ACADEMIC PRESS

Available at www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Journal of Computer and System Sciences 67 (2003) 757–771

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

On the parameterized complexity of the fixed alphabet *shortest common supersequence* and *longest common subsequence* problems

Krzysztof Pietrzak*

Computational Biology Lab, McGill University, Montréal, Canada

Received 1 October 2001; revised 13 March 2002

Abstract

We show that the fixed alphabet shortest common supersequence (SCS) and the fixed alphabet longest common subsequence (LCS) problems parameterized in the number of strings are $W[1]$-hard. Unless $W[1] = \mathrm{FPT}$, this rules out the existence of algorithms with time complexity of $O(f(k)n^{\alpha})$ for those problems. Here n is the size of the problem instance, α is constant, k is the number of strings and f is any function of k. The fixed alphabet version of the LCS problem is of particular interest considering the importance of sequence comparison (e.g. multiple sequence alignment) in the fixed length alphabet world of DNA and protein sequences.

Keywords: Parameterized complexity; Longest common subsequence; Shortest common supersequence; Multiple sequence alignment

1. Introduction

The shortest common supersequence (SCS) and the longest common subsequence (LCS) are classical problems in computer science.

Shortest common supersequence (SCS)
Instance: A set of strings $R = r_1, r_2, \ldots, r_k$ over an alphabet Σ, an integer λ.

*Current affiliation: Institute of Theoretical Computer Science, ETH Zentrum IFW E41, Haldeneggsteig 4, CH-8092 Zurich, Switzerland.

E-mail address: pietrzak@inf.ethz.ch.

0022-0000/03/$ - see front matter
doi:10.1016/S0022-0000(03)00078-3

Question: Does there exist a string $s \in \Sigma^*$ of length at most λ, that is a supersequence[1] of each string in R?

Longest common subsequence (LCS)

Instance: A set of strings $R = r_1, r_2, \ldots, r_k$ over an alphabet Σ, an integer λ.

Question: Does there exist a string $s \in \Sigma^*$ of length at least λ, that is a subsequence[2] of each string in R?

The LCS and (not so much) the SCS problems have been extensively studied over the last 30 years (see [7] and references). They are both known to be NP-complete [8,9]. In particular the case where the number of sequences is 2 has been studied in detail (see [7] and references).

1.1. Sequence comparison in bioinformatics

With the recent availability of large amounts of molecular sequence data, the LCS and related problems received much attention due to the importance of *sequence comparison* problems in bioinformatics.

> (from [6]) Sequence comparison is used by biologists for several reasons. Similarity between a set of molecular sequences may indicate significant attributes between organisms the sequences represent. Or, from the similarity and disparity of these sequences, it may be possible to infer phylogenetic relationships amongst the organisms. If the sequences represent homologous genes, the comparison may be used to obtain information on molecular structure or function.
>
> …
>
> Computationally, multiple sequence comparison problems are viewed as string matching problems.

However, due to the $O(n^k)$ time requirements [1,5,7] of the best known algorithms for these analyses, the number of sequences that can be examined at once is often limited to less than six.

1.2. Parameterized complexity

The problems were also studied in the framework of *parameterized complexity* (see [4] for a survey). The hope was to find algorithms that have running times exponential in only some parameters of the problem. In [2,3,6] several parametrizations of the SCS and LCS problem were analyzed.

Parameterized shortest common supersequence (SCS)

Instance: Alphabet Σ, set of strings $R = r_1, r_2, \ldots, r_k \in \Sigma^*$, integer λ.

Parameter: k (SCS-1).

[1] A string a is a supersequence of a string b if we can delete some characters in a such that the remaining string is equal to b, e.g. "1234" is a supersequence of "13".

[2] A string a is a subsequence of a string b if b is a supersequence of a, e.g. "13" is a subsequence of "1234".

Parameter	Alphabet Size $\|\Sigma\|$		
	Unbounded	Parameter	Constant
k	LCS-1 W[t]-hard for $t \geq 1$ [3]	LCS-4 W[t]-hard for $t \geq 1$ [2]	FLCS $W[1]$-hard (below)
λ	LCS-2 W[2]-hard [3]	FPT	FPT
k, λ	LCS-3 W[1]-complete [3]	FPT	FPT

Fig. 1. The fixed parameter complexity of the LCS problem.

Parameter: λ (SCS-2).
Parameter: k, λ (SCS-3).
Parameter: k, $|\Sigma|$ (SCS-4).
Question: Does there exist a string $s \in \Sigma^*$ of length at most λ, that is a supersequence of each string in R?

Parameterized longest common subsequence (LCS)
Instance: Alphabet Σ, set of strings $R = r_1, r_2, \ldots, r_k \in \Sigma^*$, integer λ.
Parameter: k (LCS-1).
Parameter: λ (LCS-2).
Parameter: k, λ (LCS-3).
Parameter: k, $|\Sigma|$ (LCS-4).
Question: Does there exist a string $s \in \Sigma^*$ of length at least λ, that is a subsequence of each string in R?

The complexity of the parameterized variants of the LCS problem[3] are shown in Fig. 1. Note that all variants become fixed parameter tractable as soon as λ and Σ are bounded (i.e. they are either a parameter or constant), this is by the trivial algorithm that generates all $|\Sigma|^{\lambda}$ possible subsequence strings and checks them against each r_i. Similar results are known for the SCS problem (see [6]). One gets another interesting parameterization if the size of the alphabet Σ is a fixed constant.

Fixed alphabet shortest common supersequence parameterized in the number of strings (FSCS)
Instance: A fixed size alphabet Σ, set of strings $R = r_1, r_2, \ldots, r_k \in \Sigma^*$, integer λ.
Parameter: k.
Question: Does there exist a string $s \in \Sigma^*$ of length at most λ that is a supersequence of each string in R?

