g. $B'(v_i)$, while $B'(v_j)$ has an *I-configuration*. It follows that $B'(v_i) = x_{i,1}x_{i,2}x_{i,3}y_{i,1}y_{i,2}x_{i,1}x_{i,2}x_{i,3}$, and 5 positions of $A(v_i)$ are matched with 5 positions of $B'(v_i)$ (notice that at most 3 positions of $A(v_i)$ are matched with 3 positions of $B^*(v_i)$ since $B^*(v_i)$ is identical to $B(v_i)$). Notice that both positions of $A(\{v_i, v_j\})$ can be matched with positions of $B'(\{v_i, v_j\})$: by alignment the position having value $x_{i,p}$ and by insertion the position having value $x_{j,q}$ (due to the *I-configuration* of $B'(v_j)$). The number of positions matched by a longest common subsequence of $A$ and $B'$ with respect to a longest common subsequence of $A$ and $B^*$ is decreased by at most 1 for each of the substring $A(\{v_i, v_h\})$ and $A(\{v_i, v_z\})$, with $\{v_i, v_h\}, \{v_i, v_z\} \in E$, and it is increased by at least 2, for $A(v_i)$. Hence, in the following we can assume that if $B^*(v_i)$ and the substrings of $B^*$ associated with the edges incident in $v_i$ have an *I-configuration*, then for each $\{v_i, v_j\} \in E$, $B^*(v_j)$ has a *C-configuration*.

We can assume that if $B^*(v_i)$ has not an *I-configuration*, then $B^*(v_i)$ has a *C-configuration*. Indeed, in a *C-configuration* of $B^*(v_i)$, each position where a symbol $x_{i,t}$, with $1 \leq t \leq 3$, is inserted is matched by insertion and this implies that the number of matched positions in a longest common subsequence of $A(v_i)$ and $B^*(v_i)$ (with a *C-configuration*) is 5. Any other filling $B^*(v_i)$, $B^*(\{v_i, v_j\})$, $B^*(\{v_i, v_h\})$, $B^*(\{v_i, v_z\})$, different from an *I-configuration* and a *C-configuration*, can match either: (1) at most two positions with symbols $y_{i,t}$, $1 \leq t \leq 2$, of $A(v_i)$ and at most three with symbols $x_{i,t}$, $1 \leq t \leq 3$, in $\{A(v_i), A(\{v_i, v_j\}), A(\{v_i, v_h\}), A(\{v_i, v_z\})\}$ or (2) by alignment at most three positions with symbols $x_{i,t}$, $1 \leq t \leq 3$, of $A(v_i)$, and at most two positions by insertion with symbols $x_{i,t}$, $1 \leq t \leq 3$, of $\{A(\{v_i, v_j\}), A(\{v_i, v_h\}), A(\{v_i, v_z\})\}$, otherwise it is *I-configuration*. Hence the number of matched positions containing symbols $y_{i,t}$, $1 \leq t \leq 2$, and $x_{i,t}$, $1 \leq t \leq 3$, is not increased with respect to a *C-configuration*.

Then, we can define an independent set of the graph $G$ as follows:

$$I = \{v : i \in I : B^*(v_i) \text{ has an } \textit{I-configuration}\}$$

Since we have shown that if $B^*(v_i)$ and the substrings of $B^*$ associated with the edges incident in $v_i$ have an *I-configuration*, then for each $\{v_i, v_j\} \in E$, $B^*(v_j)$ has a *C-configuration*, it follows that $I$ is an independent set. A longest common subsequence $S$ of $A$ and $B^*$ matches $4(m+n-1)$ positions in sequences $S_{A,1}, \ldots, S_{A,m+n-1}$; for each $B^*(v_i)$ associated with an *I-configuration*, $S$ matches 3 positions of $A(v_i)$ and 2 positions of each $A(\{v_i, v_j\})$, with $\{v_i, v_j\} \in E$; for each $B^*(v_i)$ associated with a *C-configuration*, $S$ matches 5 positions of $A(v_i)$. Finally, $S$ matches 1 position for each $A(\{v_i, v_t\})$ not included in an *I-configuration*. It follows that $|I| \geq p$. □

By Lemmata 4 and 5, and by the APX-hardness of Max-ISC [19] we can conclude that the 2-$\mathcal{LFCS}$ problem is APX-hard.

**Theorem 6.** *2-$\mathcal{LFCS}$ is APX-hard.*

**Proof.** By Lemma 4 and Lemma 5, and since in a cubic graph $|E| = \frac{3}{2}|V|$ and $|I| \geq \frac{1}{4}|V|$, it follows that we have designed and L-reduction from Max-ISC to 2-$\mathcal{LFCS}$ (see [20]). Since Max-ISC is APX-hard [19], it follows that 2-$\mathcal{LFCS}$ is APX-hard. □

## 4. Approximating $\mathcal{LFCS}$

In this section we give a polynomial-time approximation algorithm for $\mathcal{LFCS}$ of factor $\frac{3}{5}$. The approximation algorithm picks the largest number of matched positions returned by two polynomial-time algorithms, Approx-Algorithm-1 and

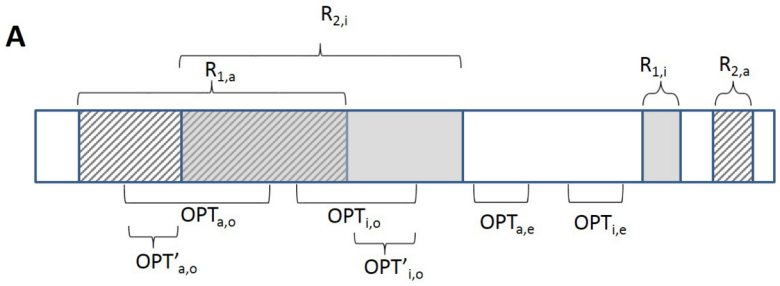


**Fig. 3.** The input sequence $A$ and the positions matched by solution $R_1$ (dashed) and by solution $R_2$ (in gray). In the upper part, brackets represent the subsets $R_{1,a}$ and $R_{1,i}$ of $R_1$, and $R_{2,a}$ and $R_{2,i}$ of $R_2$. In the lower part, the brackets represent the positions matched by $OPT$.

