The Longest Run Subsequence Problem: Further Complexity Results

Riccardo Dondi
Università degli Studi di Bergamo, Bergamo, Italy

Florian Sikora
Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016 Paris, France

Abstract

Longest Run Subsequence is a problem introduced recently in the context of the scaffolding phase of genome assembly (Schrinner et al., WABI 2020). The problem asks for a maximum length subsequence of a given string that contains at most one run for each symbol (a run is a maximum substring of consecutive identical symbols). The problem has been shown to be NP-hard and to be fixed-parameter tractable when the parameter is the size of the alphabet on which the input string is defined. In this paper we further investigate the complexity of the problem and we show that it is fixed-parameter tractable when it is parameterized by the number of runs in a solution, a smaller parameter. Moreover, we investigate the kernelization complexity of Longest Run Subsequence and we prove that it does not admit a polynomial kernel when parameterized by the size of the alphabet or by the number of runs. Finally, we consider the restriction of Longest Run Subsequence when each symbol has at most two occurrences in the input string and we show that it is APX-hard.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Approximation algorithms analysis; Theory of computation → Graph algorithms analysis

Keywords and phrases Parameterized complexity, Kernelization, Approximation Hardness, Longest Subsequence

Digital Object Identifier 10.4230/LIPIcs.CPM.2021.14

Funding *Florian Sikora*: Partially funded by ESIGMA (ANR-17-CE23-0010).

1 Introduction

A fundamental problem in computational genomics is genome assembly, whose goal is reconstructing a genome given a set of reads (a read is a sequence of base pairs) [2, 5]. After the generation of initial assemblies, called *contigs*, they have to be ordered correctly, in a phase called *scaffolding*. One of the commonly used approaches for scaffolding is to consider two (or more) incomplete assemblies of related samples, thus allowing the alignment of contigs based on their similarities [7]. However, the presence of genomic repeats and structural differences may lead to misleading connections between contigs.

Consider two sets X, Y of contigs, such that the order of contigs in Y has to be inferred using the contigs in X. Each contig in X is divided into equal size bins and each bin is mapped to a contig in Y (based on best matches). As a consequence, each bin in X can be partitioned based on the mapping to contigs of Y. However this mapping of bins to contigs, due to errors (in the sequencing or in the mapping process) or mutations, may present some inconsistencies, in particular bins can be mapped to scattered contigs, thus leading to an inconsistent partition of X, as shown in Fig. 1. In order to infer the most likely partition of X (and then distinguish between the transition from one contig to the other and errors in the mapping), the method proposed in [10] asks for a longest subsequence of the contig matches in X such that each contig run occurs at most once (see Fig. 1 for an example).

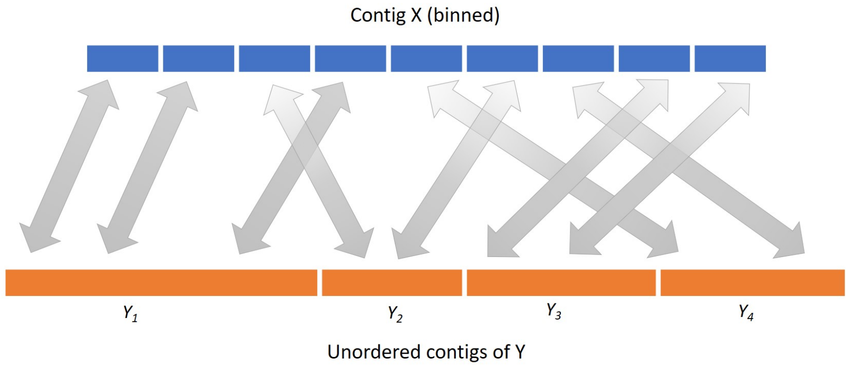


Figure 1 An example of matching a binned contig (X) with the unordered contigs of Y. The string inferred from this matching is $S = y_1\ y_1\ y_2\ y_1\ y_4\ y_2\ y_4\ y_3\ y_3$. Notice that S induces an inconsistent partition of the bins of X, for example for the mapping of Y_1 and Y_2. Indeed, Y_1 is mapped in the first, second and fourth bin of X, while Y_2 is mapped in the third and sixth bin of X. A longest run subsequence R of S is $R = y_1\ y_1\ y_1\ y_4\ y_4\ y_3\ y_3$, that induces a partition of some bins of X.

This problem, called Longest Run Subsequence, has been recently introduced and studied by Schrinner et al. [10]. Longest Run Subsequence has been shown to be NP-hard [10] and fixed-parameter tractable when the parameter is the size of the alphabet on which the input string is defined [10]. Furthermore, an integer linear program has been given for the problem [10]. Schrinner et al. let as future work approximability and parameterized complexity results on the problem [10]. Note that this problem could be seen as close to the "run-length encoded" string problems in the string literature, where a string is described as a sequence of symbols followed by the number of its consecutive occurrences, i.e. a sequence of runs where only its symbol and its length is stored (see for example [3]). While finding the longest common subsequence between two of such strings is a polynomial task, our problem is, to the best of our knowledge, not studied in literature before the work of Schrinner et al. [10].

In this paper we further investigate the complexity of the Longest Run Subsequence problem. We start in Section 2 by introducing some definitions and by giving the formal definition of the problem. Then in Section 3 we give a randomized fixed-parameter algorithm, where the parameter is the number of runs in a solution, based on the multilinear detection technique. In Section 4, we investigate the kernelization complexity of Longest Run Subsequence and we prove that it does not admit a polynomial kernel when parameterized by the size of the alphabet or by the number of runs. Notice that the problem admits a polynomial kernel when parameterized by the length of the solution (see Observation 6). Finally, in Section 5 we consider the restriction of Longest Run Subsequence when each symbol has at most two occurrences in the input string and we show that it is APX-hard. We conclude the paper with some open problems.

2 Definitions

In this section we introduce the main definitions we need in the rest of the paper.

Problem Definition. Given a string S, $|S|$ denotes the length of the string; $S[i]$, with $1 \leqslant i \leqslant |S|$, denotes the symbol of S in position i, $S[i, j]$, with $1 \leqslant i \leqslant j \leqslant |S|$, denotes the substring of S that starts in position i and ends in position j. Notice that if $i = j$, then $S[i, i]$ is the symbol $S[i]$. Given a symbol a, we denote by a^p, for some integer $p \geqslant 1$, a string consisting of the concatenation of p occurrences of symbol a.

A *run* in S is a substring $S[i, j]$, with $1 \leqslant i \leqslant j \leqslant |S|$, such that $S[z] = a$, for each $i \leqslant z \leqslant j$, with $a \in \Sigma$. Given $a \in \Sigma$, an a-run is a run in S consisting of repetitions of symbol a. Given a string S on alphabet Σ, a *run subsequence* S' of S is a subsequence that contains *at most one run* for each symbol $a \in \Sigma$.

