On the Parameterized Complexity of the Repetition Free Longest Common Subsequence Problem

Guillaume Blin, Paola Bonizzoni, Riccardo Dondi, Florian Sikora

On the Parameterized Complexity of the Repetition Free Longest Common Subsequence Problem

Guillaume Blin[a], Paola Bonizzoni[b], Riccardo Dondi[c], Florian Sikora[a,d]

[a] *Université Paris-Est, LIGM UMR CNRS 8049, France*
[b] *DISCo, Università degli Studi di Milano-Bicocca, Milano - Italy*
[c] *Dipartimento di Scienze dei Linguaggi, della Comunicazione e degli Studi Culturali, Università degli Studi di Bergamo, Via Donizetti 3, 24129 Bergamo, Italy*
[d] *Lehrstuhl für Bioinformatik, Friedrich-Schiller-Universität Jena, Germany*

Abstract

Longest common subsequence is a widely used measure to compare strings, in particular in Computational Biology. Recently, several variants of the longest common subsequence have been introduced to tackle the comparison of genomes. In particular, the Repetition Free Longest Common Subsequence problem (RFLCS) is a variant of the LCS problem that asks for a longest common subsequence of two input strings with no repetition of symbols. In this paper, we investigate the parameterized complexity of RFLCS. First, we show that the problem does not admit a polynomial kernel. Then, we present a randomized FPT algorithm for the RFLCS problem, improving the time complexity of the existent FPT algorithm.

Keywords: Repetition Free Longest Common Subsequence, Longest Common Subsequence, Parameterized Complexity.

1. Introduction

Longest Common Subsequence (LCS) has been widely used as a measure to compare strings in different fields [2], in particular for the comparison of two (or more) genomes in Bioinformatics. Genomes are usually viewed as strings, where each symbol represents a gene, and the comparison of the strings associated with the genomes provides a measure of their similarities and differences. As the order in which genes appear in the genomes is considered relevant in the comparison, LCS provides a natural measure to compare genomes.

Different variants of longest common subsequence have been proposed [5, 1, 6] to compare biological strings, where, given two strings s_1 and s_2, the computed longest common subsequence is required to satisfy some constraint. In

Email addresses: gblin@univ-mlv.fr (Guillaume Blin), bonizzoni@disco.unimib.it (Paola Bonizzoni), riccardo.dondi@unibg.it (Riccardo Dondi), sikora@univ-mlv.fr (Florian Sikora)

particular, the REPETITION FREE LONGEST COMMON SUBSEQUENCE (RFLCS) problem, proposed in [1], requires that each symbol appears at most once in the common subsequence of the input strings. The use of such constraint is motivated by the exemplar hypothesis [17], that aims to identify the original copy of a gene that has originated all the copies of that gene in the genome through duplications. As a consequence, in the RFLCS problem, the input consists of two strings s_1 and s_2, and the output consists of a longest common subsequence of s_1 and s_2 containing no repetition of symbols. The RFLCS is known to be APX-hard, even when each symbol occurs at most twice in each of the two input strings [1]. Furthermore, the problem admits a k-approximation algorithm, where k is the maximum number of occurrences of each symbol in one of the two strings [1]. Concerning parameterized complexity, if parameterized by the size k of the solution, the problem admits a deterministic FPT algorithm [6], of time complexity $O^*(c^k)$, where c is a large constant, and space complexity $O^*(2^k)$[1]. The algorithm can be easily randomized, thus giving a randomized FPT algorithm of time complexity $O^*((2e)^k))$ and space complexity $O^*(2^k)$.

In this paper, we deepen the investigation on the parameterized complexity of the RFLCS problem. For details on parameterized complexity, we refer the reader to [16]. First, we investigate the kernelization complexity of RFLCS. Kernelization is a widely used technique in parameterized complexity [16], that aims to preprocess in polynomial time an instance of a problem, in order to produce an instance having size depending only on the considered parameter. Recently, several results [3, 4] on the kernelization complexity have been introduced, in order to prove that a problem, although in FPT, does not admit a polynomial size kernel. Applying a technique of [3], we show that the RFLCS problem does not admit a polynomial size kernel, unless $NP \subseteq coNP/Poly$. Notice that $NP \subseteq coNP/Poly$ would imply a collapse to the third level of the polynomial time hierarchy.