Approx-Algorithm-2. Notice that each algorithm does not return a filling of $B$ with $\mathcal{M}$, but two disjoint subsets of positions of $A$ that have to be matched by alignment and by insertion, respectively, by a subsequence of $A$ and of a filling of $B$ with $\mathcal{M}$. We can easily compute in polynomial time a filling $B^*$ of $B$ with $\mathcal{M}$ so that there exists a common subsequence of $A$ and $B^*$ that matches these two subsets of positions.

Both algorithms consist of two phases.

**Approx-Algorithm-1** In the first phase, Approx-Algorithm-1 computes in polynomial time a longest common subsequence of $A$ and $B$. Denote by $R_{1,a}$ the positions of $A$ matched by alignment in the first phase and by $A'$ the subsequence of $A$ obtained by removing the positions of $R_{1,a}$. The second phase greedily computes in polynomial time a set $R_{1,i}$ of positions of $A'$ of maximum size that matches $\mathcal{M}$ by insertion, applying Algorithm 1 on $(A', \mathcal{M})$. Denote by $R_1 = R_{1,a} \cup R_{1,i}$ the set of positions returned by Approx-Algorithm-1. 先找alignment後找insertion

**Approx-Algorithm-2** In the first phase, Approx-Algorithm-2 computes a subset $R_{2,i}$ of positions of $A$ of maximum size that matches $\mathcal{M}$ by insertion by applying Algorithm 1 on $(A, \mathcal{M})$. Denote by $A''$ the subsequence of $A$ obtained by removing the positions of $R_{2,i}$. The second phase computes a longest common subsequence of $B$ and $A''$; denote by $R_{2,a}$ the set of positions of $A''$ (and $A$) matched by this phase. Denote by $R_2 = R_{2,a} \cup R_{2,i}$ the set of positions returned by Approx-Algorithm-2. 先找insertion後找alignment

Next, we show that the maximum number of positions matched by one of Approx-Algorithm-1 and Approx-Algorithm-2 gives a $\frac{3}{5}$-approximated solution. First, we introduce some notations (see Fig. 3). Let $B^{opt}$ be an optimal solution of $\mathcal{LFCS}$ on instance $(A, B, \mathcal{M})$ ($B^{opt}$ is a filling of $B$ with $\mathcal{M}$), and let $OPT$ be a longest common subsequence of $A$ and $B^{opt}$. We consider the following sets of positions of $OPT$. Denote by $OPT_a$ the set of positions of $A$ matched by alignment in $OPT$ and by $OPT_i$ the set of positions of $A$ matched by insertion in $OPT$. Notice that by construction it holds $OPT_a \cap OPT_i = \emptyset$.

Define $OPT_{a,o} = OPT_a \cap (R_{1,a} \cup R_{2,i})$ and $OPT_{i,o} = OPT_i \cap (R_{1,a} \cup R_{2,i})$. Informally, $OPT_{a,o}$ ($OPT_{i,o}$, respectively) is the set of positions of $A$ matched in the first phase of Approx-Algorithm-1 or Approx-Algorithm-2 that are matched by alignment (by insertion, respectively) in $OPT$.

Define $OPT_{a,e} = OPT_a \setminus OPT_{a,o}$ and $OPT_{i,e} = OPT_i \setminus OPT_{i,o}$. Finally, define $OPT'_{i,o} = OPT_{i,o} \setminus R_{1,a}$ and $OPT'_{a,o} = OPT_{a,o} \setminus R_{2,i}$.

By definition of $OPT$, $OPT_{a,o}$, $OPT_{i,o}$, $OPT_{a,e}$ and $OPT_{i,e}$, it holds $|OPT| = |OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,o}| + |OPT_{i,e}|$.

We will show that the largest of $R_1$ and $R_2$ gives a $\frac{3}{5}$-approximate solution, that is $\max(|R_1|, |R_2|) \geq \frac{3}{5}|OPT|$. We start by showing two bounds on $OPT_i$ and $OPT_a$.

**Lemma 7.** *$|R_{1,a}| \geq |OPT_a|$ and $|R_{2,i}| \geq |OPT_i|$.*

**Proof.** First, we prove that $|R_{1,a}| \geq |OPT_a|$. Consider the set of positions in $OPT_a$. Since each position in $OPT_a$ is a position of $A$ matched by alignment, it follows that the set $OPT_a$ induces a common subsequence of $A$ and $B$. Since the set $R_{1,a}$ of positions of $A$ induces a longest common subsequence of $A$ and $B$, it follows that $|R_{1,a}| \geq |OPT_a|$.

Now, we prove that $|R_{2,i}| \geq |OPT_i|$. Consider the set of positions in $OPT_i$. Each position in $OPT_i$ is matched by insertion, hence it is matched with an inserted symbol of $\mathcal{M}$. By Lemma 1, $R_{2,i}$ is a set of positions of $A$ of maximum cardinality that can be matched by insertion with symbols of $\mathcal{M}$, hence $|R_{2,i}| \geq |OPT_i|$. □

As a consequence of Lemma 7, it follows that $|R_{1,a}| + |R_{2,i}| \geq |OPT_i| + |OPT_a| \geq |OPT|$. Hence the maximum of $|R_1|$, $|R_2|$ is (at least) $\frac{1}{2}|OPT|$. In the following, we show with a more refined analysis that the maximum of $|R_1|$, $|R_2|$ is at least $\frac{3}{5}|OPT|$.

We start by proving some bounds on $|R_{1,i}|$ and $|R_{2,a}|$, then we consider three cases depending on the values of $|OPT_{a,o}|$, $|OPT_{i,o}|$, $|OPT_{a,e}|$, $|OPT_{i,e}|$, $|OPT'_{i,o}|$ and $|OPT'_{a,o}|$. First, the following result holds.

**Lemma 8.** $|R_{1,i}| \geq |OPT'_{i,o}| + |OPT_{i,e}|$ *and* $|R_{2,a}| \geq |OPT'_{a,o}| + |OPT_{a,e}|$.

**Proof.** We start by showing that $|R_{1,i}| \geq |OPT'_{i,o}| + |OPT_{i,e}|$. Consider the subsequences of $A'$ obtained by removing the set of positions $R_{1,a}$ matched in the first phase of Approx-Algorithm-1. By Lemma 1, $R_{1,i}$ is a set of positions of $A'$ of maximum cardinality matched by insertion with $\mathcal{M}$. Since $OPT'_{i,o}$, $OPT_{i,e}$ are disjoint and $OPT'_{i,o} \cup OPT_{i,e}$ is a set of positions of $A'$ matched by insertion, the first part of the lemma follows.