Fixed alphabet longest common subsequence parameterized in the number of strings (FLCS)
Instance: A fixed size alphabet Σ, set of strings $R = r_1, r_2, \ldots, r_k \in \Sigma^*$, integer λ.

[3] FLCS is named LCS-5 in [3].

Parameter: k.

Question: Does there exist a string $s \in \Sigma^*$ of length at least λ that is a subsequence of each string in R?

(from [2]) The most compelling of these problems (LCS-1, LCS-2, LCS-3 and FLCS) is FLCS, since the alphabet for biological sequences is often of fixed constant size, e.g. DNA and protein sequences have alphabets of size 4 and 20, respectively.

…

Our failure to find a hardness result for FLCS invites hope that it could be fixed-parameter tractable.

We will show that this is unfortunately not the case, namely that

Theorem 1. *The* fixed alphabet shortest common supersequence *problem parameterized in the number of strings is* $W[1]$ *hard.*

and

Theorem 2. *The* fixed alphabet longest common subsequence *problem parameterized in the number of strings is* $W[1]$ *hard.*

We will prove Theorems 1 and 2 by first showing the $W[1]$ completeness for a problem we call Partitioned Clique. Then a parameterized reduction from Partitioned Clique to FSCS is shown; finally we show how this reduction can be changed to get a reduction to FLCS instead.

2. $W[1]$ completeness for Partitioned Clique

The Clique and the Partitioned Clique problem are defined as follows.

Clique

Instance: A simple graph $G = (V, E)$, and integer k.

Parameter: k.

Question: Is there a subset $V' \subseteq V$ of cardinality k such that $\forall u, v \in V'$, $(u, v) \in E$?

Partitioned Clique (pClique)

Instance: A simple graph $G = (V, E)$, an integer k, a partition $\{U_1, \ldots, U_k\}$ of V into k sets of equal size.[4]

Parameter: k.

Question: Is there a subset $V' \subseteq V$ of cardinality k such that $\forall u, v \in V'$, $(u, v) \in E$ and $\forall i \in \{1, \ldots, k\}$: $|V' \cap U_i| = 1$?

[4] Note that $|V|$ must be a multiple of k.

Theorem 3. Partitioned Clique *is* $W[1]$ *complete.*

We first show $W[1]$-hardness of pClique reducing to it the $W[1]$-complete problem Clique (see e.g. [4]). Then we show that pClique is in $W[1]$ by a reduction in the other direction (for the following we actually only need $W[1]$ hardness, we will prove completeness anyway because it is easy).

Theorem 4. Partitioned Clique *is* $W[1]$ *hard.*

Proof. Given an instance $(G = (V = \{v_1, \ldots, v_n\}, E), k)$ of the Clique problem we construct an instance $(G' = (V', E'), k, \{U'_1, \ldots, U'_k\})$ for the pClique problem as follows. Every set $U'_j = \{u_1^j, \ldots, u_n^j\}$ consists of $n = |V|$ vertices, we will say that a vertex $u_i^j \in U'_j$ corresponds to vertex $v_i \in V$. There is an edge $(u_x^i, u_y^j) \in E'$ iff $(v_x, v_y) \in E$. To see that this a correct reduction we must show that

$$(G = (V, E), k) \in \text{Clique} \Leftrightarrow (G' = (V', E'), k, \{U_1, \ldots, U_k\}) \in \text{pClique}$$

$\Rightarrow$ If G has a Clique C of size k, we can assign every vertex from C to the corresponding vertex in a different set U'_i. By the construction, these vertices form a pClique in G'.

$\Leftarrow$ If we are given a pClique C', then all vertices in C' correspond to a different vertices in G (two vertices that correspond to the same vertex in G are not adjacent in G'), and those vertices build a Clique in G by the construction. □

Theorem 5. Partitioned Clique *is in* $W[1]$.

Proof. Given an instance $(G', V', \{U'_1, \ldots, U'_k\})$ for pClique, construct an instance (G, V) for Clique by removing all edges from G' between vertices that are in the same set U'_i. This does not change the size of pClique. Now every Clique of G is also a pClique in G'. □

3. $W[1]$ hardness for fixed alphabet shortest common supersequence

In this section we show a reduction from an instance of pClique to an instance of FSCS.

3.1. Notation and definitions

We reduce from the instance

$$\mathcal{G} = (G = (V, E), k, \{U_1, \ldots, U_k\}) \tag{1}$$

of pClique to the instance

$$\mathscr{S} = (S = \{s_1, \ldots, s_k, s_t\}, \lambda) \tag{2}$$

of FSCS.

The *fixed size* alphabet Σ for FSCS is the binary alphabet $\{0, 1\}$. We define n and m as

$$n = |V|, \tag{3}$$

$$m = |U_i| = \frac{n}{k}. \tag{4}$$

We denote the vertices V by $\{v_1, v_2, \ldots, v_n\}$. W.l.o.g. we assume that the set U_1 contains the first m vertices $\{v_1, \ldots, v_m\}$, U_2 the next m vertices $\{v_{m+1}, \ldots, v_{2m}\}$ and so on. We will also write v_j^i for $v_{(i-1)m+j}$, the jth vertex in the set U_i and v^i to denote any vertex in U_i. We write $q[i]$ for the ith character in a string q and $q[i \ldots j]$ for the substring[5] of q starting at the ith and ending at the jth character.

Definition 1 (Alignment, optimal alignment). Let s be a supersequence for a set of strings S. An *alignment* of S for s is a map $\phi : (q \in \{S \cup s\}, i \in \mathbb{N}) \to j \in \mathbb{N}$. It assigns to the i'th character in a string $q \in \{S \cup s\}$[6] an index $j = \phi(q, i) : 1 \leqslant j \leqslant |s|$ such that $q[i] = s[j]$ and $\forall i : \phi(q, i) < \phi(q, i+1)$. We say that two characters $q[i]$ and $q'[i']$ from two different strings $\{q, q'\} \in \{S \cup s\}$ align if $\phi(q, i) = \phi(q', i')$. We say that an alignment is *optimal* if every supersequence for S has length at least $|s|$.