On the positive side, we present a randomized fixed-parameter algorithm for the RFLCS problem parameterized by the size of the solution (denoted as k). Our randomized fixed-parameter algorithm has time complexity $O^*(2^k)$ and polynomial space complexity, thus improving upon the existing algorithm proposed in [6].

The rest of the paper is organized as follows. In Section 2, we give some preliminary definitions. In Section 3, we investigate the kernelization complexity of RFLCS, while in Section 4, we present the fixed-parameter algorithm for RFLCS.

2. Preliminaries

In this section we introduce some basic definitions. Let Δ denote a finite alphabet and Δ^* the set of all finite length strings over Δ. Let $\Pi \subseteq \Delta^* \times \mathbb{N}$

[1] We recall that in the $O^*()$ notation, the polynomially bounded terms are suppressed.

be a parameterized problem and let $1 \notin \Delta$. The derived classical problem Π^C associated with Π is $\{x1^k : (x, k) \in \Pi\}$.

Let s be a string over alphabet Σ. We denote by $|s|$ the length of s. The i-th symbol of s is denoted by $s[i]$. 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.

Consider two strings s_1 and s_2. A common subsequence of s_1 and s_2 is a string s that can be computed by deleting some symbols (possibly none) in s_1 or s_2. A longest common subsequence s of s_1, s_2 is a common subsequence of s_1 and s_2 having maximum length. Given two strings s_1 and s_2, we define $s_1 \odot s_2$ as the concatenation of s_1 and s_2.

The RFLCS problem is a constraint version of the longest common subsequence problem that has been introduced in [1]. Given two strings s_1, s_2 over alphabet Σ, RFLCS asks for a longest common subsequence of s_1, s_2 where each symbol of Σ occurs at most once. Formally, the problem is defined as follows:

Problem 1. RFLCS
Input: *A pair of strings $I = (s_1, s_2)$ over alphabet Σ.*
Parameter: *k.*
Output: *A common subsequence s of s_1 and s_2, such that each symbol $\sigma \in \Sigma$ occurs at most once in s and $|s| \geq k$.*

The derived classic problem RFLCS^C is known to be NP-hard [1], even when each symbol occurs at most twice in each string.

3. Kernelization Complexity

In order to prove a lower bound on the polynomial kernel of RFLCS, we need to introduce in Section 3.1 some preliminary notions.

3.1. Preliminaries on Kernelization Complexity

In this section we introduce some preliminary notions about Kernelization Complexity, and in particular the notion of composition algorithm [3].

Definition 1. *[3] A composition algorithm for a parameterized problem $\Pi \subseteq \Delta \times \mathbb{N}$ is an algorithm that, given in input a sequence $\langle (x_1, k), (x_2, k), \ldots, (x_t, k) \rangle$ where each $(x_i, k) \in \Delta \times \mathbb{N}$, runs in time polynomial in $\sum_{i=1}^{t} x_i + k$, and outputs an instance $(y, k') \in \Delta \times \mathbb{N}$ such that*

1. *$(y, k') \in \Pi$ iff $(x_i, k) \in \Pi$, for some $1 \leq i \leq t$;*
2. *k' is polynomial in k.*

A parameterized problem is compositional if it has a composition algorithm.

We will apply the following fundamental result on kernelization complexity [9].

Theorem 2. *[9] Let Π be a compositional parameterized problem whose derived classical problem Π^C is NP-complete. If Π has a polynomial kernel, then $NP \subseteq coNP/Poly$.*

3.2. Kernelization Complexity of RFLCS

In this section we prove that the RFLCS does not admit a polynomial kernel, unless $NP \subseteq coNP/Poly$. Since the derived classical problem RFLCS^C is NP-complete [1], we can prove the result applying the concept of composition algorithm given in Section 3.1.