Now, we show that $|R_{2,a}| \geq |OPT'_{a,o}| + |OPT_{a,e}|$. Consider the subsequence $A''$ obtained from $A$ by removing the set $R_{2,i}$ of positions matched by insertion in the first phase of Approx-Algorithm-2. Since by construction $OPT'_{a,o}$, $OPT_{a,e}$ are disjoint and $OPT'_{a,o} \cup OPT_{a,e}$ is a set of positions of $A''$ matched by alignment by a subsequence of $A$ and $B^{opt}$, and since by construction $R_{2,a}$ is a set of positions of $A''$ of maximum size matched by alignment with positions of $B$, the second part of the lemma follows. □

Now, in the analysis of the approximation factor of Approx-Algorithm-1 and Approx-Algorithm-2, we consider three cases, depending on the values of $OPT_{i,e}$, $OPT_{i,o}$, $OPT'_{i,o}$.

*Case 1.*

Assume that $|OPT_{i,e}| + |OPT'_{i,o}| \geq \frac{1}{2}|OPT_{i,o}|$, we show the following result.

**Lemma 9.** *Assume that* $|OPT_{i,e}| + |OPT'_{i,o}| \geq \frac{1}{2}|OPT_{i,o}|$, *then* $|R_1| \geq \frac{3}{5}|OPT|$.

**Proof.** By Lemma 8 it holds that $|R_{1,i}| \geq |OPT'_{i,o}| + |OPT_{i,e}|$, hence

$$|R_{1,a}| + |R_{1,i}| \geq |R_{1,a}| + |OPT'_{i,o}| + |OPT_{i,e}| \geq$$
$$\frac{3}{5}(|R_{1,a}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT_{i,e}|) + |OPT'_{i,o}|.$$

By Lemma 7 it follows that $|R_{1,a}| \geq |OPT_a|$ and, since $|OPT_a| = |OPT_{a,o}| + |OPT_{a,e}|$, it follows that $|R_{1,a}| \geq |OPT_{a,o}| + |OPT_{a,e}|$, hence

$$\frac{3}{5}(|R_{1,a}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT_{i,e}|) + |OPT'_{i,o}| \geq$$
$$\frac{3}{5}(|OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT_{i,e}|) + |OPT'_{i,o}|.$$

Hence, it holds

$$|R_{1,a}| + |R_{1,i}| \geq \frac{3}{5}(|OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT_{i,e}|) + |OPT'_{i,o}|. \quad (1)$$

Notice that $|R_{1,a}| + |OPT'_{i,o}| \geq |OPT_{i,o}|$, since, by construction, each position in $OPT_{i,o}$ is either in $OPT'_{i,o}$ or in $R_{1,a}$. Then,

$$\frac{2}{5}(|R_{1,a}| + |OPT'_{i,o}|) \geq \frac{2}{5}|OPT_{i,o}|. \quad (2)$$

Since we are assuming that $|OPT_{i,e}| + |OPT'_{i,o}| \geq \frac{1}{2}|OPT_{i,o}|$, it holds

$$\frac{2}{5}(|OPT_{i,e}| + |OPT'_{i,o}|) \geq \frac{1}{5}|OPT_{i,o}|. \quad (3)$$

Combining Inequalities 2 and 3 with Inequality 1, we can conclude that, under the hypothesis $|OPT_{i,e}| + |OPT'_{i,o}| \geq \frac{1}{2}|OPT_{i,o}|$, it holds

$$|R_{1,a}| + |R_{1,i}| \geq \frac{3}{5}(|OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT_{i,e}|) + |OPT'_{i,o}| \geq$$
$$\frac{3}{5}(|OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,e}|) + \frac{2}{5}(|R_{1,a}| + |OPT'_{i,o}|) + \frac{2}{5}(|OPT_{i,e}| + |OPT'_{i,o}|) \geq$$
$$\frac{3}{5}(|OPT_{a,o}| + |OPT_{a,e}| + |OPT_{i,o}| + |OPT_{i,e}|).$$

It follows that, under the hypothesis $|OPT_{i,e}| + |OPT'_{i,o}| \geq \frac{1}{2}|OPT_{i,o}|$, it holds $|R_1| \geq \frac{3}{5}|OPT|$. □

*Case 2.*

Assume that $|OPT_{a,e}| + |OPT'_{a,o}| \geq \frac{1}{2}|OPT_{a,o}|$. Similarly to Case 1, we can prove the following result.

**Lemma 10.** *Assume that* $|OPT_{a,e}| + |OPT'_{a,o}| \geq \frac{1}{2}|OPT_{a,o}|$, *then* $|R_2| \geq \frac{3}{5}|OPT|$.

**Proof.** By Lemma 8 it follows that $|R_{2,a}| \geq |OPT'_{a,o}| + |OPT_{a,e}|$, hence it holds

$$|R_2| = |R_{2,i}| + |R_{2,a}| \geq |R_{2,i}| + |OPT'_{a,o}| + |OPT_{a,e}| \geq$$
$$\frac{3}{5}(|R_{2,i}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT_{a,e}|) + |OPT'_{a,o}|.$$

By Lemma 7, it follows that $|R_{2,i}| \geq |OPT_i|$ and, since $|OPT_i| = |OPT_{i,o}| + |OPT_{i,e}|$, it follows that $|R_{2,i}| \geq |OPT_{i,o}| + |OPT_{i,e}|$, hence

$$\frac{3}{5}(|R_{2,i}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT_{a,e}|) + |OPT'_{a,o}| \geq$$
$$\frac{3}{5}(|OPT_{i,o}| + |OPT_{i,e}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT_{a,e}|) + |OPT'_{a,o}|.$$

Hence, the following inequality holds:

$$|R_{2,i}| + |R_{2,a}| \geq \frac{3}{5}(|OPT_{i,o}| + |OPT_{i,e}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT_{a,e}|) + |OPT'_{a,o}|. \tag{4}$$