Now, we are ready to define the Longest Run Subsequence problem.

> Longest Run Subsequence
> - **Input**: A string S on alphabet Σ, an integer k.
> - **Output**: Does there exist a run subsequence R of length k?

A string $S[i, j]$ contains an a-run with $a \in \Sigma$, if it contains a run subsequence which is an a-run. A run subsequence of S which is an a-run, with $a \in \Sigma$, is maximal if it contains all the occurrences of symbol a in S. Note that an optimal solution may not take maximal runs. For example, consider

$$S = abacaabbab$$

an optimal run subsequence in S is

$$R = aaaabbb$$

Note that no run in R is maximal and even that some symbol of Σ may not be in an optimal solution of Longest Run Subsequence, in the example no c-run belongs to R.

Graph Definitions. Given a graph $G = (V, E)$, we denote by $N(v) = \{u : \{u, v\} \in E\}$, the neighbourhood of v. The closed neighbourhood of v is $N[v] = N(v) \cup \{v\}$. $V' \subset V$ is an independent set when $\{u, v\} \notin E$ for each $u, v \in V'$. We recall that a graph $G = (V, E)$ is *cubic* when $|N(v)| = 3$ for each $v \in V$.

Parameterized Complexity. A *parameterized problem* is a decision problem specified together with a *parameter*, that is, an integer k depending on the instance. A problem is *fixed-parameter tractable* (FPT for short) if it can be solved in time $f(k) \cdot |I|^c$ (often briefly referred to as FPT-*time*) for an instance I of size $|I|$ with parameter k, where f is a computable function and c is a constant. Given a parameterized problem P, a *kernel* is a polynomial-time computable function which associates with each instance of P an equivalent instance of P whose size is bounded by a function h of the parameter. When h is a polynomial, the kernel is said to be *polynomial*. See the book [6] for more details.

In order to prove that such polynomial kernel is unlikely, we need additional definitions and results.

▶ **Definition 1** (Cross-Composition [4]). *We say that a problem L* cross-composes *to a parameterized problem Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which given t instances $x_1, x_2, \ldots, x_t$ of L belonging to the same equivalence class $\mathcal{R}$, computes an instance (x^*, k^*) of Q in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that (i) $(x^*, k^*) \in Q \iff x_i \in L$ for some i and (ii) k^* is bounded by a polynomial in $(\max_i |x_i| + \log t)$.*

This definition is useful for the following result, which we will use to prove that a polynomial kernel for Longest Run Subsequence with parameter $|\Sigma|$ is unlikely.

▶ **Theorem 2** ([4]). *If an NP-hard problem L has a cross-composition into a parameterized problem Q and Q has a polynomial kernel, then NP $\subseteq$ coNP/poly.*

For our FPT algorithm, we will reduce our problem to another problem, called k-Multilinear Detection problem (k-MLD), which can be solved efficiently. In this problem, we are given a polynomial over a set of variables X, represented as an arithmetic circuit $\mathcal{C}$, and the question is to decide if this polynomial contains a multilinear term of degree exactly k. A polynomial is a sum of monomials. The *degree* of a monomial is the sum of its variables degrees and a monomial is *multilinear* if the degree of all its variables is equal to 1 (therefore, a multilinear monomial of degree k contains k different variables). For example, $x_1^2x_2 + x_1x_2x_3$ is a polynomial over 3 variables, both monomials are of degree 3 but only the second one is multilinear.

Note that the size of the polynomial could be exponentially large in $|X|$ and thus we cannot just check each monomial. We will therefore encode the polynomial in a compressed form: the circuit $\mathcal{C}$ is represented as a Directed Acyclic Graph (DAG), where leaves are variables X and internal nodes are multiplications or additions. The following result is fundamental for k-MLD.

▶ **Theorem 3** ([8, 11]). *There exists a randomized algorithm solving k-MLD in time $O(2^k|C|)$ and $O(|C|)$ space.*

Approximation. In Section 5, we prove the APX-hardness of Longest Run Subsequence with at most two occurrences for each symbol in Σ, by designing an L-reduction from Maximum Independent Set on cubic graphs. We recall here the definition of L-reduction. Notice that, given a solution S of a problem (A or B in the definition), we denote by $val(S)$ the value of S (for example, in our problem, the length of a run subsequence).

▶ **Definition 4** (L-reduction [9]). *Let A and B be two optimization problems. Then A is said to be L-reducible to B if there are two constants $\alpha, \beta > 0$ and two polynomial-time computable functions f, g such that: (i) f maps an instance I of A into an instance I' of B such that $opt_B(I') \leqslant \alpha \cdot opt_A(I)$, (ii) g maps each solution S' of I' into a solution S of I such that $|val(S) - opt_A(I)| \leqslant \beta \cdot |val(S') - opt_B(I')|$.*

L-reductions are useful in order to apply the following theorem.

▶ **Theorem 5** ([9]). *Let A and B be two optimization problems. If A is L-reducible to B and B has a PTAS, then A has a PTAS.*

Parameterized Complexity Status of the Problem

In the paper, we consider the parameterized complexity of Longest Run Subsequence under the different parameterizations. We consider the following parameters:

- The length k of the solution of Longest Run Subsequence
- The size $|\Sigma|$ of the alphabet
- The number r of runs in a solution of Longest Run Subsequence

Notice that $r \leqslant |\Sigma| \leqslant k$. Indeed, there always exists a solution consisting of one occurrence for each symbol in Σ, hence we can assume that $|\Sigma| \leqslant k$. Clearly, $r \leqslant |\Sigma|$, since each run in a solution of Longest Run Subsequence is associated with a distinct symbol of Σ.

In Table 1, we present the status of the parameterized complexity of Longest Run Subsequence for these parameters.

It is easy to see that Longest Run Subsequence has a polynomial kernel for parameter k.

Table 1 Parameterized complexity status for the three different parameters considered in this paper. Since these parameters are in decreasing value order, note that positive results propagate upwards, while negative results propagate downwards. In bold the new results we present in this paper.

	FPT	Poly Kernel
k	Yes (Obs. 6)	**Yes** (Obs. 6)
$\|\Sigma\|$	Yes [10]	**No** (Th. 9)
r	**Yes & Poly Space** (Th. 8)	**No** (Cor. 10)

▶ **Observation 6.** Longest Run Subsequence *has a* k^2 *kernel.*

Proof. First, notice that if there exists an a-run R' of length at least k, for some $a \in \Sigma$, then R' is a solution of Longest Run Subsequence. Also note that if $|\Sigma| \geqslant k$, let R^+ be a subsequence of S consisting of one occurrence of each symbol of Σ (notice that it is always possible to define such a solution). Then R^+ is a solution of Longest Run Subsequence of sufficient size.