Consider two instances $I_1 = \Big((s_{1,a}, s_{1,b}), k\Big)$, $I_2 = \Big((s_{2,a}, s_{2,b}), k\Big)$ of the RFLCS problem, such that $s_{1,a}, s_{1,b}$ ($s_{2,a}, s_{2,b}$ respectively) are over alphabet Σ_1 (Σ_2 respectively). We assume that $\Sigma_1 \cap \Sigma_2 = \emptyset$, otherwise, starting from $s_{1,a}, s_{1,b}$, we can compute in time $O(|s_{1,a}| + |s_{1,b}|)$ an instance $I'_1 = \Big((s'_{1,a}, s'_{1,b}), k\Big)$ of RFLCS such that (1) $s'_{1,a}, s'_{1,b}$ are over alphabet Σ'_1, with $\Sigma'_1 \cap \Sigma_2 = \emptyset$; (2) RFLCS on input $\Big((s'_{1,a}, s'_{1,b}), k\Big)$ admits a feasible solution if and only if RFLCS on input $\Big((s_{1,a}, s_{1,b}), k\Big)$ admits a feasible solution.

Indeed assume that $\Sigma_1 \cap \Sigma_2 \neq \emptyset$. Define a new alphabet Σ'_1 such that for each $\alpha \in \Sigma_1$, there is a symbol $\alpha' \in \Sigma'_1$, where $\alpha' \notin \Sigma_1 \cup \Sigma_2$. Then, define the string $s'_{1,x}$, with $x \in \{a, b\}$, as follows: if $s_{1,x}[i] = \alpha$, with $1 \leq i \leq |s_{1,x}|$, then $s'_{1,x}[i] = \alpha'$. By construction, $s'_{1,a}, s'_{1,b}$ are over alphabet Σ'_1, with $\Sigma'_1 \cap \Sigma_2 = \emptyset$. Furthermore, it is easy to see that RFLCS on input $\Big((s'_{1,a}, s'_{1,b}), k\Big)$ admits a feasible solution if and only if RFLCS on input $\Big((s_{1,a}, s_{1,b}), k\Big)$ admits a feasible solution.

Given $(s_{1,a}, s_{1,b})$ and $(s_{2,a}, s_{2,b})$, the composition algorithm defines the operation $(s_{1,a}, s_{1,b}) \otimes (s_{2,a}, s_{2,b})$, which starting from $(s_{1,a}, s_{1,b})$ and $(s_{2,a}, s_{2,b})$ outputs the strings $s^a_{1,2}$, $s^b_{1,2}$, defined as follows:

- $s^a_{1,2} = s_{1,a} \odot s_{2,a}$, that is $s^a_{1,2}$ is the concatenation of $s_{1,a}$ and $s_{2,a}$;
- $s^b_{1,2} = s_{2,b} \odot s_{1,b}$, that is $s^b_{1,2}$ is the concatenation of $s_{2,b}$ and $s_{1,b}$.

Example 1. *Consider the instances $I_1 = \Big((s_{1,a}, s_{1,b}), k\Big)$ and $I_2 = \Big((s_{2,a}, s_{2,b}), k\Big)$ of* RFLCS*, where*

- $s_{1,a} = abc$
- $s_{1,b} = bca$
- $s_{2,a} = def$
- $s_{2,b} = ddf$

Then $(s_{1,a}, s_{1,b}) \otimes (s_{2,a}, s_{2,b})$ *produces the following strings:*

$$s_{1,2}^a = s_{1,a} \odot s_{2,a} = abcdef$$

$$s_{1,2}^b = s_{2,b} \odot s_{1,b} = ddfbca.$$

Theorem 3. *The* RFLCS *problem does not admit a polynomial kernel unless* $NP \subseteq coNP/Poly$.