Definition 2 (Map, overlap). Given an alignment ϕ of S for s, we say that a substring q' of $q \in \{S \cup s\}$ *maps* on a substring r' of $r \in \{S \cup s\}$ if $\forall i \exists j : \phi(q'[i]) = \phi(r'[j])$. Note that if q' maps on r', q' must be a subsequence of r' and that every q' maps on s. We say that q' *overlaps* with r' if $\exists i \exists j : \phi(q'[i]) = \phi(r'[j])$.

We give a small example for the above definitions. Let $s = 010011$ and $S = \{a = 1011, b = 101, c = 001\}$. Let ϕ be the alignment of S for s shown below ($\phi(a, 1) = 2, \phi(a, 2) = 4, \ldots$).

s	=	0	1	0	0	1	1
a	=		1		0	1	1
b	=		1		0	1	
c	=			0	0	1	

Here b maps on a. c overlaps with, but does not map on a. The substring $c[2 \ldots 3]$ of c maps on a. $b[2]$ aligns with $c[2]$.

[5] A string a is a substring of a string b if $b = uav$ for some strings u and v, e.g. "34" is substring of "1234".

[6] If $q = s$ we simply have $\phi(s, i) = i$, the definition of ϕ for s is redundant, we do it as to avoid a special treatment for s in the definition of "align" and "map" below.

3.2. The FSCS instance

We define the following strings from which we will construct our instance for FSCS ($\prod_{j=1}^{n} a_j$ denotes the concatenation of the strings $a_1 \ldots a_n$)

$$\mathscr{I} = 1^{7n^3}, \tag{5}$$

$$\mathscr{O} = 0^{7n^3}, \tag{6}$$

$$\mathscr{E}(u \in V, v \in V) = \begin{cases} \mathscr{I}\mathscr{I} & \text{if } (u = v) \text{or } (u \in U_i, v \in U_j) \in E : i \neq j, \\ \mathscr{I}0\mathscr{I} & \text{otherwise,} \end{cases} \tag{7}$$

$$\mathscr{V}(u \in V) = \prod_{j=1}^{n} \mathscr{E}(u, v_j), \tag{8}$$

$$\mathscr{B}_i = \mathscr{V}(v_1^i) \prod_{j=2}^{m} \mathscr{O}\mathscr{V}(v_j^i), \tag{9}$$

$$\mathscr{T}_{\mathscr{I}0\mathscr{I}} = (\mathscr{I}0\mathscr{I})^n, \tag{10}$$

$$\mathscr{T}_{\mathscr{I}\mathscr{I}} = (\mathscr{I}\mathscr{I})^n, \tag{11}$$

$$\mathscr{T} = (\mathscr{T}_{\mathscr{I}0\mathscr{I}}\mathscr{O})^{m-1}\mathscr{T}_{\mathscr{I}\mathscr{I}}. \tag{12}$$

The Instance $\mathscr{S} = \{s_1, \ldots, s_k, s_t, \lambda\}$ for FSCS is[7]

$$s_i = (\mathscr{B}_i\mathscr{O})^{2n+2n^2}\mathscr{B}_i,$$

$$s_t = (\mathscr{T}\mathscr{O})^{1+2n+2n^2}(\mathscr{T}_{\mathscr{I}0\mathscr{I}}\mathscr{O})^{m-2}\mathscr{T}_{\mathscr{I}0\mathscr{I}}.$$

3.3. Outline of the proof

Let s_{opt} be a shortest common supersequence for S.[8] In order to prove our main Theorem 1 we will prove the following Lemmas:

Lemma 1. *If $\mathscr{G} \in$ pClique[9] then there is a string s_λ of length λ that is a supersequence of all strings in S (or equivalently $\mathscr{S} \in$ FSCS).*[10]

Lemma 2. *If $\mathscr{G} \notin$ pClique then $|s_{\text{opt}}| > \lambda$ (or equivalently $\mathscr{S} \notin$ FSCS).*

[7] A concrete example will be given in Section 3.5.

[8] More precisely, let s_{opt} be any string that is a supersequence of all strings in S and has minimal possible length.

[9] $\mathscr{G} \in$ pClique means that G has a pClique of size k and $\mathscr{S} \in$ FSCS means that S has a shortest common supersequence of length at most λ.

[10] s_λ is not necessarily a *shortest* supersequence for S.

We will first prove Lemma 1 by showing how to construct a sequence s_λ of length λ that is a supersequence of all strings in S under the assumption that $\mathscr{G} \in \text{pClique}$. s_λ is similar to s_t, only the $1+2n+2n^2$ occurrences of a $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ substring in s_t are replaced by a substring $\mathscr{M}$ in s_λ. This $\mathscr{M}$ differs from $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ by $n-k$ additional 0's. We then have $|s_\lambda| = |s_t| + (1+2n+2n^2)(n-k) = \lambda$. To prove Lemma 2 we first show that in an optimal alignment at least $(1+2n^2)$ occurrences of a $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ substring in s_t map, for all $i : 1 \leqslant i \leqslant k$, to a $\mathscr{V}(v^i) : v^i \in U_i$ substring from s_i, we will call these $\mathscr{T}_{\mathscr{I}\mathscr{I}}$'s nice. Because we cannot choose those v^i such that they form a pClique, we show that, for every nice $\mathscr{T}_{\mathscr{I}\mathscr{I}}$, we have at least $n-k+1$ 0's in s_{opt} that do not align with a character in s_t. This gives a lower bound $|s_t| + (1+2n^2)(n-k+1) > \lambda$ for $|s_{\text{opt}}|$ which is bigger than λ.

3.4. Indices

In this section we will prove a simple claim (Claim 3) that will be useful for the following sections.