Therefore, we can assume that S is defined over an alphabet $|\Sigma| < k$ and that each symbol has less then k occurrences (otherwise there exists an a-run of length at least k for some $a \in \Sigma$). Hence Longest Run Subsequence has a kernel of size k^2. ◀

Schrinner et al. prove that Longest Run Subsequence is in FPT for parameter $|\Sigma|$, using exponential space [10]. Due to a folklore result [6], this also implies that there is a kernel for this parameter. We will prove that there is no polynomial kernel for this parameter in Section 4.

3 An FPT Algorithm for Parameter Number of Runs

In this section, we consider Longest Run Subsequence when parameterized by the number of different runs, denoted by r, in the solution, that is whether there exists a solution of Longest Run Subsequence consisting of exactly r runs such that it has length at least k. We present a randomized fixed-parameter algorithm for Longest Run Subsequence based on multilinear monomial detection.

The algorithm we present is for a variant of Longest Run Subsequence that asks for a run subsequence R of S such that (1) $|R| = k$ and (2) R contains exactly r runs. In order to solve the general problem where we only ask for a solution of length at least k, we need to apply the algorithm for each k, with $r \leqslant k \leqslant |S|$.

Now, we describe the circuit on which our algorithm is based on. The set of variables is:

$$\{x_a : a \in \Sigma\}$$

Essentially, x_a represents the fact that we take an a-run (not necessarily maximal) in a substring of S.

Define a circuit $\mathcal{C}$ as follows. It has a root P and a set of intermediate vertices $P_{i,l,h}$, with $1 \leqslant i \leqslant |S|$, $1 \leqslant l \leqslant r$ and $1 \leqslant h \leqslant k$. The multilinear monomials of $P_{i,l,h}$ informally encode a run subsequence of $S[1, i]$ having length h and consisting of l runs. $P_{i,l,h}$ is recursively defined as follows:

$$P_{i,l,h} = \begin{cases} P_{i-1,l,h} + \sum_{j:1 \leqslant j \leqslant i-1} P_{j,l-1,h-z} x_a & \text{if } i \geqslant 1, l \geqslant 1, h \geqslant 1, \\ & 1 \leqslant z \leqslant i-j-1, S[i] = a,\, a \in \Sigma, \\ & \text{and } S[j+1,i] \text{ contains an } a\text{-run} \\ & \text{of length } z \\ 1 & \text{if } i \geqslant 0, l = h = 0, \\ 0 & \text{if } i = 0, l > 0 \text{ or } h > 0. \end{cases} \tag{1}$$

Then, define $P = P_{|S|,r,k}$.

Next, we show that we can consider the circuit $\mathcal{C}$ to compute a run subsequence of S.

▶ **Lemma 7.** *There exists a run subsequence of S of length h consisting of l runs over symbols $a_1, \dots a_l$ if and only if there exists a multilinear monomial in $\mathcal{C}$ consisting of l monomials $x_{a_1}, \dots, x_{a_l}$.*

Proof. We will prove that there is a run subsequence of S of length k consisting of l runs over symbols $a_1, \dots a_l$ if and only if there exists a multilinear monomial in $\mathcal{C}$ of degree l, consisting of l distinct variables $x_{a_1}, \dots, x_{a_l}$. In order to prove this result, we prove by induction on i , $1 \leqslant i \leqslant |S|$, that there exists a run subsequence R of $S[1...i]$, such that $|R| = h$ and R contains l runs, an a_z-run for each $a_z \in \Sigma$, $1 \leqslant z \leqslant l$, if and only if there exists a multilinear monomial $x_{a_1} \dots x_{a_l}$ in $P_{i,l,h}$.

We start with the case $i = 1$. Assume that there is a run subsequence consisting of a single run of length 1 (say an a_1-run). It follows that $S[1] = a_1$ and, by Equation 1, $P_{1,1,1} = P_{0,0,0} \cdot x_{a_1} = x_{a_1}$. Conversely, if $P_{1,1,1} = P_{0,0,0} \cdot x_{a_1} = x_{a_1}$, then by construction $S[1] = a_1$, which is a run of length 1.

Assume that the lemma holds for $j < i$, we prove that it holds for i.

($\Rightarrow$) Assume that there exists a run subsequence R of $S[1, i]$ that consists of l runs and that has length h. Let the l runs in R be over symbols $a_1, \dots, a_l$ and assume that the rightmost run in R is an a_l-run. If $S[i]$ does not belong to the a_l-run in R, then R is a run subsequence in $S[1, i-1]$ and by induction hypothesis $P_{i-1,l,h}$ contains a multilinear monomial of degree l over variables $x_{a_1} \dots x_{a_l}$. If $S[i]$ belongs to the a_l-run in R, then consider the a_l-run in R and assume that it belongs to substring $S[j+1, i]$ of S, with $1 \leqslant j+1 \leqslant i$, and that it has length z. Consider the run subsequence R' of S obtained from R by removing the a_l-run. Then, R' is a run subsequence of $S[1, j]$ that does not contain a_l (hence it contains $l-1$ runs) and has length $h - z$. By induction hypothesis, $P_{j,l-1,h-z}$ contains a multilinear monomial of length $l-1$ over variables $x_{a_1} \dots x_{a_{l-1}}$. Hence by the first case of Equation 1, it follows that $P_{i,l,h}$ contains a multilinear monomial of length l over variables $x_{a_1} \dots x_{a_l}$.

($\Leftarrow$) Assume that $P_{i,l,h}$ contains a multilinear monomial of length l over variables $x_{a_1} \dots x_{a_l}$, we will prove that there is a run subsequence of S of length k consisting of l runs. By Equation 1, it follows that (1) $P_{i-1,l,h}$ contains a multilinear monomial of length l over variables $x_{a_1} \dots x_{a_l}$ or (2) $P_{j,l-1,h-z}$, for some $1 \leqslant j \leqslant i-1$, contains a multilinear monomial of length $l-1$ that does not contain one of $x_{a_1} \dots x_{a_l}$ (without loss of generality x_{a_l}) and $S[j+1, i]$ contains an a_l-run of length z.

In case (1), by induction hypothesis there exists a run subsequence in $S[1, i-1]$ (hence also in $S[1, i]$) of length h consisting of l runs over symbols $a_1, \dots, a_l$.