Proof. Let $\Big((s_{1,a}, s_{1,b}), k\Big)$, $\Big((s_{2,a}, s_{2,b}), k\Big), \ldots, \Big((s_{t,a}, s_{t,b}), k\Big)$ be t instances of RFLCS, defined over pairwise disjoint alphabets $\Sigma_1, \ldots, \Sigma_t$. Let $(s_{y,a}, s_{y,b})$ be a pair of strings defined as follows: $(s_{y,a}, s_{y,b}) = (s_{1,a}, s_{1,b}) \otimes (s_{2,a}, s_{2,b}) \otimes \cdots \otimes (s_{t,a}, s_{t,b})$, that is $s_{y,a} = s_{1,a} \odot s_{2,a} \odot \cdots \odot s_{t,a}$, while $s_{y,b} = s_{t,b} \odot s_{t-1,b} \odot \cdots \odot s_{1,b}$. We claim that there is a solution for RFLCS over instance $\Big((s_{y,a}, s_{y,b}), k\Big)$ if and only if there exists a $j \in [t]$ such that RFLCS admits a solution over instance $\Big((s_{j,a}, s_{j,b}), k)\Big)$.

First, assume that there is a solution of RFLCS over instance $\Big((s_{j,a}, s_{j,b}), k\Big)$, for some $j \in [t]$. Consider the substrings $s_{y,a}^j$, $s_{y,b}^j$ of $s_{y,a}$, $s_{y,b}$ respectively, consisting only of symbols in Σ_j. Since by construction the instances $\Big((s_{1,a}, s_{1,b}), k\Big)$, $\Big((s_{2,a}, s_{2,b}), k\Big), \ldots, \Big((s_{t,a}, s_{t,b}), k\Big)$ are over pairwise disjoint alphabets $\Sigma_1, \ldots, \Sigma_t$, it follows by construction that $s_{y,a}^j$, $s_{y,b}^j$ are identical to $s_{j,a}$, $s_{j,b}$ respectively. Since there is a repetition free common subsequence of $s_{j,a}$, $s_{j,b}$ of length at least k, then there is a repetition free common subsequence of $s_{y,a}^j$, $s_{y,b}^j$ of length at least k, which implies that there is a solution for RFLCS over instance $\Big((s_{y,a}, s_{y,b}), k\Big)$.

Assume now that there is a solution s of RFLCS over instance $\Big((s_{y,a}, s_{y,b}), k\Big)$. Then we claim that s consists of symbols from exactly one alphabet Σ_j, for some $j \in [t]$. Assume to the contrary that such a solution s contains a symbol $\alpha_i \in \Sigma_i$ and a symbol $\alpha_j \in \Sigma_j$ with $i, j \in [t]$ and w.l.o.g. $i < j$. Recall that the instances $\Big((s_{1,a}, s_{1,b}), k\Big)$, $\Big((s_{2,a}, s_{2,b}), k\Big), \ldots, \Big((s_{t,a}, s_{t,b}), k\Big)$ are defined over pairwise disjoint alphabets $\Sigma_1, \ldots, \Sigma_t$. By construction α_i appears on the left of α_j is $s_{y,a}$, while α_i appears on the rigth of α_j is $s_{y,b}$. Hence s cannot contain both a symbol $\alpha_i \in \Sigma_i$ and a symbol $\alpha_j \in \Sigma_j$. It follows that s is a string over alphabet Σ_j, for some $j \in [t]$, hence s is a repetition free common subsequence of $s_{y,a}^j$, $s_{y,b}^j$, which implies that s is a repetition free common subsequence of $s_{j,a}$, $s_{j,b}$.

By Theorem 2, it follows that RFLCS does not admit a polynomial kernel unless $NP \subseteq coNP/Poly$. $\square$

4. A Fixed-Parameter Algorithm

In order to present the fixed-parameter algorithm for RFLCS, we need to introduce in Section 4.1 some definitions and results concerning the multilinear detection technique.

4.1. Arithmetic circuits

Intuitively, the aim of this technique is to efficiently detect a multilinear monomial of a given degree in an arithmetic circuit, which is a compressed encoding of a multivariate polynomial.