By construction $|R_{2,i}| + |OPT'_{a,o}| \geq |OPT_{a,o}|$, since each position of $OPT_{a,o}$ is either in $R_{2,i}$ or in $OPT'_{a,o}$. Then,

$$\frac{2}{5}(|R_{2,i}| + |OPT'_{a,o}|) \geq \frac{2}{5}|OPT_{a,o}|. \tag{5}$$

Since we are assuming that $|OPT'_{a,o}| + |OPT_{a,e}| \geq \frac{1}{2}|OPT_{a,o}|$, it holds

$$\frac{2}{5}(|OPT'_{a,o}| + |OPT_{a,e}|) \geq \frac{1}{5}|OPT_{a,o}|. \tag{6}$$

Combining Inequalities 5, 6 with Inequality 4, we can conclude that, under the hypothesis $|OPT'_{a,o}| + |OPT_{a,e}| \geq \frac{1}{2}|OPT_{a,o}|$, it holds

$$|R_{2,i}| + |R_{2,a}| \geq \frac{3}{5}(|OPT_{i,o}| + |OPT_{i,e}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT_{a,e}|) + |OPT'_{a,o}| \geq$$
$$\frac{3}{5}(|OPT_{i,o}| + |OPT_{i,e}| + |OPT_{a,e}|) + \frac{2}{5}(|R_{2,i}| + |OPT'_{a,o}|) + \frac{2}{5}(|OPT_{a,e}| + |OPT'_{a,o}|) \geq$$
$$\frac{3}{5}(|OPT_{i,o}| + |OPT_{i,e}| + |OPT_{a,e}| + |OPT_{a,o}|).$$

It follows that, under the hypothesis $|OPT'_{a,o}| + |OPT_{a,e}| \geq \frac{1}{2}|OPT_{a,o}|$, it holds $|R_2| \geq \frac{3}{5}|OPT_{a,o}|$. □

*Case 3.*

Assume that both Case 1 and Case 2 do not hold. Then,

$$|OPT_{i,e}| + |OPT'_{i,o}| < \frac{1}{2}|OPT_{i,o}| \text{ and } |OPT_{a,e}| + |OPT'_{a,o}| < \frac{1}{2}|OPT_{a,o}|.$$

Since $|OPT_{i,e}| + |OPT'_{i,o}| < \frac{1}{2}|OPT_{i,o}|$, it follows that $|OPT_{i,e}| < \frac{1}{2}|OPT_{i,o}|$ and, since $|OPT_{a,e}| + |OPT'_{a,o}| < \frac{1}{2}|OPT_{a,o}|$, it follows that $|OPT_{a,e}| < \frac{1}{2}|OPT_{a,o}|$. But then, since $|OPT| = |OPT_{a,o}| + |OPT_{i,o}| + |OPT_{a,e}| + |OPT_{i,e}|$, it follows that

$$|OPT| \leq \frac{3}{2}(|OPT_{a,o}| + |OPT_{i,o}|)$$

We show that $|R_1| \geq |OPT_{a,o}| + |OPT_{i,o}|$, thus implying that $|R_1| \geq \frac{3}{5}|OPT|$.

**Lemma 11.** $|R_{1,a} \cup R_{1,i}| \geq |OPT_{a,o}| + |OPT_{i,o}|$.

**Proof.** Consider the sequence $A'$ obtained by removing the set $R_{1,a}$ of matched positions in the first phase of Approx-Algorithm-1. Now, consider the position of $R_{2,i}$. It follows that the positions of $R_{2,i} \setminus R_{1,a}$ belongs to $A'$ and thus can be aligned by insertion in the second phase of Approx-Algorithm-1. Since $R_{1,i}$ is a set of positions of $A'$ having maximum size that matches (by insertion) $\mathcal{M}$, it holds $|R_{1,i}| \geq |R_{2,i} \setminus R_{1,a}|$, and, since $R_{1,a} \cap R_{1,i} = \emptyset$, it holds $|R_{1,a} \cup R_{1,i}| \geq |R_{1,a} \cup R_{2,i}|$. By construction $|R_{1,a} \cup R_{2,i}| \geq |OPT_{a,o}| + |OPT_{i,o}|$, hence $|R_{1,a} \cup R_{1,i}| \geq |OPT_{a,o}| + |OPT_{i,o}|$. □

By Lemma 11, $|R_{1,a} \cup R_{1,i}| \geq |OPT_{a,o}| + |OPT_{i,o}|$. Since in this case we have shown that $|OPT| \leq \frac{3}{2}(|OPT_{a,o}| + |OPT_{i,o}|)$, it follows that $|R_1| = |R_{1,a} \cup R_{1,i}| \geq \frac{2}{3}|OPT| \geq \frac{3}{5}|OPT|$. From Lemma 9, Lemma 10 and Lemma 11, it follows the main result of this section.

**Theorem 12.** *Given an instance $(A, B, \mathcal{M})$ of $\mathcal{LFCS}$, the largest solution returned by Approx-Algorithm-1 and Approx-Algorithm-2 is an approximate solution of factor $\frac{3}{5}$.*

**Proof.** From Lemma 9, Lemma 10 and Lemma 11, it follows that $\max(|R_1|, R_2|) \geq \frac{3}{5}|OPT|$.

We can compute a filling $B_1$ of $B$ with $\mathcal{M}$ that matches at least $|R_1|$ positions with $A$ as follows: we consider the positions in $R_{1,a}$ as matched by alignment, we insert symbols of $\mathcal{M}$ in $B$ in order to match by insertion the positions in $R_{1,i}$. It follows that a longest common subsequence of $A$ and $B_1$ matches at least $|R_1|$ positions.

Similarly, we can compute a filling $B_2$ of $B$ with $\mathcal{M}$ that matches at least $|R_2|$ positions of $A$. We insert symbols of $\mathcal{M}$ in $B$ so that the positions in $R_{1,i}$ are matched by insertion. Consider the subsequence $A''$ obtained after the removal of positions in $R_{1,i}$; a longest common subsequence of $A''$ and $B$ matches at least $|R_{2,a}|$ positions. It follows that a longest common subsequence of $A$ and $B_2$ matches at least $|R_2|$ positions. □

## 5. Exact algorithms

In this section, we give two exact algorithms for $\mathcal{LFCS}$. First, we present an FPT algorithm for $\mathcal{LFCS}$ parameterized by the number $k$ of positions of $A$ matched by insertions. Then, we give an algorithm for $\mathcal{LFCS}$ when the alphabet has constant size. The two algorithms are similar and both based on dynamic programming.