In case (2), by induction hypothesis there exists a run subsequence R' of $S[1, j]$ of length $h - z$ consisting of $l-1$ runs over symbols $a_1, \dots, a_{l-1}$. Now, by concatenating R' with the a_l-run of length z in $S[j+1, i]$, we obtain a run subsequence of R of $S[1, i]$ consisting of l runs and having length h. ◀

▶ **Theorem 8.** Longest Run Subsequence *can be solved by a randomized algorithm in* $O(2^r r|S|^3)$ *time and polynomial space.*

Proof. The correctness of the randomized algorithm follows by Lemma 7.

We compute P in polynomial time and we decide if $P_{|S|,r,k}$ contains a multilinear monomial of degree r in $O(2^r r|S|^2)$ time and polynomial space. The result follows from Lemma 7, Theorem 3, and from the observation that $|\mathcal{C}| = |S| \cdot l \cdot r$, with $l \leqslant |S|$. Finally, we have to iterate the algorithm for each k, with $r \leqslant k \leqslant |S|$, thus the overall time complexity is $O(2^r r|S|^3)$. ◀

4 Hardness of Kernelization

As discussed in Section 2, Longest Run Subsequence has a trivial polynomial kernel for parameter k and its FPT status implies an (exponential) kernel for parameters $|\Sigma|$ and r. In the following, we will prove that it is unlikely that Longest Run Subsequence admits a polynomial kernel for parameter $|\Sigma|$ and parameter r.

▶ **Theorem 9.** Longest Run Subsequence *does not admit a polynomial kernel for parameter* $|\Sigma|$*, unless NP* $\subseteq$ *coNP/poly.*

Proof. We will define an OR-cross-composition (see Definition 1) from the Longest Run Subsequence problem itself, whose unparameterized version is NP-Complete [10].

Consider t instances $(S_1, \Sigma_1, k_1), (S_2, \Sigma_2, k_2), \ldots, (S_t, \Sigma_t, k_t)$ of Longest Run Subsequence, where, for each i with $1 \leqslant i \leqslant t$, S_i is the input string built over the alphabet Σ_i, and $k_i \in \mathbb{N}$ is the length of the solution, respectively. We will define an equivalence relation $\mathcal{R}$ such that strings that are not encoding valid instances are equivalent, and two valid instances $(S_i, \Sigma_i, k_i), (S_j, \Sigma_j, k_j)$ are equivalent if and only if $|S_i| = |S_j|$, $|\Sigma_i| = |\Sigma_j|$, and $k_i = k_j$. We now assume that $|S_i| = n$, $|\Sigma_i| = m$ and $k_i = k$ for all $1 \leqslant i \leqslant t$.

We will build an instance of Longest Run Subsequence (S', k', Σ') where S' is a string built over the alphabet Σ' and k' an integer such that there is a solution of size at least k' for S' iff there is an i, $1 \leqslant i \leqslant t$ such that there is a solution of size at least k in S_i.

We first show how to redefine the input strings S_1, S_2, …, S_t, such that they are all over the same alphabet. Notice that this will not be an issue, since we will construct a string S' such that a solution of Longest Run Subsequence is not spanning over two different input strings. For all instances (S_i, Σ_i, k_i), $1 \leqslant i \leqslant t$, we consider any ordering of the symbols in Σ_i and we define a string $\sigma(S_i)$ starting from S_i, by replacing the j-th, $1 \leqslant j \leqslant m$, symbol of Σ_i by j, that is its position in the ordering of Σ_i. That way, it is clear that all strings $\sigma(S_i)$, $1 \leqslant i \leqslant t$, are built over the same alphabet $\{1, 2, \ldots, m\}$.

Now, the instance (S', k', Σ') of Longest Run Subsequence is build as follows. First, Σ' is defined as follows:

$$\Sigma' = \{1, 2, \ldots, m\} \cup \{\#, \$\}$$

where $\#$ and $\$$ are two symbols not in Σ.

The string S' is defined as follows:

$$S' = \$^{2n}\sigma(S_1)\#^{2n}\$^{2n}\sigma(S_2)\#^{2n}, ..., \$^{2n}\sigma(S_t)\#^{2n}$$

where $\$^{2n}$ ($\#^{2n}$, respectively) is a string consisting of the repetition $2n$ times of the symbol $\$$ ($\#$, respectively).

Finally,

$k' = k + (t+1)(2n).$

Since we are applying OR-cross-composition for parameter $|\Sigma|$, we need to show that property (ii) of Definition 1 holds. By construction, we see that $|\Sigma'| = m + 2$, which is independent of n and t and it is bounded by the size of the largest input instance.

We now show that S' contains a run subsequence of size at least k' if and only if there exists at least one string S_i, $1 \leqslant i \leqslant t$, that contains a run subsequence of size at least k.

($\Leftarrow$) First, assume that some S_i, with $1 \leqslant i \leqslant t$, contains a run subsequence R_i of length at least k. Then, define the following run subsequence R' of S', obtained by concatenating these substrings of S':

- The concatenation of the leftmost i-th substrings $\$^{2n}$ of S',
- The substring $\sigma(R_i)$ of S_i,
- The concatenation of the rightmost $(t-i+1)$-th substrings $\#^{2n}$ of S'.

It follows that S' contains a run subsequence of length at least $i \times (2n) + k + (t-i+1) \times (2n) = k'$.

($\Rightarrow$) Conversely, assume now that S' contains a run subsequence R' of length at least k'. First, we prove that R' contains exactly one \$-run and one #-run. Indeed, if it is not the case, we can add the leftmost (the rightmost, respectively) substring $\$^{2n}$ ($\#^{2n}$, respectively) as a run of R'.

Consider a run r in R', which is either the \$-run or the #-run of R'. Assume that R' contains a substring

$rR'(S_i)R'(S_j)$

or a substring

$R'(S_i)R'(S_j)\ r$

with $1 \leqslant i < j \leqslant t$, where $R'(S_i)$ ($R'(S_j)$, respectively) is a substring of $\sigma(S_i)$ (of $\sigma(S_j)$, respectively). We consider without loss of generality the case that $rR'(S_i)R'(S_j)$ is a substring of R'. Then, we can modify R', increasing its length, as follows: we remove $R'(S_i)$ and extend the run r with a string $\$^{2n}$ or a string $\#^{2n}$ (depending on the fact that r is a \$-run or a #-run, respectively) that is between $\sigma(S_i)$ and $\sigma(S_j)$. The size of R' is increased, since $|R'(S_i)| \leqslant n$.

Now, assume that R' contains a substring

$r = \#^{2n}\ R'(S_i)\ \$^{2n}$

where $R'(S_i)$ is a substring of $\sigma(S_i)$. We can replace r with the substring $\#^{2n}\#^{2n}\$^{2n}$, where $\#^{2n}$ is the substring between $\sigma(S_i)$ and $\sigma(S_{i+1})$ in S'. Again, the size of R' is increased, since $|R'(S_i)| \leqslant n$.