More formally, let X be a set of variables $\{x_1, x_2, \ldots, x_z\}$. A multivariate polynomial is a sum of monomials. The degree of a monomial is the sum of the monomial variables degrees. A monomial is *multilinear* if the degree of all the variables is equal to 1. Thus, a multilinear monomial of degree k has exactly k different variables. For example, the degree of the monomial $x_1 \cdot x_2^2$ is 3, and so is the degree of the multilinear monomial $x_1 \cdot x_2 \cdot x_3$.

An *arithmetic circuit* over X is a pair $\mathcal{C} = (C, r)$, where C is a labeled directed acyclic graph (DAG) such that each leaf (with an out-degree equal to zero) is labeled by a variable of X, each internal node is labeled either by $+$ or $\times$, and r is a distinguished node of C (the *root* of C).

We can encode a polynomial with an arithmetic circuit. Recursively, the polynomial corresponding to a leaf is the label of the leaf, and the polynomial of an internal node labeled by $+$ (resp. $\times$) is the sum (resp. the product) of its childrens' polynomials.

Now, we can introduce the Multilinear Detection problem (MLD). Informally, given an arithmetic circuit $\mathcal{C}$ and an integer k, the Multilinear Detection problem asks if the polynomial $P_{\mathcal{C}}$ encoded by $\mathcal{C}$ has a multilinear monomial of degree k. More formally, we give the definition of the Multilinear Detection problem.

Problem 2. MLD
Input: *An arithmetic circuit $\mathcal{C}$ encoding a polynomial $P_{\mathcal{C}}$ over a set of variables X.*
Parameter: *k.*
Output: *Does $P_{\mathcal{C}}$ contains a multilinear monomial of degree k?*

In [14, 18], it is shown the following fundamental result for the MLD problem.

Theorem 4 ([14, 18])**.** *There exists a randomized algorithm that solves* MLD *in $O(2^k|\mathcal{C}|)$ time and in $O(|\mathcal{C}|)$ space.*

In Section 4.2, we apply this result in order to obtain a new fixed-parameter algorithm for RFLCS.

4.2. A Fixed-Parameter Algorithm for RFLCS

In this section we present a randomized fixed-parameter algorithm for RFLCS of time complexity $O^*(2^k)$ and of polynomial space complexity. The algorithm is based on the detection of multilinear monomials technique, presented in Section 4.1. Let s_1, s_2 be the two input strings of RFLCS over alphabet Σ, we construct a circuit $\mathcal{C}$ as follows. $\mathcal{C}$ is defined over the set of variables

$\{x_a : a \in \Sigma\}$. Moreover, the circuit has a root P and a set of intermediary nodes $P_{i,j,l}$, for $0 \leq i \leq |s_1|$, $0 \leq j \leq |s_2|$ and $0 \leq l \leq |\Sigma|$. Informally, the multilinear monomial $P_{i,j,l}$ encodes a repetition free common subsequence of the strings $s_1[1,\ldots,i]$, $s_j[1,\ldots,j]$, having length l. $P_{i,j,l}$ is defined as follows:

$$P_{i,j,l} = \begin{cases} P_{i-1,j,l} + P_{i,j-1,l} + P_{i-1,j-1,l} & \text{if } i > 0,\ j > 0,\ s_1[i] \neq s_2[j] \text{ and } l \geq 1, \\ P_{i-1,j,l} + P_{i,j-1,l} + P_{i-1,j-1,l-1} \cdot x_a & \text{if } i > 0,\ j > 0,\ s_1[i] = s_2[j] = a \text{ and } l \geq 1, \\ 1 & \text{if } l = 0 \text{ and } i, j \geq 0, \\ 0 & \text{if } i = 0 \text{ or } j = 0, \text{ and } l > 0. \end{cases} \tag{1}$$

Finally, define $P = P_{|s_1|,|s_2|,k}$. The resulting instance of MLD is $I = (\mathcal{C}, k)$. Next, we prove the correctness of the reduction.