For a string $\mathscr{P}$ of the form[11]

$$\mathscr{P} = \prod_{j=1}^{n} (\mathscr{I}X\mathscr{I})_j : (\mathscr{I}X\mathscr{I})_j \in \{\mathscr{I}\mathscr{I}, \mathscr{I}0\mathscr{I}\} \tag{13}$$

we will say that $\mathscr{P}$ has a 0 at index j if $(\mathscr{I}X\mathscr{I})_j = \mathscr{I}0\mathscr{I}$. Let $C = \{v^1, v^2, \ldots, v^k\} : v^i \in U_i$ and let $J_C = \{j_1, j_2, \ldots, j_k\}$ be the indices of the vertices in C (this is to say $v^i = v_{j_i}$). Let J_C^0 be the indices where *no* string $\mathscr{V}(v^i) : v^i \in C$ has a 0.[12]

Claim 1. $J_C^0 \subseteq J_C$.

Proof. Let $j \notin J_C$, we will show that then $j \notin J_C^0$. For some i, v_j is in the same set U_i as $v^i \in C$. $v^i \neq v_j$ because the index j_i is in J_C but j is not. Now $\mathscr{E}(v^i, v_j) = \mathscr{I}0\mathscr{I}$ (7,8) and so $\mathscr{V}(v^i)$ has a 0 at index j. □

Claim 2. *C is a* pClique *if and only if* $J_C \equiv J_C^0$.

Proof. *Case* 1: Assume C is a pClique. For any $v^i \in C$ and any $j \in J_C$ we either have $v^i = v_j$ or there is an edge between v^i and v_j. In both cases $\mathscr{E}(v^i, v_j) = \mathscr{I}\mathscr{I}$ so $\mathscr{V}(v^i)$ has no 0 at index j (7,8). We have shown that $J_C \subseteq J_C^0$ and with $J_C^0 \subseteq J_C$ from the previous claim we have $J_C \equiv J_C^0$.

Case 2: Assume C is not a pClique. Then there are two nonadjacent vertices $\{v^h, v^i\} \in C$. Now $\mathscr{E}(v^h, v^i) = \mathscr{I}0\mathscr{I}$ (7,8) and $\mathscr{V}(v^h)$ has a 0 at index j_i. So $j_i \notin J_C^0$, but $j_i \in J_C$ from which $J_C \not\equiv J_C^0$ follows. □

Claim 3. *If C is a* pClique $|J_C^0| = k$ *otherwise* $|J_C^0| < k$.

[11] e.g. $\mathscr{V}_i, \mathscr{T}_{\mathscr{I}\mathscr{I}}$ and $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ are of this form.

[12] Or more formally $\forall(j \in J_C^0)\forall(v^i \in C) : \mathscr{V}(v^i)$ has no 0 at index j.

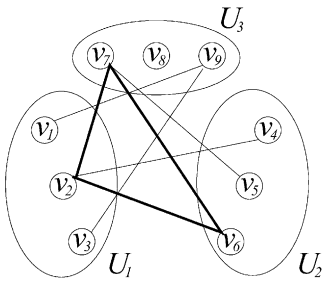

Fig. 2. An instance $(G, k = 3, \{U_1 = \{v_1, v_2, v_3\}, U_2 = \{v_4, v_5, v_6\}, U_3 = \{v_7, v_8, v_9\}\})$ for pClique. The vertices $\{v_2, v_6, v_7\}$ form a pClique of size 3.

Proof. The claim trivially follows from the two claims above. □

3.5. Proof of Lemma 1

For this section we will assume that $\mathscr{G} \in$ pClique. Let $C = \{v^1, v^2, \ldots, v^k\} : v^i \in U_i$ be a pClique. In Fig. 2 we have an example where $C = \{v_2, v_6, v_7\}$. Let $J_C = \{j_1, j_2, \ldots, j_k\}$ be the indices of the vertices in C (for our example $J_C = \{2, 6, 7\}$). Let s_λ be the string s_t with the only distinction that all $1 + 2n + 2n^2$ occurrences of a $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ substring are replaced by the substring $\mathscr{M}$ (defined below).

$$\mathscr{M} = \prod_{j=1}^{n} \begin{cases} \mathscr{I}\mathscr{I} & \text{if } j \in J_C, \\ \mathscr{I}0\mathscr{I} & \text{otherwise.} \end{cases} \tag{14}$$

Claim 4. *For all $v^i \in C$, $\mathscr{M}$ is a supersequence of $\mathscr{V}(v^i)$.*

Proof. $\mathscr{M}$ and $\mathscr{V}(v^i)$ are both of type (13). $\mathscr{M}$ has a 0 at all indices except J_C, but at the indices J_C also no $\mathscr{V}(v^i)$ has a 0 (Claim 3). The claim now follows from the observation that $\mathscr{I}0\mathscr{I}$ is a supersequence of $\mathscr{I}\mathscr{I}$. □

Claim 5. *s_λ is a supersequence of all strings in $S = \{s_1, \ldots, s_k, s_t\}$.*

Proof. Because $\mathscr{M}$ is a supersequence of $\mathscr{T}_{\mathscr{I}\mathscr{I}}$, s_λ is a supersequence of s_t. We now must show that for all i: $1 \leqslant i \leqslant k$, s_λ is a supersequence of s_i. We can map s_i on s_λ[13] such that (see example below)

1. Every $\mathscr{V}(v^i)$ substring in s_i maps on a $\mathscr{M}$ substring in s_t (see Claim 4).
2. Every $\mathscr{V}(v)$: $v \neq v^i$ substring in s_i maps on a $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ substring in s_t. (Note that for every vertex $v \in V$, $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ is a supersequence of $\mathscr{V}(v)$.)
3. Every $\mathscr{O}$ substring in s_i maps on a $\mathscr{O}$ substring in s_λ.

[13] If we can map s_i on s_λ then s_λ must be a supersequence of s_i.