By iterating these modifications on R', we obtain that R' is one of the following string:

1. A prefix $\$^{j(2n)}$ concatenated with a substring $R'(S_j)$ of $\sigma(S_j)$, for some $1 \leqslant j \leqslant t$, concatenated with a suffix $\#^{2n(t-j+1)}$
2. A substring $R'(S_1)$ of $\sigma(S_1)$ concatenated with a string of $\#^{2nj}$ concatenated with a string of $\$^{2n(t-j)}$ concatenated with a substring $R'(S_t)$ of $\sigma(S_t)$.

Notice that in this second case, it holds that $|R'| = (t \times 2n) + |R'(S_1)| + |R'(S_t)| < k'$, since $|R'(S_1)| + |R'(S_t)| \leqslant 2n$, and we can assume that $k \geqslant 1$. Hence R' must be a string described at point 1. It follows that

$$|R'| = 2n(t+1) + |R'(S_j)|$$

Since $|R'| = 2n(t+1) + |R'(S_j)|$, then $R'(S_j)$ has length at least k.

We have described an OR-cross-composition of LONGEST RUN SUBSEQUENCE to itself. By Theorem 2, it follows that LONGEST RUN SUBSEQUENCE does not admit a polynomial kernel for parameters $|\Sigma|$, unless NP $\subseteq$ coNP/poly. ◀

We can complement the FPT algorithm of Section 3, with a hardness of kernelization for the same parameter.

▶ **Corollary 10.** LONGEST RUN SUBSEQUENCE *does not admit a polynomial kernel for parameters r, unless NP $\subseteq$ coNP/poly.*

Proof. The result follows from Theorem 9 and from the fact that $r \leqslant |\Sigma|$. ◀

5 APX-hardness for Bounded Number of Occurrences

In this section, we show that LONGEST RUN SUBSEQUENCE is hard even when the number of occurrences of a symbol in the input string is bounded by two. We denote this restriction of the problem by 2-LONGEST RUN SUBSEQUENCE. Notice that if the number of occurrences of a symbol is bounded by one, then the problem is trivial, as a solution of LONGEST RUN SUBSEQUENCE can have only runs of length one.

We prove the result by giving a reduction from the MAXIMUM INDEPENDENT SET PROBLEM ON CUBIC GRAPHS (MISC), which is known to be APX-hard [1]. We recall the definition of MISC:

MAXIMUM INDEPENDENT SET PROBLEM ON CUBIC GRAPHS (MISC)
• **Input**: A cubic graph $G = (V, E)$.
• **Output**: Does there exist an independent set in G of size at least q?

Given a cubic graph $G = (V, E)$, with $V = \{v_1, \ldots v_n\}$ and $|E| = m$, we construct a corresponding instance S of 2-LONGEST RUN SUBSEQUENCE (see Fig. 2 for an example of our construction). First, we define the alphabet Σ:

$$\Sigma = \{w_i : 1 \leqslant i \leqslant n\} \cup \{x^i_{i,j}, x^j_{i,j}, e^1_{i,j}, e^2_{i,j} : \{v_i, v_j\} \in E, i < j\} \cup \{\natural_{i,z} : 1 \leqslant i \leqslant m+n, 1 \leqslant z \leqslant 3\}$$

This alphabet is of size $n + 4m + 3(m+n) = 4n + 7m$.

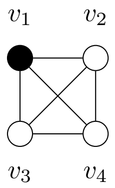
Now, we define a set of substrings of the instance S of 2-LONGEST RUN SUBSEQUENCE that we are constructing.

- For each $v_i \in V$, $1 \leqslant i \leqslant n$, such that v_i is adjacent to v_j, v_h, v_z, $1 \leqslant j < h < z \leqslant n$, we define a substring $S(v_i)$:

  $$S(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$$

  Notice that in the definition of $S(v_i)$ given above, we have assumed without loss of generality that $1 \leqslant i < j < h < z \leqslant n$. If, for example, $1 \leqslant j < i < h < z \leqslant n$, the symbol associated with $\{v_i, v_j\}$ is then $x^i_{j,i}$ and $S(v_i)$ is defined as follows:

  $$S(v_i) = w_i x^i_{j,i} x^i_{i,h} x^i_{i,z} w_i$$



$$\begin{aligned} S = & w_1 x^1_{1,2} x^1_{1,3} x^1_{1,4} w_1 \sharp_{1,1} \sharp_{1,2} \sharp_{1,3} \\ & w_2 x^2_{1,2} x^2_{2,3} x^2_{2,4} w_2 \sharp_{2,1} \sharp_{2,2} \sharp_{2,3} \\ & w_3 x^3_{1,3} x^3_{2,3} x^3_{3,4} w_3 \sharp_{3,1} \sharp_{3,2} \sharp_{3,3} \\ & w_4 x^4_{1,4} x^4_{2,4} x^4_{3,4} w_4 \sharp_{4,1} \sharp_{4,2} \sharp_{4,3} \\ & e^1_{1,2} x^1_{1,2} e^2_{1,2} e^1_{1,2} x^2_{1,2} e^2_{1,2} \sharp_{5,1} \sharp_{5,2} \sharp_{5,3} \\ & e^1_{1,3} x^1_{1,3} e^2_{1,3} e^1_{1,3} x^3_{1,3} e^2_{1,3} \sharp_{6,1} \sharp_{6,2} \sharp_{6,3} \\ & e^1_{1,4} x^1_{1,4} e^2_{1,4} e^1_{1,4} x^4_{1,4} e^2_{1,4} \sharp_{7,1} \sharp_{7,2} \sharp_{7,3} \\ & e^1_{2,3} x^2_{2,3} e^2_{2,3} e^1_{2,3} x^3_{2,3} e^2_{2,3} \sharp_{8,1} \sharp_{8,2} \sharp_{8,3} \\ & e^1_{2,4} x^2_{2,4} e^2_{2,4} e^1_{2,4} x^4_{2,4} e^2_{2,4} \sharp_{9,1} \sharp_{9,2} \sharp_{9,3} \\ & e^1_{3,4} x^3_{3,4} e^2_{3,4} e^1_{3,4} x^4_{3,4} e^2_{3,4} \sharp_{10,1} \sharp_{10,2} \sharp_{10,3} \end{aligned}$$

Figure 2 A sample cubic graph (with 4 nodes and 6 edges) and the associated sequence. Black vertex of the graph corresponds to an independent set (of size 1 here), red symbols in the sequence correspond to the subsequence (of size $5 \cdot 1 + 4 \cdot 3 + 3 \cdot 6 + 3 \cdot 10 = 65$).