Here we assume that the input sequences $A$ and $B$ have been extended by adding two symbols $\$_A, \$_B \notin \Sigma$, respectively, in position 0 of $A$ and $B$, respectively. Hence we assume that position 0 of $A$ and of a filling $B^*$ of $B$ with $\mathcal{M}$ is not matched by alignment or by insertion by any solution of $\mathcal{LFCS}$ of length greater than zero.

### 5.1. An FPT algorithm

The algorithm we present is based on the color-coding technique [21]. Next, we present the definition of perfect families of hash functions for a multiset of symbols, on which our color-coding approach is based.

**Definition 13.** Let $\mathcal{M}$ be a multiset of positions and let $F$ be a family of hash functions from $\mathcal{M}$ to a set $\{c_1, \ldots, c_k\}$ of colors. $F$ is called *perfect* if for any subset $W \subseteq \mathcal{M}$, such that $|W| = k$, there exists a function $f \in F$ which is injective on $W$.

A perfect family $F$ of hash functions from $\mathcal{M}$ to $\{c_1, \ldots, c_k\}$, having size $O(\log|\mathcal{M}|2^{O(k)})$, can be constructed in time $O(2^{O(k)}|\mathcal{M}|\log|\mathcal{M}|)$ (see [21]).

Consider a perfect family of hash functions $F : \mathcal{M} \to \{c_1, \ldots, c_k\}$. Let $f \in F$ be an injective function, and define $L[i, j, C, l]$, with $C \subseteq \{c_1, \ldots, c_k\}$, $0 \leq i, l \leq |A|$ and $0 \leq j \leq |B|$, as follows:

- $L[i, j, C, l] = 1$ if and only if there exists a common subsequence of $A[0, i]$ and of a filling $B^*$ of $B[0, j]$ with $\mathcal{M}$ having length $l$, such that there exist $|C|$ symbols of $\mathcal{M}$ inserted in $B[0, j]$, each one associated with a distinct color of $C$ and matched by insertion with a position of $A$
- else $L[i, j, C, l] = 0$.

Next, we define the recurrence to compute $L[i, j, C, l]$, where $i \geq 1$ and $j \geq 1$.

$$L[i, j, C, l] = \max \begin{cases} L[i-1, j, C, l] & \text{if } i \geq 1 \\ L[i, j-1, C, l] & \text{if } j \geq 1 \\ L[i-1, j-1, C, l-1] & \text{if } A[i] = B[j] \text{ and } j \geq 1 \\ L[i-1, j, C \setminus \{c\}, l-1] & \text{if } A[i] = \alpha \text{ and there exists} \\ & \alpha \in \mathcal{M} \text{ with } f(\alpha) = c \in C \end{cases} \quad (7)$$

For the base case, since we have extended $A$ and $B$ so that position 0 in $A$ and in the filling of $B$ cannot be matched by insertions or by alignment, it holds $L[0,0,C,l]=1$, if $C=\emptyset$ and $l=0$, else $L[0,0,C,l]=0$. Next, we prove the correctness of the recurrence.

**Lemma 14.** *Let $F:\mathcal{M}\to\{c_1,\ldots,c_k\}$ be a perfect family of hash functions, let $f\in F$ be an injective function and let $C$ be a subset of $\{c_1,\ldots,c_k\}$. Then there exists a common subsequence of length $l$, $l\geq 0$, of $A[0,i]$, $0\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'\subseteq\mathcal{M}$ such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$ if and only if $L[i,j,C,l]=1$.*

**Proof.** We prove the lemma by induction on $i+j$. By construction, the lemma holds in the base case, when $i+j=0$, that is $i=0$ and $j=0$. We assume that the lemma holds for $i+j-1$, and we show that it holds for $i+j$.

First, consider a common subsequence having length $l$ of $A[0,i]$ and of a filling $B[0,j]$, $0\leq i\leq|A|$ and $0\leq j\leq|B|$, with $\mathcal{M}'\neq\emptyset$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$, we prove that $L[i,j,C,l]=1$. 1 2

Assume that $A[i]$ is not matched, or that it is matched by alignment with $B[t]$, with $t<j$. In the former case, there exists a length $l$ common subsequence of $A[0,i-1]$ and of a filling of $B[0,j]$, $0\leq i\leq|A|$ and $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$, hence by induction hypothesis $L[i-1,j,C,l]=1$ and thus, by the first case of Recurrence 7, it holds $L[i,j,C,l]=1$. In the latter case, there exists a common subsequence having length $l$ of $A[0,i]$ and of a filling of $B[0,t]$, $0\leq i\leq|A|$ and $0\leq t<j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$, hence by induction hypothesis $L[i,t,C,l]=1$, with $t<j$, and $L[i,j-1,C,l]=1$. By the second case of Recurrence 7 it follows that $L[i,j,C,l]=1$.

Assume that 3 $A[i]$ is matched by alignment with $B[j]$. It follows that there exists a length $l-1$ common subsequence of $A[0,i-1]$ and of a filling of $B[0,j-1]$, $0\leq i\leq|A|$ and $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$, hence by induction hypothesis $L[i-1,j-1,C,l-1]=1$. Since $A[i]=B[j]$, it follows by the third case of Recurrence 7 that $L[i,j,C,l]=1$.

Now, consider the case that 4 $A[i]$ is matched by insertion with a symbol $\alpha\in\mathcal{M}'$. By our assumption, $\alpha$ is associated with a color $c\in C$ and each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$. We distinguish two cases. If $A[i]$ is matched by insertion with a symbol inserted in $B$ in position $t<j$, then by induction hypothesis there exists a common subsequence having length $l$ of $A[0,i]$, $1\leq i\leq|A|$, and of a filling of $B[0,j-1]$, $0\leq j\leq|B|$, with $\mathcal{M}'$ such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$. Hence, by induction hypothesis $L[i,j-1,C,l]=1$ and by the second case of Recurrence 7 $L[i,j,C,l]=1$. If $A[i]$ is matched by insertion with a symbol inserted in $B$ in position $j$, then there exists a common subsequence having length $l-1$ of $A[0,i-1]$, $1\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'\setminus\{\alpha\}$, such that each symbol of $\mathcal{M}'\setminus\{\alpha\}$ matched by insertion is associated with a distinct color in $C\setminus\{c\}$. Hence, by induction hypothesis $L[i-1,j,C\setminus\{c\},l-1]=1$ and, since $A[i]$ is matched by insertion with a symbol $\alpha\in\mathcal{M}'$ associated with color $c\in C$, it follows by the fourth case of Recurrence 7 that $L[i,j,C,l]=1$.