The reader may convince himself that by the definition of s_λ and s_i this can always be done. An example is given below. □

In Fig. 2 an instance for the pClique problem is given. An alignment of S for s_λ appears as follows:[14]

$$
\begin{array}{lcccccccccccccccccc}
s_t & = & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & (\mathscr{T}_{\mathscr{I}\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O})^{2n^2+2n} & \mathscr{T}_{\mathscr{I}\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} \\
s_1 & = & & & \mathscr{V}(v_1) & \mathscr{O} & (\mathscr{V}(v_2) & \mathscr{O} & \mathscr{V}(v_3) & \mathscr{O} & \mathscr{V}(v_1) & \mathscr{O})^{2n^2+2n} & \mathscr{V}(v_2) & \mathscr{O} & \mathscr{V}(v_3) & & \\
s_2 & = & \mathscr{V}(v_4) & \mathscr{O} & \mathscr{V}(v_5) & \mathscr{O} & (\mathscr{V}(v_6) & \mathscr{O} & \mathscr{V}(v_4) & \mathscr{O} & \mathscr{V}(v_5) & \mathscr{O})^{2n^2+2n} & \mathscr{V}(v_6) & & & & \\
s_3 & = & & & & & (\mathscr{V}(v_7) & \mathscr{O} & \mathscr{V}(v_8) & \mathscr{O} & \mathscr{V}(v_9) & \mathscr{O})^{2n^2+2n} & \mathscr{V}(v_7) & \mathscr{O} & \mathscr{V}(v_8) & \mathscr{O} & \mathscr{V}(v_9) \\
s_\lambda & = & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & (\mathscr{M} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O})^{2n^2+2n} & \mathscr{M} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & \mathscr{O} & \mathscr{T}_{\mathscr{I}0\mathscr{I}}
\end{array}
$$

The alignment of $\mathscr{T}_{\mathscr{I}\mathscr{I}}$, the $\mathscr{V}(v^i)$'s and $\mathscr{M}$ appears as follows:

$$
\begin{array}{lcccccccccc}
\mathscr{T}_{\mathscr{I}\mathscr{I}} & = & \mathscr{I}\ \mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\ \mathscr{I} & \mathscr{I}\ \mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\ \mathscr{I} & \mathscr{I}\ \mathscr{I} \\
\mathscr{V}(v_2) & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\ \mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} \\
\mathscr{V}(v_6) & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} \\
\mathscr{V}(v_7) & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\ \mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} \\
\mathscr{M} & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I}
\end{array}
$$

Proof of Lemma 1. Lemma 1 follows from Claim 5, we only have to check if s_λ has indeed length $\leqslant\lambda$. s_λ differs from s_t by $n-k$ additional 0's in every of the $1+2n+2n^2$ $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ substrings from s_t[15] and so has length $|s_t|+(1+2n+2n^2)(n-k)=\lambda$. □

3.6. Proof of Lemma 2

In this section we prove that if $\mathscr{G}$ does not have a pClique of size k, then the shortest common supersequence s_{opt} of the instance FSCS constructed from $\mathscr{G}$ must have length larger than λ.

All claims in this section relate to an optimal alignment. To save on notation, in the sequel we will write

$\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ for either a $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ or a $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ substring from s_t,

$\mathscr{V}_i$ for a $\mathscr{V}(v^i)$ substring from s_i.

Observation 1. By replacing every $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ with $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ in s_t, we get a string s_{up} that is a supersequence for every string in S. So $|s_{\text{up}}|$ is an upper bound for $|s_{\text{opt}}|$. We will need the following inequalities: $|s_{\text{up}}|=|s_t|+(1+2n+2n^2)n<|s_t|+7n^3-n=|s_t|+|\mathscr{O}|-n<|s_t|+|\mathscr{I}|$

[14] The alignment is not unique.

[15] A 0 at all n indices except at the k indices J_C.

Observation 2. s_{opt} has less than $|\mathscr{I}|$ more 1's than s_t, otherwise $|s_{\text{opt}}| \geqslant |s_t| + |\mathscr{I}| > |s_{\text{up}}|$.[16]

Observation 3. s_{opt} has less than $|\mathscr{O}| - n$ more 0's than s_t, otherwise $|s_{\text{opt}}| \geqslant |s_t| + |\mathscr{O}| - n > |s_{\text{up}}|$.

Claim 6. $\forall i$: *No* $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ *may overlap with two different* $\mathscr{V}_i$'s.

Proof. Note that two $\mathscr{V}_i$'s are separated by at least one $\mathscr{O}$ block and $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ has either any or n 0's. If a $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ is aligned to two different $\mathscr{V}_i$'s at most n 0's from $\mathscr{O}$ can align with a 0 from s_t, and we have $|s_{\text{opt}}| \geqslant |s_t| + |\mathscr{O}| - n$ contradicting Observation 3. □

Claim 7. $\forall i$: *Every* $\mathscr{V}_i$ *overlaps with at least one* $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$.

Proof. If not, the $2|\mathscr{I}|$ 1's from that $\mathscr{V}_i$ are not aligned to any 1 from s_t (all 1's in s_t are in the $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s). Then s_{opt} has at least $2|\mathscr{I}|$ more 1's than s_t contradicting Observation 2. □

Claim 8. $\forall i$: *At most* $m-1$ $\mathscr{V}_i$*'s overlap with more than one* $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$.

Proof. Suppose more than $m-1$ $\mathscr{V}_i$'s do overlap with more than one $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$, with Claim 7 and the observation that there are only $m-1$ more $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s than there are $\mathscr{V}_i$'s we must have that at least one $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ overlaps with two different $\mathscr{V}_i$'s contradicting Claim 6. □

Definition 3 (Nice $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$, good $\mathscr{V}_i$). We call a $\mathscr{V}_i$ *good* if it overlaps with exactly one $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$.

We call a $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ *nice*, if, for all i: $1 \leqslant i \leqslant k$, it overlaps with exactly one $\mathscr{V}_i$, and this $\mathscr{V}_i$ is good (overlaps with no other $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$).