- For each edge $\{v_i, v_j\} \in E$, with $1 \leqslant i < j \leqslant n$, we define a substring $S(e_{ij})$:

  $S(e_{ij}) = e^1_{i,j} x^i_{i,j} e^2_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$

- We define separation substrings $S_{Sep,i}$, with $1 \leqslant i \leqslant m + n$:

  $S_{Sep,i} = \sharp_{i,1} \sharp_{i,2} \sharp_{i,3}$

Now, given the lexical ordering[1] of the edges of G, the input string S is defined as follows (we assume that $\{v_1, v_z\}$ is the first edge and $\{v_p, v_t\}$ is the last edge in the lexicographic ordering of E):

$$S = S(v_1) S_{Sep,1} S(v_2) S_{Sep,2} \dots S(v_n) S_{Sep,n} S(e_{1,z}) S_{Sep,n+1} \dots S(e_{p,t}) S_{Sep,n+m}$$

Now, we prove some properties on the string S.

▶ **Lemma 11.** *Let $G = (V, E)$ be an instance of* MISC *and let S be the corresponding built instance of* 2-Longest Run Subsequence. *Then S contains at most two occurrences for each symbol of Σ.*

Proof. Notice that each symbol w_i, $1 \leqslant i \leqslant n$, appears only in substring $S(v_i)$ of S. Symbols $e^1_{i,j}$, $e^2_{i,j}$, with $\{v_i, v_j\} \in E$ and $1 \leqslant i < j \leqslant n$, appear only in substring $S(e_{i,j})$ of S. Each symbol $\sharp_{i,z}$, with $1 \leqslant i \leqslant m+n$ and $1 \leqslant z \leqslant 3$, appears only in substring $S_{Sep,i}$ of S. Finally, each symbol $x^i_{i,j}$, with $\{v_i, v_j\} \in E$, appears once in exactly two substrings of S, namely $S(v_i)$ and $S(e_{i,j})$. ◀

Now, we prove a property of solutions of 2-Longest Run Subsequence relative to separation substrings.

▶ **Lemma 12.** *Let $G = (V, E)$ be an instance of* MISC *and let S be the corresponding instance of* 2-Longest Run Subsequence. *Given a run subsequence R of S, if R does not contain some separation substring $S_{Sep,i}$, with $1 \leqslant i \leqslant m + n$, then there exists a run subsequence R' of S that contains $S_{Sep,i}$ and such that $|R'| > |R|$.*

[1] $\{v_i, v_j\} < \{v_h, v_z\}$ (assuming $i < j$ and $h < z$) if and only if $i < h$ or $i = h$ and $j < z$.

Proof. Notice that, since R does not contain substring $S_{Sep,i}$, with $1 \leqslant i \leqslant m+n$, it must contain a run r that connects two symbols that are on the left and on the right of $S_{Sep,i}$ in S, otherwise $S_{Sep,i}$ can be added to R increasing its length. Since each symbol in S, hence also in R, has at most two occurrences (see Lemma 11), then $|r| = 2$. Then, starting from R, we can compute in polynomial time a run subsequence R' by removing run r and by adding substring $S_{Sep,i}$. Notice that, after the removal of r, we can add $S_{Sep,i}$ since it contains three symbols each one having a single occurrence in S. Since $|S_{Sep,i}| = 3$, it follows that $|S_{Sep,i}| > r$ and $|R'| > |R|$. ◀

Given a cubic graph $G = (V, E)$ and the corresponding instance S of 2-Longest Run Subsequence, a run subsequence R of 2-Longest Run Subsequence on instance S is called *canonical* if:

- for each $S_{Sep,i}$, $1 \leqslant i \leqslant m+n$, R contains $S_{Sep,i}$ (a substring denoted by $R_{Sep,i}$)
- for each $S(v_i)$, with $v_i \in V$, R contains a substring $R(v_i)$ such that either $R(v_i) = w_i w_i$ or it is a substring of length 4 ($w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$ or $x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$); moreover if $\{v_i, v_j\} \in E$, then at least one of $R(v_i)$ or $R(v_j)$ has length 4
- for each $S(e_{i,j})$, with $\{v_i, v_j\} \in E$, R contains a substring $R(e_{i,j})$ such that $R(e_{i,j})$ is either of length 4 ($e^1_{i,j} x^i_{i,j} e^2_{i,j} e^2_{i,j}$ or $e^1_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$), if one of $R(v_i)$, $R(v_j)$ has length 2, or of length 3 ($e^1_{i,j} e^1_{i,j} e^2_{i,j}$ or $e^1_{i,j} e^2_{i,j} e^2_{i,j}$).

▶ **Lemma 13.** *Let $G = (V, E)$ be an instance of* MISC *and let S be the corresponding instance of* 2-Longest Run Subsequence*. Given a run subsequence R of S, we can compute in polynomial time a canonical run subsequence of S of length at least $|R|$.*

Proof. Consider a run subsequence R of S. First, notice that by Lemma 12 we assume that R contains each symbol $\sharp_{i,p}$, with $1 \leqslant i \leqslant n+m$ and $1 \leqslant p \leqslant 3$. We start by proving some bounds on the run subsequence of $S(v_i)$ and $S(e_{i,j})$.

Consider a substring $R(v_i)$ of $S(v_i)$, $1 \leqslant i \leqslant n$. Each run subsequence of $S(v_i)$ can have length at most 4, since $|S(v_i)| = 5$ and if run $w_i w_i$ belongs to $R(v_i)$, then $R(v_i) = w_i w_i$. It follows that if $|R(v_i)| > 2$, then it cannot contain the two occurrences of symbol w_i. Notice that the two possible run subsequences of length 4 of $S(v_i)$ are $x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$ and $w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$.

Consider a run subsequence $R(e_{i,j})$ of $S(e_{ij}) = e^1_{i,j} x^i_{i,j} e^2_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$. First, we prove that a run subsequence of $S(e_{ij})$ has length at most 4 and in this case it must contain at least one of $x^i_{i,j}$, $x^j_{i,j}$. By its interleaved construction, at most one of runs $e^1_{i,j} e^1_{i,j}$, $e^2_{i,j} e^2_{i,j}$ can belong to $R(e_{i,j})$. Moreover if $e^1_{i,j} e^1_{i,j}$ ($e^2_{i,j} e^2_{i,j}$, respectively) belongs to $R(e_{i,j})$, then $|R(e_{i,j})| \leqslant 4$, since the longest run in $S(e_{i,j})$ is then $e^1_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$ ($e^1_{i,j} x^i_{i,j} e^2_{i,j} e^2_{i,j}$, respectively). If none of runs $e^1_{i,j} e^1_{i,j}$, $e^2_{i,j} e^2_{i,j}$ belongs to $R(e_{i,j})$, then $|R(e_{i,j})| \leqslant 4$, since $|S(e_{i,j})| = 6$; in this case both $x^i_{i,j}$ and $x^j_{i,j}$ must be in $R(e_{i,j})$ to have $|R(e_{i,j})| = 4$.