Now, we prove that if $L[i,j,C,l]=1$ there exists a common subsequence having length $l$ of $A[0,i]$, $0\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$ such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$. First, assume that $L[i-1,j,C,l]=1$. Then, by induction hypothesis, there exists a length $l$ common subsequence of $A[0,i-1]$, $1\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each position of $A$ matched by insertion is associated with a distinct color in $C$. Then, the same result holds also for $A[0,i]$, $B[0,j]$ and $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color of $C$. A similar proof holds when $L[i,j-1,C,l]$.

Assume that $L[i-1,j-1,C,l-1]=1$, with $A[i]=B[j]$. Then, by induction hypothesis, there exists a common subsequence having length $l-1$ of $A[0,i]$, $0\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$. Then, by defining a match by alignment between position $i$ in $A$ and position $j$ in $B$, there exists a length $l$ common subsequence of $A[0,i]$, $0\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$ such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$.

Finally, assume that $L[i-1,j,C\setminus\{c\},l-1]$, with $A[i]=\alpha$, such that there exists $\alpha\in\mathcal{M}'$ associated with color $c\in C$. By induction hypothesis there exists a common subsequence having length $l-1$ of $A[0,i]$, $0\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C\setminus\{c\}$. Then, by defining for position $i$ of $A$ a match by insertion with $c\in C$, it follows that there exists a common subsequence having length $l-1$ of $A[0,i]$, $1\leq i\leq|A|$, and of a filling of $B[0,j]$, $0\leq j\leq|B|$, with $\mathcal{M}'$, such that each symbol of $\mathcal{M}'$ matched by insertion is associated with a distinct color in $C$. □

Now, we are able to prove the main result of this section.

**Theorem 15.** *Let $A$, $B$ be two sequences and $\mathcal{M}$ a multiset of symbols. Then it is possible to compute in time $2^{O(k)}poly(|A|+|B|+|\mathcal{M}|)$ if there exists a solution $B^*$ of $\mathcal{LFCS}$ over instance $(A,B,\mathcal{M})$ such that a longest common subsequence $S$ of $A$ and $B^*$ has length $l$ and it matches by insertion $k$ positions of $A$.*

**Proof.** The correctness of the algorithm follows from Lemma 14 and from the fact that entry $L[|A|,|B|,C,l]=1$ if and only if there exists a solution of $\mathcal{LFCS}$ over instance $(A,B,\mathcal{M})$ having length $l$ that matches by insertion $k$ positions of $A$.

Next, we consider the time complexity of the algorithm. A perfect family of hash functions that color-codes the symbols of $\mathcal{M}$ can be computed in time $2^{O(k)}poly(|\mathcal{M}|)$. Then, the algorithm iterates through $2^{O(k)}poly(|\mathcal{M}|)$ color-codings. For each color-coding, the table $L[i, j, C, l]$ is computed in time $O(2^k|A|^2|B|k)$ (since $l \leq |A|$), since for each of the $2^k|A|^2|B|$ entries we need to look for at most $k$ possible entries. The overall complexity is then $2^{O(k)}poly(|A| + |B| + |\mathcal{M}|)$. □

### 5.2. A polynomial-time algorithm for constant-size alphabet

In this subsection, we give a polynomial-time algorithm for $\mathcal{LFCS}$ when the alphabet $\Sigma = \{\alpha_1, \ldots, \alpha_d\}$, for some constant $d$. We denote this restriction of the $\mathcal{LFCS}$ problem by $\mathcal{LFCS}(d)$. We assume that the number of occurrences in $\mathcal{M}$ of each symbol of $\Sigma$ is bounded by $|A|$, as at most $|A|$ insertion can lead to a match by insertion.

Similarly to the previous subsection, define $L[i, j, l, a_1, \ldots, a_d]$, with $1 \leq i \leq |A|$ and $1 \leq j \leq |B|$, as a function equal to 1 if there exists a common subsequence of length $l$ of $A[0, i]$ and of a filling of $B[0, j]$ with $a_t$, $1 \leq t \leq d$, occurrences of symbol $\alpha_t$ and $\mathcal{M}$ contains at least $a_t$ occurrences of symbol $\alpha_t$. The recurrence to compute function $L$ is defined as follows:

$$L[i, j, l, a_1, \ldots, a_d] = \max \begin{cases} L[i-1, j, l, a_1, \ldots, a_d] & \text{if } i \geq 1 \\ L[i, j-1, l, a_1, \ldots, a_d] & \text{if } j \geq 1 \\ L[i-1, j-1, l-1, a_1, \ldots, a_d] + 1 & \text{if } A[i] = B[j] \\ & \text{and } i, j \geq 1 \\ L[i-1, j, l-1, b_1, \ldots, b_d] & \text{if } A[i] = \alpha_t,\ 1 \leq t \leq d, \\ & b_i = a_i \text{ with } i \neq t, \\ & b_t = a_t - 1 \text{ and} \\ & \mathcal{M} \text{ contains } a_t \\ & \text{occurrences of } \alpha_t \end{cases} \tag{8}$$

In the base case, that is when $i = 0$ and $j = 0$, it holds $L[0, 0, l, a_1, \ldots, a_d] = 1$, if and only if $l = 0$ and $a_i = 0$, with $1 \leq i \leq d$.

Next, we prove the correctness of the recurrence.