Claim 9. *There are at least* $(1+2n^2)$ *nice* $\mathscr{T}_{\mathscr{I}\mathscr{I}}$*'s.*

Proof. For any i we have at most $(m-1)$ $\mathscr{V}_i$'s that overlap with more than one $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ (Claim 8), all other $\mathscr{V}_i$'s overlap with exactly one $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ (Claim 7). With this and the observation that there are $m-1$ more $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s than there are $\mathscr{V}_i$'s, we see that at most $2(m-1)$ $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s can fail to overlap with exactly one *good* $\mathscr{V}_i$. If we sum over all i's $(1 \leqslant i \leqslant k)$ we get that at most $2(m-1)k$ $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s can fail to overlap with exactly one good $\mathscr{V}_i$ for all i, or equivalently, can fail to be nice. Because the $\mathscr{T}_{\mathscr{I}\mathscr{I}}$'s are a subset of the $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s, we also have that at most $2(m-1)k < 2n$ $\mathscr{T}_{\mathscr{I}\mathscr{I}}$'s are not nice, and so at least $(1+2n+2n^2) - 2n$ $\mathscr{T}_{\mathscr{I}\mathscr{I}}$'s are nice. □

Let $\mathscr{A} = \{\mathscr{T}'_{\mathscr{I}\mathscr{I}}, \mathscr{V}'_1, \ldots, \mathscr{V}'_k\}$ where $\mathscr{T}'_{\mathscr{I}\mathscr{I}}$ is nice and it overlaps with $\mathscr{V}'_i$ for all i.

Claim 10. $\forall i,j$: *A* 0 *from* $\mathscr{V}'_i$ *never aligns with a* 0 *from* $\mathscr{V}'_j$ *if the two* 0*'s do not have the same index.*

[16] This follows from the simple observation that if s_{opt} has $|\mathscr{I}|$ more 1's than s_t, at least $|\mathscr{I}|$ 1's from s_{opt} cannot align with a character in s_t.

Proof. Assume we have a 0 in $\mathscr{V}'_i$ and a 0 in $\mathscr{V}'_j$ with different indices that align. This implies that at least $2|\mathscr{I}|$ 1's from $\mathscr{V}'_i$ cannot align with 1's from $\mathscr{V}'_j$ and so q has at least $2|\mathscr{I}|$ more 1's than $|\mathscr{T}_{\mathscr{I}\mathscr{I}}|$. This is best seen by a small example. Assume we align a 0 from $\mathscr{V}(v_2)$ at index 3 with a 0 from $\mathscr{V}(v_9)$ at index 2.

$$\begin{array}{lcccccccccccc} \mathscr{V}(v_2) & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathbf{0}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} \\ \mathscr{V}(v_9) & = & & \mathscr{I}\mathscr{I} & \mathscr{I}\mathbf{0}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} \end{array}$$

Then any supersequence q of $\mathscr{V}(v_2)$ and $\mathscr{V}(v_9)$ must have $2|\mathscr{I}|$ more 1's than $|\mathscr{T}_{\mathscr{I}\mathscr{I}}|$ e.g.

$$\begin{array}{lcccccccccccc} q & = & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}\mathbf{0}\mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}0\mathscr{I} & \mathscr{I}\mathscr{I} \\ T_{\mathscr{I}\mathscr{I}} & = & \mathscr{I} & \mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I}\mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} & \mathscr{I} \end{array}$$

Now $\mathscr{V}'_i$ and $\mathscr{V}'_j$ may not overlap with any $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ other than $\mathscr{T}'_{\mathscr{I}\mathscr{I}}$ (because $\mathscr{T}'_{\mathscr{I}\mathscr{I}}$ is nice), but 1's in s_t appear only in $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s, so we get at least $2|\mathscr{I}|$ more 1's in s_{opt} than in s_t contradicting Observation 2. □

Claim 11. *A* 0 *from a* $\mathscr{V}'_i \in \mathscr{A}$ *does not align with a* 0 *that is not from a string in* $\mathscr{A}$.

Proof. When a 0 from $\mathscr{V}'_i$ aligns with a 0 from s_j that is not in $\mathscr{V}'_j$, this implies that there are at least $|\mathscr{I}|$ more 1's in s_{opt} than in s_t (by an argument similar to that in the previous claim) contradicting Observation 2. □

Proof of Lemma 2. Let s_{opt} be a shortest common supersequence of S, where S is constructed from an instance $(G, k, \{U_1, \ldots, U_k\})$ which has *no* pClique of size k. Let $\mathscr{T}'_{\mathscr{I}\mathscr{I}}$ be any nice $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ that overlaps with $\{\mathscr{V}'(v^1), \ldots, \mathscr{V}'(v^k)\}$: $v^i \in U_i$. A 0 from a $\mathscr{V}'(v^i)$ may align only with 0's from $\mathscr{V}'(v^j)$ with the same index (Claims 10 and 11). Because $\{v^1, \ldots, v^k\}$ cannot be a pClique, we have fewer than k indices where no $\mathscr{V}'(v^i)$ has a 0 (Claim 3), and so s_{opt} has at least $n-k+1$ more 0's than s_t. This is the case for all of the at least $1+2n^2$ nice $\mathscr{T}_{\mathscr{I}\mathscr{I}}$'s (Claim 9). We now get a lower bound for s_{opt} that is bigger than λ.

$|s_{\text{opt}}| > |s_t| + (1+2n^2)(n-k+1) > |s_t| + (1+2n+2n^2)(n-k) = \lambda$. □

Proof of Theorem 1. Theorem 1 follows from Lemmas 1 and 2. Note that the reduction is a parameterized many-one reduction as required for the result (see [4] for any details). □

4. $W[1]$ hardness for fixed alphabet longest common subsequence

The reduction from pClique to FLCS is very similar to the reduction we constructed for FSCS. We will only sketch it.

The definitions of *alignment*, *map*, *overlap* and *nice* can be redefined for an alignment of a subsequence in a natural way.