Now, we compute a canonical run subsequence R' of S of length at least $|R|$. Consider $R(v_i)$, $1 \leqslant i \leqslant n$, and $R(e_{i,j})$, with $\{v_i, v_j\} \in E$.

If $|R(v_i)| = 4$, then define $R'(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$ (or equivalently define $R'(v_i) = x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$).

If $|R(v_i)| = 3$, then by construction of $S(v_i)$ at least two of $x^i_{i,j}$, $x^i_{i,h}$, $x^i_{i,z}$ belong to $R(v_i)$. Then, at most one of $R(e_{i,j})$, $R(e_{i,h})$, $R(e_{i,z})$ can contain a symbol in $\{x^i_{i,j}, x^i_{i,h}, x^i_{i,z}\}$, assume w.lo.g. that $x^i_{i,j}$ belongs to $R(e_{i,j})$. We define $R'(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$ (or equivalently $R'(v_i) = x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$) and $R'(e_{i,j}) = e^1_{i,j} e^1_{i,j} e^2_{i,j}$ (or equivalently $R'(e_{i,j}) = e^1_{i,j} e^2_{i,j} e^2_{i,j}$).

Since $|R(e_{i,j})| \leqslant 4$, we have that

$$|R'(e_{i,j})| \geqslant |R(e_{i,j})| - 1$$

and

$$|R'(v_i)| = |R(v_i)| + 1.$$

It follows that the size of R' is not decreased with respect to the length of R.

If $|R(v_i)| = 2$, then define $R'(v_i) = w_i w_i$.

By construction of R', either $|R'(v_i)| = 4$ and

$$R'(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z} \text{ or } R'(v_i) = x^i_{i,j} x^i_{i,h} x^i_{i,z} w_i$$

or $|R'(v_i)| = 2$ and

$$R'(v_i) = w_i w_i.$$

Again the size of R' is not decreased with respect to the size of R.

In order to compute a canonical run subsequence, we consider an edge $\{v_i, v_j\} \in E$ and the run subsequences $R'(v_i)$ and $R'(v_j)$ of $S(v_i)$, $S(v_j)$, respectively. Consider the case that $R'(v_i) = w_i w_i$ and $R'(v_j) = w_j w_j$. Then by construction $|R'(e_{i,j})| = 4$ and assume with loss of generality that $R'(e_{i,j}) = e^1_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$. Now, we can modify R' so that

$$R'(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$$

by eventually removing $x^i_{i,h}$, $x^i_{i,z}$ from $R'(e_{i,h})$ and $R'(e_{i,z})$. In this way, we decrease by at most one the length of each of $R'(e_{i,h})$, $R'(e_{i,z})$ and we increase of two the length of $R'(v_i)$. It follows that the length of R' is not decreased by this modification. By iterating this modification, we obtain that for each edge $\{v_i, v_j\} \in E$ at most one of $R'(v_i)$, $R'(v_j)$ has length two.

The run subsequence R' we have built is then a canonical run subsequence of S such that $|R'| \geqslant |R|$. ◀

Now, we are ready to prove the main results of the reduction.

▶ **Lemma 14.** *Let $G = (V, E)$ be an instance of* MISC *and let S be the corresponding instance of* 2-Longest Run Subsequence*. Given an independent set I of size at least q in G, we can compute in polynomial time a run subsequence of S of length at least $5q + 4(n - q) + 3m + 3(n + m)$.*

Proof. We construct a subsequence run R of S as follows:

- For each $v_i \in I$, define for the substring $S(v_i)$ the run subsequence $R(v_i) = w_i w_i$;
- For each $v_i \in V \setminus I$, define for the substring $S(v_i)$ the run subsequence $R(v_i) = w_i x^i_{i,j} x^i_{i,h} x^i_{i,z}$;
- For each $\{v_i, v_j\} \in E$, if $v_i \in I$ (or $v_j \in I$, respectively) define for the substring $S(e_{i,j})$ the run subsequence $R(e_{i,j}) = e^1_{i,j} x^i_{i,j} e^2_{i,j} e^2_{i,j}$ ($R(e_{i,j}) = e^1_{i,j} e^1_{i,j} x^j_{i,j} e^2_{i,j}$, respectively); if both $v_i, v_j \in V \setminus I$, define for the substring $S(e_{i,j})$ the run subsequence $R(e_{i,j}) = e^1_{i,j} e^1_{i,j} e^2_{i,j}$

Moreover, R contains each separation substring of S, denoted by $R_{Sep,i}$, $1 \leqslant i \leqslant n + m$.

First, we prove that R is a run subsequence, that is R contains a single run for each symbol in Σ. This property holds by construction for each symbol in Σ having only occurrences in $R(v_i)$, with $v_i \in V$, $R(e_{i,j})$, with $\{v_i, v_j\} \in E$, and $R_{Sep,i}$, $1 \leqslant i \leqslant n + m$. What is left to prove is that $x^i_{i,j}$ appears in at most one of $R(v_i)$, with $v_i \in V$, $R(e_{i,j})$, with $\{v_i, v_j\} \in E$. Indeed, by construction, $R(e_{i,j})$ contains $x^i_{i,j}$ only if $R(v_i) = w_i w_i$. It follows that R is a run subsequence of S.

Consider the length of R. For each $v_i \in V \setminus I$, R contains a run subsequence of $S(v_i)$ of length 4. For each $v_i \in I$, R contains a run subsequence of length 2. For each $\{v_i, v_j\} \in E$, with $v_i, v_j \in V \setminus I$, R contains a run subsequence of $S(e_{i,j})$ of length 3; for each $\{v_i, v_j\} \in E$, with $v_i \in I$ or $v_j \in I$, R contains a run subsequence of $S(e_{i,j})$ of length 4. Finally, each separation substring $R_{Sep,i}$, $1 \leqslant i \leqslant n + m$, in R has length 3. Hence the total length of R is at least $5q + 4(n - q) + 3m + 3(n + m)$ (by accounting, for each $R(v_i)$ of length 2, the increasing of the length of the three run subsequences $R(e_{i,j})$, $R(e_{i,h})$, $R(e_{i,z})$ from 3 to 4 to $R(v_i)$). ◀

▶ **Lemma 15.** *Let $G = (V, E)$ be an instance of* MISC *and let S be the corresponding instance of* 2-Longest Run Subsequence. *Given a run subsequence of S of length at least $5q + 4(n - q) + 3m + 3(n + m)$, we compute in polynomial time an independent of G of size at least q.*

Proof. Consider a run subsequence R of S of length at least $5q + 4(n - q) + 3m + 3(n + m)$. By Lemma 13, we assume that R is a canonical run subsequence of S. It follows that we can define an independent V' of size at least q in G as follows:

$$V' = \{v_i : |R(v_i)| = 2\}$$

By the definition of canonical run subsequence, it follows that V' is an independent set, since if $|R(v_i)| = |R(v_j)| = 2$, with $1 \leqslant i, j \leqslant n$, then $\{v_i, v_j\} \notin E$. Furthermore, by the definition of canonical run subsequence, since $|R| \geqslant 5q + 4(n - q) + 3m + 3(n + m)$, there are at least q run subsequences $R(v_i)$, with $1 \leqslant i \leqslant n$, of length two such that $|R(e_{i,j})| = |R(e_{i,h})| = |R(e_{i,z})| = 4$, with $\{v_i, v_j\}, \{v_i, v_h\}, \{v_i, v_z\} \in E$. It follows that $|V'| \geqslant q$. ◀

Now, we can prove the main result of this section.