**Lemma 16.** *Let $A$, $B$ be two sequences and let $\mathcal{M}$ be a multiset of symbols. Then there exists a common subsequence of $A[1, i]$ and of a filling of $B[1, j]$ having length $l$ where $a_i$, with $1 \leq i \leq d$, occurrences of symbol $\alpha_i$ are inserted in $B$ if and only if $L[i, j, l, a_1, \ldots, a_d] = 1$.*

**Proof.** We prove the lemma by induction on $i + j$. By construction, the lemma holds in the base case, when $i + j = 0$, that is $i = 0$ and $j = 0$. We assume that the lemma holds for $i + j - 1$, and we show that it holds for $i + j$.

Consider a common subsequence having length $l$ of $A[0, i]$ and of a filling $B[0, j]$, $0 \leq i \leq |A|$ and $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$, we prove that $L[i, j, l, a_1, \ldots, a_d] = 1$.

Assume that[1] $A[i]$ is not matched, or that[2] it is matched by alignment with $B[t]$, with $t < j$. In the former case, there exists a length $l$ common subsequence of $A[0, i-1]$ and of a filling of $B[0, j]$, $0 \leq i \leq |A|$ and $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. By induction hypothesis $L[i-1, j, l, a_1, \ldots, a_d] = 1$ and thus by the first case of Recurrence 8, it holds $L[i-1, j, l, a_1, \ldots, a_d] = 1$. In the latter case, there exists a common subsequence having length $l$ of $A[0, i]$ and of a filling of $B[0, t]$, $0 \leq i \leq |A|$ and $0 \leq t < j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. By induction hypothesis[3] $L[i, j-1, l, a_1, \ldots, a_d] = 1$ and by the second case of Recurrence 8 it follows that $L[i, j, l, a_1, \ldots, a_d] = 1$.

Assume that[3] $A[i]$ is matched by alignment with $B[j]$. Then, there is a common subsequence of $A[0, i-1]$ and of a filling of $B[0, j-1]$ with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. By induction hypothesis $L[i-1, j-1, l, a_1, \ldots, a_d] = 1$. Since $A[i] = B[j]$, it follows[4] by the third case of Recurrence 8 that $L[i, j, l, a_1, \ldots, a_d] = 1$.

Now, consider the case that[4] $A[i]$ is matched by insertion with a symbol $\alpha_i \in \mathcal{M}'$. We distinguish two cases. If $A[i]$ is matched by insertion with a symbol inserted in $B$ in position $t < j$, then by induction hypothesis there exists a common subsequence having length $l$ of $A[0, i]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j-1]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. Hence, by induction hypothesis $L[i, j-1, l, a_1, \ldots, a_d] = 1$ and by the second case of Recurrence 8 $L[i, j, l, a_1, \ldots, a_d] = 1$.

If $A[i]$ is matched by insertion with a symbol $\alpha_t$ inserted in $B$ in position $j$, then there exists a common subsequence of $A[0, i-1]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$, that has length $l-1$. Hence, by induction hypothesis $L[i-1, j-1, l-1, b_1, \ldots, b_d] = 1$, where $b_i = a_i$, with $t \neq i$, and $b_t = a_t - 1$. It follows by the fourth case of Recurrence 8 that $L[i, j, l, a_1, \ldots, a_d] = 1$.

Now, we prove that if $L[i, j, l, a_1, \ldots, a_d] = 1$ there exists a common subsequence having length $l$ of $A[0, i]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$.

First, assume that $L[i-1, j, l, a_1, \ldots, a_d] = 1$. By induction hypothesis, there exists a length $l$ common subsequence of $A[0, i-1]$, $1 \leq i \leq |A|$, (hence of $A[0, i]$) and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. A similar proof holds when $L[i, j-1, l, a_1, \ldots, a_d] = 1$.

Assume that $L[i-1, j-1, l-1, a_1, \ldots, a_d] = 1$, with $A[i] = B[j]$. Then, by induction hypothesis, there exists a common subsequence having length $l-1$ of $A[0, i-1]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j-1]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. Then, by defining a match by alignment between position $i$ in $A$ and position $j$ in $B$, there exists a common subsequence having length $l$ of $A[0, i]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$.

Finally, assume that $L[i-1, j, l-1, b_1, \ldots, b_d] = 1$, with $A[i] = \alpha_t$, where $b_p = a_p$, with $p \neq t$, and $a_t = b_t + 1$. By induction hypothesis there exists a common subsequence having length $l-1$ of $A[0, i-1]$, $0 \leq i \leq |A|$, and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $b_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. Then, by defining for position $i$ of $A$ a match by insertion with $\alpha_t$, it follows that there exists a common subsequence having length $l$ of $A[0, i]$, $1 \leq i \leq |A|$, and of a filling of $B[0, j]$, $0 \leq j \leq |B|$, with $a_i$ inserted occurrences of symbol $\alpha_i$, with $1 \leq i \leq d$. □

We can conclude that $\mathcal{LFCS}(d)$ is polynomial-time solvable.

**Theorem 17.** *Let $A$, $B$ be two sequences and $\mathcal{M}$ a multiset of symbols, where $|\Sigma| = d$. Then it is possible to compute in time $O(|A|^{|d+2}|B|)$ an optimal solution of $\mathcal{LFCS}(d)$.*

**Proof.** An optimal solution of $\mathcal{LFCS}(d)$ can be computed with Recurrence 8, by computing the maximum value $l$ such that $L[i, j, l, a_1, \ldots, a_d] = 1$, where each $a_t$, $1 \leq t \leq d$, is at most the number of occurrences of $\alpha_t$ in $\mathcal{M}$. The time complexity to compute each entry $L[i, j, l, a_1, \ldots, a_d]$ is constant (since $d$ is a constant). The number of entries of $L[i, j, l, a_1, \ldots, a_d]$ is $|A|^{d+2}|B|$, since there exist at most $|A|^d$ values $a_1, \ldots, a_d$, as each $a_i$, with $1 \leq i \leq d$, is bounded by the number of occurrences of $\alpha_i$ in $A$, hence by $|A|$, and $l \leq |A|$. □

## 6. Conclusion

We have introduced a variant of the LCS problem, called Longest Filled Common Subsequence ($\mathcal{LFCS}$), to compare a sequence $A$ with an incomplete sequence $B$ to be filled with a multiset $\mathcal{M}$ of symbols. We have shown that the problem is APX-hard (hence NP-hard), even when each symbol occurs at most twice in the input sequence $A$. Then, we have given an approximation algorithm of factor $\frac{3}{5}$ for the problem. Finally, we have given a fixed-parameter algorithm, where the parameter is the number of symbols in $\mathcal{M}$ matched by insertion, and a polynomial-time algorithm when the size of the alphabet $\Sigma$ is bounded by a constant.