For any string $\mathscr{X}$, let $\overline{\mathscr{X}}$ denote the string we get when we replace all occurrences of $\mathscr{I}\mathscr{I}$ in $\mathscr{X}$ by $\mathscr{I}0\mathscr{I}$ and vice versa.

The instance $\mathscr{L} = (L = \{l_1, \ldots, l_k, l_t\}, \gamma)$ for FLCS is

$$l_i = (\overline{\mathscr{B}_i}\mathscr{O})^{2n+2n^2}\overline{\mathscr{B}_i},$$

$$l_t = \mathscr{T}_{\mathscr{I}0\mathscr{I}}(\mathscr{O}\overline{\mathscr{T}})^{2n+2n^2},$$

$$\gamma = |l_t| - (1 + 2n + 2n^2)(n - k).$$

Lemma 3. *If* $\mathscr{G} \in$ pClique *then there is a string* l_γ *of length* γ *that is a subsequence of all strings in* L (*or equivalently* $\mathscr{L} \in$ FSCS).

Proof (sketch). The proof is very similar to the proof of the Lemma 1. Let l_γ be the string l_t where all $(1 + 2n + 2n^2)$ occurrences of a $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ substring are replaced by the substring $\overline{\mathscr{M}}$ (see 14). l_γ is a subsequence of all strings in L (proof omitted but similar to proof of Claim 5). For the example from Section 3.5 an alignment appears as follows:

$$\begin{array}{lllllllllllllll}
l_t & = & & & & & (\mathscr{T}_{\mathscr{I}0\mathscr{I}}\ \mathscr{O}\ \mathscr{T}_{\mathscr{I}\mathscr{I}}\ \mathscr{O}\ \mathscr{T}_{\mathscr{I}\mathscr{I}}\ \mathscr{O})^{2n^2+2n} & & \mathscr{T}_{\mathscr{I}0\mathscr{I}} & & & & \\
l_1 & = & & \overline{\mathscr{V}(v_1)} & \mathscr{O} & (\overline{\mathscr{V}(v_2)}\ \mathscr{O}\overline{\mathscr{V}(v_3)}\ \mathscr{O}\overline{\mathscr{V}(v_1)}\ \mathscr{O})^{2n^2+2n} & & \overline{\mathscr{V}(v_2)} & \mathscr{O} & \overline{\mathscr{V}(v_3)} & & \\
l_2 & = & \overline{\mathscr{V}(v_4)} & \mathscr{O} & \overline{\mathscr{V}(v_5)} & \mathscr{O} & (\overline{\mathscr{V}(v_6)}\ \mathscr{O}\ \overline{\mathscr{V}(v_4)}\ \mathscr{O}\overline{\mathscr{V}(v_5)}\ \mathscr{O})^{2n^2+2n} & & \overline{\mathscr{V}(v_6)} & & & & \\
l_3 & = & & & & & (\overline{\mathscr{V}(v_7)}\ \mathscr{O}\ \overline{\mathscr{V}(v_8)}\ \mathscr{O}\ \overline{\mathscr{V}(v_9)}\ \mathscr{O})^{2n^2+2n} & & \overline{\mathscr{V}(v_7)} & \mathscr{O} & \overline{\mathscr{V}(v_8)} & \mathscr{O} & \overline{\mathscr{V}(v_9)} \\
l_\lambda & = & & & & & (\overline{\mathscr{M}}\mathscr{O}\ \mathscr{T}_{\mathscr{I}\mathscr{I}}\ \mathscr{O}\ \mathscr{T}_{\mathscr{I}\mathscr{I}}\ \mathscr{O})^{2n^2+2n} & & \overline{\mathscr{M}} & & & & \\
\end{array}$$

$\overline{\mathscr{M}}$ has length $|\mathscr{T}_{\mathscr{I}0\mathscr{I}}| - (n - k)$, and so l_γ is a subsequence of L of length γ. □

Let C be defined as in Section 3.4. Let $\overline{J_C^0}$ be the indices where *all* strings $\overline{\mathscr{V}(v^i)} : v^i \in C$ have a 0. We have $J_C^0 \equiv \overline{J_C^0}$[17] because by the definition the string $\overline{\mathscr{V}(v^i)}$ has a 0 at an index j if and only if $\mathscr{V}(v^i)$ has no 0 at index j. We can now restate Claim 3 by replacing J_C^0 with $\overline{J_C^0}$.

Claim 12. *If* C *is a* pClique $|\overline{J_C^0}| = k$ *otherwise* $|\overline{J_C^0}| < k$.

The proofs of Claims 13–18 below are omitted because they are similar to the proofs of Claims 6–11.[18] Note that Claims 13–15 are almost identical to the Claims 6–8, only the roles of $\overline{\mathscr{V}_i}$ (resp. $\mathscr{V}_i$) and $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ are interchanged. Claims 17 and 18 are identical to Claims 10 and 11, only the $\mathscr{V}$'s are replaced by $\overline{\mathscr{V}}$'s. Claim 16 is identical to Claim 9, only $\mathscr{T}_{\mathscr{I}\mathscr{I}}$ is replaced by $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$.

Claim 13. $\forall i$: *No* $\overline{\mathscr{V}_i}$ *may overlap with two different* $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s.

[17] J_C^0 is defined in Section 3.4.

[18] In the sequel we will only need Claims 16–18, Claims 13–15 are only stated as to ease the reproduction of the full proof.

Claim 14. $\forall i$: *Every* $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$ *overlaps with at least one* $\overline{\mathscr{V}}_i$.

Claim 15. $\forall i$: *At most* $m-1$ $\mathscr{T}_{\mathscr{I}X\mathscr{I}}$'s *overlap with more than one* $\overline{\mathscr{V}}_i$.

Claim 16. *There are at least* $(1+2n^2)$ *nice* $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$'s.