▶ **Theorem 16.** 2-Longest Run Subsequence *is APX-hard.*

Proof. We have shown a reduction from MISC to 2-Longest Run Subsequence. By Lemma 11, the instance of 2-Longest Run Subsequence we have built consists of a string with at most two occurrences for each symbol. We will now show that this reduction is an L-reduction from MISC to 2-Longest Run Subsequence (see Definition 4).

Consider an instance I of MISC and a corresponding instance I' of 2-Longest Run Subsequence. Then, given any optimal solution $opt(I')$ of 2-Longest Run Subsequence on instance I', by Lemma 15 it holds that

$$opt(I') \leqslant 5 \cdot opt(I) + 4(n - opt(I)) + 3m + 3(n + m) = opt(I) + 7n + 6m$$

In a cubic graph $G = (V, E)$, $|E| = \frac{3}{2}|V|$, hence $m = \frac{3}{2}n$. Furthermore, we can assume that an independent set has size at least $\frac{n}{4}$. Indeed, such an independent set V' can be greedily computed as follows: pick a vertex v in the graph, add it to V' and delete $N[v]$ from G. At each step, we add one vertex in V' and we delete at most 4 vertices from G.

Since $m = \frac{3}{2}n$ and $n \leqslant 4 \cdot opt(I)$, we thus have that

$$opt(I') \leqslant opt(I) + 7n + 6m = opt(I) + 16 \cdot n \leqslant opt(I) + 64 \cdot opt(I)$$

and then $\alpha = 65$ in Definition 4.

Conversely, consider a solution S' of length $5q+4(n-q)+3m+3(n+m)$ of 2-Longest Run Subsequence on instance I'. First, notice that by Lemma 13, we can assume that S' and $opt(I')$ are both canonical. By Lemma 14, if $opt(I) = p$, then $opt(I') \geqslant 5p + 4(n-p) + 3m + 3(n+m)$. By Lemma 15, starting from S', we can compute in polynomial time a solution V' of MISC on instance I, with $|V'| \geqslant q$. It follows that

$$|opt(I) - |V'|| \leqslant |opt(I) - q| = |p - q| =$$

$$|5p + 4(n-p) + 3m + 3(n+m) - (5q + 4(n-q) + 3m + 3(n+m))| \leqslant$$

$$|opt(I') - (5q + 4(n-q) + 3m + 3(n+m))|$$

Then $\beta = 1$ in Definition 4. Thus we indeed have designed an L-reduction, therefore, the APX-hardness of 2-Longest Run Subsequence follows from the APX-hardness of MISC [1] and from Theorem 5. ◀

6 Conclusion

In this paper, we deepen the understanding of the complexity of the recently introduced problem Longest Run Subsequence. We show that the problem remains hard (even from the approximation point of view) also in the very restricted setting where each symbol occurs at most twice. We also complete the parameterized complexity landscape. From the more practical point of view, it is however unclear how our FPT algorithm could compete with implementations done in [10].

An interesting future direction is to further investigate the approximation complexity of the Longest Run Subsequence problem beyond APX-hardness. Note that a trivial $\min(|\Sigma|, occ)$-approximation algorithm (occ is the maximum number of occurrences of a symbol in the input S) can be designed by taking the solution having maximum length between: (1) a solution having one occurrence for each symbol in Σ and (2) a solution consisting of the a-run of maximum length, among each $a \in \Sigma$. This leads to a $\sqrt{|S|}$-approximation algorithm. Indeed, if the a-run of maximum length is greater than $\sqrt{|S|}$, then solution (2) has length at least $\sqrt{|S|}$, thus leading to the desired approximation factor. If this is not the case, then each symbol in Σ has less then $\sqrt{|S|}$ occurrences, thus a solution of Longest Run Subsequence on instance S has at length smaller than $|\Sigma|\sqrt{|S|}$. It follows that (1) is a solution with the desired approximation factor. We let for future work closing the gap between the APX-hardness and the $\sqrt{|S|}$-approximation factor of Longest Run Subsequence.

References

1 Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.*, 237(1-2):123–134, 2000. doi:10.1016/S0304-3975(98)00158-3.

2 Michael Alonge, Sebastian Soyk, Srividya Ramakrishnan, Xingang Wang, Sara Goodwin, Fritz J. Sedlazeck, Zachary B Lippman, and Michael C. Schatz. Fast and accurate reference-guided scaffolding of draft genomes. *Genome Biology*, 20:244, 2019. doi:10.1101/519637.

3 Alberto Apostolico, Gad M. Landau, and Steven Skiena. Matching for run-length encoded strings. *J. Complex.*, 15(1):4–16, 1999. doi:10.1006/jcom.1998.0493.

4 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

5 Lauren Coombe, Vladimir Nikolic, Justin Chu, Inanc Birol, and René Warren. ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. *Bioinformatics*, 36:3885–3887, 2020. doi:10.1093/bioinformatics/btaa253.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

7 Manish Goel, Hequan Sun, Wen-Biao Jiao, and Korbinian Schneeberger. SyRI: finding genomic rearrangements and local sequence differences from whole-genome assemblies. *Genome Biology*, 20:277, 2019. doi:10.1186/s13059-019-1911-0.

8 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings*, volume 5125 of *LNCS*, pages 575–586. Springer, 2008. doi:10.1007/978-3-540-70575-8_47.

9 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

10 Sven Schrinner, Manish Goel, Michael Wulfert, Philipp Spohr, Korbinian Schneeberger, and Gunnar W. Klau. The longest run subsequence problem. In Carl Kingsford and Nadia Pisanti, editors, *20th International Workshop on Algorithms in Bioinformatics, WABI 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 172 of *LIPIcs*, pages 6:1–6:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.WABI.2020.6.

11 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.