There are some interesting open problems related to $\mathcal{LFCS}$. It would be interesting to extend $\mathcal{LFCS}$ to the comparison of two incomplete sequences, similar to what has been done for Scaffold Filling [15]. We conjecture that this extension of $\mathcal{LFCS}$ can be also approximated within constant factor. Consider a longest common subsequence $s_1$ of $A$ and $B$ and a sequence $s_2$ that maximizes the number of positions matched by insertion. We conjecture that the longest of $s_1$, $s_2$ is a $\frac{1}{2}$-approximated solution for the extension of $\mathcal{LFCS}$ to two incomplete sequences and that Lemma 7 holds also in this case. Moreover, we conjecture that $\mathcal{LFCS}$ on two incomplete sequences can be solved in polynomial time for constant-size alphabet, in particular by extending the recurrence of Section 5.2 adding occurrences of symbols that can be inserted into $A$.

Moreover, it would be interesting to design more efficient parameterized algorithms for $\mathcal{LFCS}$, for example by considering the algebraic technique used for the repetition-free longest common subsequence [10].

## Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

## Acknowledgement

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia) by the project GADgET (DSAIPA/DS/0022/2018). Mauro Castelli acknowledges the financial support from the Slovenian Research Agency (research core funding No. P5-0410).

## References

[1] M. Castelli, R. Dondi, G. Mauri, I. Zoppis, The longest filled common subsequence problem, in: J. Kärkkäinen, J. Radoszewski, W. Rytter (Eds.), 28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017, July 4-6, 2017, Warsaw, Poland, in: LIPIcs, vol. 78, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 14:1–14:13.

[2] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol. 147 (1) (1981) 195–197, https://doi.org/10.1016/0022-2836(81)90087-5.

[3] T. Jiang, M. Li, On the approximation of shortest common supersequences and longest common subsequences, SIAM J. Comput. 24 (5) (1995) 1122–1139, https://doi.org/10.1137/S009753979223842X.

[4] A.N. Arslan, Ö. Egecioglu, Algorithms for the constrained longest common subsequence problems, Int. J. Found. Comput. Sci. 16 (6) (2005) 1099–1109.

[5] P. Bonizzoni, G.D. Vedova, R. Dondi, Y. Pirola, Variants of constrained longest common subsequence, Inf. Process. Lett. 110 (20) (2010) 877–881.
[6] F.Y.L. Chin, A.D. Santis, A.L. Ferrara, N.L. Ho, S.K. Kim, A simple algorithm for the constrained sequence problems, Inf. Process. Lett. 90 (4) (2004) 175–179.
[7] Z. Gotthilf, D. Hermelin, M. Lewenstein, Constrained LCS: hardness and approximation, in: Combinatorial Pattern Matching, 19th Annual Symposium, CPM 2008, Pisa, Italy, June 18-20, 2008, Proceedings, 2008, pp. 255–262.
[8] Y. Tsai, The constrained longest common subsequence problem, Inf. Process. Lett. 88 (4) (2003) 173–176, https://doi.org/10.1016/j.ipl.2003.07.001.
[9] S.S. Adi, M.D.V. Braga, C.G. Fernandes, C.E. Ferreira, F.V. Martinez, M. Sagot, M.A. Stefanes, C. Tjandraatmadja, Y. Wakabayashi, Repetition-free longest common subsequence, Discrete Appl. Math. 158 (12) (2010) 1315–1324.
[10] G. Blin, P. Bonizzoni, R. Dondi, F. Sikora, On the parameterized complexity of the repetition free longest common subsequence problem, Inf. Process. Lett. 112 (7) (2012) 272–276.
[11] P. Bonizzoni, G.D. Vedova, R. Dondi, G. Fertin, R. Rizzi, S. Vialette, Exemplar longest common subsequence, IEEE/ACM Trans. Comput. Biol. Bioinform. 4 (4) (2007) 535–543.
[12] C.E. Ferreira, C. Tjandraatmadja, A branch-and-cut approach to the repetition-free longest common subsequence problem, Electron. Notes Discrete Math. 36 (2010) 527–534.
[13] P. Chain, et al., Genomics. Genome project standards in a new era of sequencing, Science 326 (2009) 236–237.
[14] A. Muñoz, C. Zheng, Q. Zhu, V.A. Albert, S. Rounsley, D. Sankoff, Scaffold filling, contig fusion and comparative gene order inference, BMC Bioinform. 11 (2010) 304, https://doi.org/10.1186/1471-2105-11-304.
[15] N. Liu, H. Jiang, D. Zhu, B. Zhu, An improved approximation algorithm for scaffold filling to maximize the common adjacencies, IEEE/ACM Trans. Comput. Biol. Bioinform. 10 (4) (2013) 905–913, https://doi.org/10.1109/TCBB.2013.100.
[16] L. Bulteau, A.P. Carrieri, R. Dondi, Fixed-parameter algorithms for scaffold filling, Theor. Comput. Sci. 568 (2015) 72–83.
[17] B. Zhu, Genomic scaffold filling: a progress report, in: D. Zhu, S. Bereg (Eds.), Frontiers in Algorithmics, 10th International Workshop, FAW 2016, Qingdao, China, June 30-July 2, 2016, Proceedings, in: Lecture Notes in Computer Science, vol. 9711, Springer, 2016, pp. 8–16.
[18] L. Bulteau, G. Fertin, C. Komusiewicz, Beyond adjacency maximization: scaffold filling for new string distances, in: J. Kärkkäinen, J. Radoszewski, W. Rytter (Eds.), 28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017, July 4-6, 2017, Warsaw, Poland, in: LIPIcs, vol. 78, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 27:1–27:17.
[19] P. Alimonti, V. Kann, Some APX-completeness results for cubic graphs, Theor. Comput. Sci. 237 (1–2) (2000) 123–134, https://doi.org/10.1016/S0304-3975(98)00158-3.
[20] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer-Verlag, Heidelberg, 1999.
[21] N. Alon, R. Yuster, U. Zwick, Color-coding, J. ACM 42 (4) (1995) 844–856.