Let $\mathscr{A} = \{\mathscr{T}'_{\mathscr{I}0\mathscr{I}}, \overline{\mathscr{V}}'_1, \ldots, \overline{\mathscr{V}}'_k\}$ where $\mathscr{T}'_{\mathscr{I}0\mathscr{I}}$ is nice and it overlaps with $\overline{\mathscr{V}}'_i$ for all i.

Claim 17. $\forall i,j$: *A* 0 *from* $\overline{\mathscr{V}}'_i$ *never aligns with a* 0 *from* $\overline{\mathscr{V}}'_j$ *if the two* 0's *do not have the same index.*

Claim 18. *A* 0 *from a* $\overline{\mathscr{V}}'_i \in \mathscr{A}$ *does not align with a* 0 *that is not from a string in* $\mathscr{A}$.

Lemma 4. *If* $\mathscr{G} \notin$ pClique *then* $|l_{\text{opt}}| < \gamma$[19] (*or equivalently* $\mathscr{L} \notin$ FSCS).

Proof. The proof is very similar to the proof of Lemma 2. Let l_{opt} be a longest common subsequence of L, where L is constructed from an Instance $(G, k, \{U_1, \ldots, U_k\})$ which has *no* pClique of size k. Let $\mathscr{T}'_{\mathscr{I}0\mathscr{I}}$ be any nice $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$ that overlaps with $\{\overline{\mathscr{V}}'(v^1), \ldots, \overline{\mathscr{V}}'(v^k)\} : v^i \in U_i$. A 0 from a $\overline{\mathscr{V}}'(v^i)$ may align only with 0's from $\overline{\mathscr{V}}'(v^j)$ with the same index (Claims 17 and 18). Because $\{v^1, \ldots, v^k\}$ cannot be a pClique, we have fewer than k indices where all $\mathscr{V}'(v^i)$ have a 0 (Claim 12), and so l_{opt} has at least $n-k+1$ fewer 0's than l_t. This is the case for all of the at least $1+2n^2$ nice $\mathscr{T}_{\mathscr{I}0\mathscr{I}}$'s (Claim 16). We now get an upper bound for l_{opt} that is smaller than γ

$$|l_{\text{opt}}| < |l_t| - (1+2n^2)(n-k+1) < |l_t| - (1+2n+2n^2)(n-k) = \gamma \quad \square$$

Proof of Theorem 2. Theorem 2 follows from Lemmas 3 and 4. Note that the reduction is a parameterized many-one reduction as required for the result (see [4] for any details). $\square$

5. Conclusion

We have shown that the fixed alphabet shortest common supersequence (SCS) and the fixed alphabet longest common subsequence (LCS) problems parameterized in the number of strings are $W[1]$-hard.

Unless an unlikely collapse in the parameterized hierarchy occurs, this rules out the existence of exact algorithms with running time $f(k)n^{O(1)}$ (i.e., exponential only in k) for those problems. This does not mean that there are no algorithms with much better asymptotic time-complexity than the known $O(n^k)$ algorithms based on dynamic programming, e.g. algorithms with running time $n^{\sqrt{k}}$ are not deemed impossible by our results.

The exact classification of the FSCS and FLCS problems within the parameterized hierarchy is left open. We have evidence[20] that leads us to the conjecture that an exact classification of these

[19] l_{opt} is any string that is subsequence of all strings in L and has maximal possible length.

[20] Notes can be obtained from the author.

problems (and many others) within the parameterized hierarchy is not possible, because they seem to be not even in $W[P]$, the highest (reasonable) class these problems could be complete for.

Acknowledgments

This work is part of a diploma thesis at ETH Zürich, it was mainly written in the Computational Biology Lab at the McGill University in Montréal. I thank the following people: Prof. Gaston Gonnet, my supervisor at ETH; Prof. Mike Hallett, my supervisor at McGill, for introducing me to the theory of parameterized complexity, for many interesting discussions and comments and for his proofreading; Kaleigh Smith, with whom I shared an office at McGill, for proofreading, many interesting discussions (though mostly not on this topic) and for letting me stay at her place during my last two weeks in Montréal; Nicolas Dutil for proofreading; And last but not least the anonymous referees for their countless helpful comments and corrections.

References

[1] A.V. Aho, D.S. Hirschberg, J.D. Ullman, Bounds on the complexity of the longest common subsequence problem, J. Assoc. Comput. Mach. 23 (1) (1976) 1–12.
[2] H.L. Bodlaender, R.G. Downey, M.R. Fellows, M.T. Hallett, H.T. Wareham, Parameterized complexity analysis in computational biology, CABIOS 11 (1) (1995) 49–57.
[3] H.L. Bodlaender, R.G. Downey, M.R. Fellows, H.T. Wareham, The parameterized complexity of sequence alignment and consensus, Theoret. Comput. Sci. 147 (1994) 31–54.
[4] R.G. Downey, M.R. Fellows, Parameterized Complexity, Monographs in Computer Science, Springer, Berlin, 1999.
[5] K. Hakata, H. Imai, The longest common subsequence problem for small alphabet size between many strings. Proceedings of the third Annual International Symposium on Algorithms and Computation, Lecture Notes in Computer Science, Vol. 670, Springer, Berlin, 1992.
[6] M.T. Hallett, An integrated complexity analysis of problems from computational biology, Ph.D. Dissertation, University of Victoria, 1996.
[7] R.W. Irving, C.B. Fraser, Two algorithms for the longest common subsequence of three (or more) strings, Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching (CPM), Lecture Notes in Computer Science, Vol. 644, Springer, Berlin, 1992, pp. 214–229.
[8] D. Maier, The complexity of some problems on subsequences and supersequences, J. Assoc. Comput. Mach. 25 (2) (1978) 322–336.
[9] K.J. Räihä, E. Ukkonen, The shortest common supersequence problem over a binary alphabet is $\mathcal{NP}$-complete, Theoret. Comput. Sci. 16 (1981) 187–198.