Theorem 5. RFLCS *is solvable by a randomized algorithm in* $O(2^k k|s_1||s_2|)$ *time and* $O(k|s_1||s_2|)$ *space.*

Proof. We will prove that there is a RFLCS for the strings s_1, s_2 of length k if and only if there is a multilinear monomial in $\mathcal{C}$ of length k. To do so, we prove by induction on $i+j$ that there exists a repetition free common subsequence of $s_1[1\ldots i]$, $s_2[1\ldots j]$ containing the set of symbols $\{a_1,\ldots a_l\}$ (hence of length l) if and only if there is a multilinear monomial $x_{a_1}\ldots x_{a_l}$ in $P_{i,j,l}$.

When $i = j = 1$, assume that there is a repetition free common subsequence consisting w.l.o.g. of a_1. Then $s_1[1] = s_2[1] = a_1$, and, by Equation 1, $P_{1,1,1} = P_{0,0,0} \cdot x_{a_1} = x_{a_1}$. Similarly, if $P_{1,1,1} = P_{0,0,0} \cdot x_{a_1} = x_{a_1}$, then by construction it must hold $s_1[1] = s_2[1] = a_1$.

Now, assume that the lemma holds for $i + j = h$, we prove that the lemma also holds for $i + j = h + 1$. Assume that there is a repetition free common subsequence s of $s_1[1\ldots i]$, $s_2[1\ldots j]$ consisting of the set of symbols $\{a_1,\ldots a_l\}$. It follows, that either $s[l] = s_1[i] = s_2[j] = a_z$ for a given $1 \leq z \leq l$, or $s[l] \neq s_1[i]$, or $s[l] \neq s_2[j]$. In the first case, by Equation 1, $P_{i,j,l} = P_{i-1,j-1,l-1} \cdot x_{a_z}$. By induction, if there is a repetition free common subsequence s' of $s_1[1\ldots i-1]$, $s_2[1\ldots j-1]$ consisting of symbols $\{a_1,\ldots,a_l\} \setminus \{a_z\}$, then $P_{i-1,j-1,l-1}$ contains a multilinear monomial of length $l-1$ not including x_{a_z}. If $s[l] \neq s_1[i]$ or $s[l] \neq s_2[j]$, then s is a repetition free common subsequence of $s_1[1\ldots i], s_2[1\ldots j-1]$, or of $s_1[1\ldots i-1], s_2[1\ldots j]$, or of $s_1[1\ldots i-1], s_2[1\ldots j-1]$. By induction hypothesis, one of $P_{i,j-1,l}$, $P_{i-1,j,l}$, $P_{i-1,j-1,l}$ contains a multilinear monomial $x_{a_1}\ldots x_{a_l}$. By Equation 1, $P_{i,j,l}$ also contains such a multilinear monomial.

Assume that $P_{i,j,l}$ contains a multilinear monomial $m_x = x_{a_1}\ldots x_{a_l}$. By construction (Equation 1) it holds that m_x is either contained in one of $P_{i-1,j,l}$, $P_{i,j-1,l}$, $P_{i-1,j-1,l}$ or it is obtained from $P_{i-1,j-1,l-1} \cdot x_a$ and $P_{i-1,j-1,l-1}$ contains a multilinear monomial over the set of variables $\{x_{a_1}\ldots x_{a_l}\} \setminus \{x_a\}$. In the first three cases, by induction hypothesis there is a repetition free longest common subsequence of $s_1[1\ldots i], s_2[1\ldots j]$ consisting of symbols $a_1,\ldots,a_l$. Hence, assume that the latter case holds, that is $P_{i,j,l} = P_{i-1,j-1,l-1} \cdot x_a$. Hence $P_{i-1,j-1,l-1}$ contains a monomial over the set of variables $\{x_{a_1}\ldots x_{a_l}\} \setminus \{x_a\}$.

As a consequence, by induction hypothesis, there is a repetition free longest common subsequence s of $s_1[1 \dots i-1], s_2[1 \dots j-1]$ over the set of symbols $\{a_1, \dots, a_l\} \setminus \{a\}$. But then $s' = s \odot a$ is a repetition free longest common subsequence of $s_1[1 \dots i], s_2[1 \dots j]$ over the set of symbols $\{a_1, \dots, a_l\}$, concluding the proof for the induction.

By applying Theorem 4 of Section 4.1, and by observing that $|\mathcal{C}| = k|s_1||s_2|$, we can solve RFLCS with a randomized algorithm with the claimed complexities.

□

5. Conclusion

We have investigated the parameterized complexity of the RFLCS, a variant of the LCS problem that, given two strings s_1, s_2, asks for a common subsequence s of s_1, s_2 of length at least k such that s contains at most one occurrence of each symbol. We have proved that the RFLCS does not admit a polynomial size kernel, unless $NP \subseteq coNP/Poly$ and we have given a fixed-parameter algorithm for RFLCS of resp. $O(2^k k|s_1||s_2|)$ time and $O(k|s_1||s_2|)$ space complexities. To obtain these complexities, we used the multilinear detection framework, which seems to be quite powerful to improve algorithms based on dynamic programming and color-coding, widely used in computational biology. Indeed, some authors already gave improvements of some algorithms using this framework [11] [10].

An interesting open problem lies on the analysis of the approximation complexity of RFLCS: it is still open whether RFLCS admits a constant factor approximation algorithm or not.

References

[1] S. Adi, M. Braga, C. Fernandes, C. Ferreira, F. Martinez, M. Sagot, M. Stefanes, C. Tjandraatmadja, and Y. Wakabayashi. Repetition-free Longest Common Subsequence. In *Discrete Applied Mathematics*, 158(12): 1315–1324, 2010.

[2] L. Bergroth, H. Hakonen, T. Raita. A Survey of Longest Common Subsequence Algorithms. In *SPIRE*, pages 39–48, 2000.

[3] H. L. Bodlaender, R. G. Downey, M. R. Fellows, D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423-434, 2009.

[4] H. L. Bodlaender, S.Thomassé, A. Yeo. Kernel Bounds for Disjoint Cycles and Disjoint Paths. In *ESA*, volume 5757 of LNCS, pages 635-646. Springer, 2009.

[5] P. Bonizzoni, G. Della Vedova, R. Dondi, G. Fertin, R. Rizzi, and S. Vialette. Exemplar Longest Common Subsequence. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 4(4):535–543, 2007.

[6] P. Bonizzoni, G. Della Vedova, R. Dondi, Y. Pirola. Variants of Constrained Longest Common Subsequence. *Inf. Process. Lett.*, 110(20): 877–881, 2010.

[7] F. Y. L. Chin, A. D. Santis, A. L. Ferrara, N. L. Ho, and S. K. Kim. A Simple Algorithm for the Constrained Sequence Problems. *Inf. Process. Lett.*, 90(4):175–179, 2004.

[8] R. Downey, and M. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[9] L. Fortnow, R. Santhanam, Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.* 77(1): 91–106, 2011

[10] B. Fu, L. Zhang, A Polynomial Algebra Method for Computing Exemplar Breakpoint Distance. In *ISBRA 2011*, volume 6674 or LNCS, pages 297–305, 2011.

[11] S. Guillemot, F. Sikora, Finding and Counting Vertex-Colored Subtrees. In *MFCS 2010*, volume 6281 or LNCS, pages 405–416, 2010.

[12] T. Jiang and M. Li, On the Approximation of Shortest Common Supersequences and Longest Common Subsequences. *SIAM J. on Computing*, 24(5):1122-1139, 1995.

[13] I. Koutis, Faster Algebraic Algorithms for Path and Packing Problems. In *ICALP 2008*, volume 5125 of LNCS, pages 575-586, 2008.

[14] I. Koutis and R. Williams, Limits and Applications of Group Algebras for Parameterized Problems. In *ICALP 2009*, volume 5555 of LNCS, pages 653-664, 2009.

[15] D. Maier, The Complexity of Some Problems on Subsequences and Supersequences. *J. ACM*, 25:322–336, 1978.

[16] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.

[17] D. Sankoff, Genome Rearrangement with Gene Families. *Bioinformatics*, 11: 909–917, 1999

[18] R. Williams, Finding paths of length k in $O^*(2^k)$ time. *IPL*, 109(6):315-318, 2009