On non-coding RNA
multiple structural alignment

Romeo Rizzi¹ and Stéphane Vialette²

¹ Dipartimento di Matematica ed Informatica (DIMI),
Università di Udine
Via delle Scienze 208, I-33100 Udine, Italy
Romeo.Rizzi@dimi.uniud.it

² Laboratoire de Recherche en Informatique (LRI), UMR CNRS 8623
Faculté des Sciences d’Orsay - Université Paris-Sud, 91405 Orsay, France
Stephane.Vialette@lri.fr

Abstract. In the context of non-coding RNA (ncRNA) multiple structural alignment, Davydov and Batzoglou introduced in [10] the problem of finding a largest nested linear graph that occurs in a set \( G \) of linear graphs, the so-called \textbf{Max-NLS} problem. This problem generalizes both the \textit{longest common subsequence} problem and the \textit{maximum common homeomorphic subtree} problem for rooted ordered trees. In the present paper, we give a fast algorithm for finding the largest nested linear subgraph of a linear graph and a polynomial-time algorithm for fixed \#\( G \). Also, we strongly strengthen the result of [10] by proving that the problem is \textbf{NP}-complete even if \( G \) is composed of nested linear graphs of height at most 2, thereby precisely defining the borderline between tractable and intractable instances of the problem. Of particular importance, we improve the result of [10] by showing that the \textbf{Max-NLS} problem is approximable within ratio \( \lceil \log k \rceil + 1 \), where \( k \) is the size of an optimal solution. Also, it is shown here that the \textbf{Max-NLS} problem can be approximated within ratio \( h \), where \( h \) if the maximum height of a linear graph in \( G \) and \( m \) is the maximum size of a linear graph in \( G \). From a parameterized complexity point of view, we show that the problem has complexity varying from \textbf{FPT} to \textbf{W}[t]-hard for all \( t \geq 1 \) depending on the parameter and the structure of \( G \). The emphasis here is on the combination of several parameters: nested or not, the height and the depth.

1 Introduction

Some genes produce RNA that are functional instead of coding for proteins [13]. These enigmatic genes are the genes for non-coding RNA (ncRNA). Genes for ncRNA produce transcripts that do not encode proteins but rather function directly as RNA. ncRNA are surprisingly numerous [13]. The best known ncRNA have complex three-dimensional RNA structures and play roles as catalytic or structural parts of RNA-protein machines [14]; transfer RNA, ribosomal RNA and spliceosomal RNA are textbook examples. The structural stability and function of ncRNA genes are largely determined by the formation of stable secondary structures through complementary bases (see [32] for a detailed introduction to RNA secondary structures). However, Rivas and Eddy show in [29] that the stability of ncRNA secondary structures is not sufficiently different from the predicted stability of random genomic fragments to yield a discernible statistical signal. One approach for comparing ncRNA is to align the sequences and predict a consensus secondary structure for the most conserved parts of the alignment [21, 33]. These methods however fail in the absence of significant sequence
similarity and reliable alignment. The algorithm presented in [26] takes as input a set of unaligned RNA sequences expected to share a common motif and outputs the regions that are most conserved throughout the sequences, according to a similarity measure that takes into account both the sequence and the secondary structure they can form according to base-pairing and thermodynamics rules. According to [10] (see also [30]), one of the most promising ways of detecting ncRNA genes and predicting reliable a secondary structure for them is comparative sequence analysis: RNA genes across different species are similar in the pattern of bases complementarity rather than in the genomic sequence, and hence conventional sequence alignment methods are not able to properly align ncRNA [14].

In this context, Davydov and Batzoglou proposed in [10] that a meaningful approach to ncRNA multiple alignment is finding the largest secondary structure common to all sequences, lining up the bases forming this structure, and then aligning corresponding leftover pieces as one would align genomic sequences which have no evolutionary pressure favoring complementary substitution. In [10], this is nicely modeled in terms of an optimization problem called Max-NLS (Maximum Nested Linear Graph) defined as follows. Let \( G = \{G_1, G_2, \ldots, G_n\} \) be the linear graphs derived from the ncRNA sequences \( S_1, S_2, \ldots, S_n \), respectively. The maximum common nested linear graph of these graphs is the largest nested linear graph \( G_{sol} \) such that \( G_{sol} \) is a subgraph of each \( G_i \in G \). For the Max-NLS problem, it has been shown in [10] that the problem is NP-complete (for unbounded \( \#G \)) and approximable within ratio \( O(\log^2 k) \), where \( k \) is the size of the optimal solution, i.e., \( k = \#E(G_{sol}) \). The Max-NLS problem shows strong similarities with the well-known Longest Common Subsequence problem. In case each linear graph \( G_i \in G \) is nested, the Max-NLS problem generalizes both the Maximum Common Homeomorphic Subtree problem for rooted ordered trees [23, 28, 7] and the Ordered Trees Alignment problem [22]. On the other hand, the Max-NLS problem can naturally be seen as a special case of the pattern matching problem over sets of 2-intervals for model \( \mathcal{R} = \{<, \sqsubset\} \) [31, 3, 19].

This paper is organized as follows. Section 2 presents some preliminaries. We give in Section 3 a fast and simple dynamic programming algorithm for finding a nested linear graph in a linear graph, and present in Section 4 a polynomial-time algorithm for fixed \( \#G \). Section 5 strongly refines the hardness result of [10] by giving a tight description of the borderline between NP-completeness and \( \mathbf{P} \). In Section 6, we improve the approximation ratio of [10] and present two simple approximation algorithms for the Max-NLS problem. Finally, we propose in Section 7 a fixed-parameter algorithm parameterized by the size of the optimal solution, together with fixed-parameter intractability results with strong emphasis on the combination of several parameters. Due to space constraints, several details and proofs are not presented in this paper.

## 2 Preliminaries

Basic familiarity with graph-theoretic terminology is assumed. For a graph \( G \), we denote by \( V(G) \) the set of vertices and \( E(G) \) the set of edges. The order and the size of \( G \) stand for \( \#V(G) \) and \( \#E(G) \), respectively. A linear graph of order \( n \) is a vertex-labeled graph where each vertex is labeled by a distinct label from \( \{1, 2, \ldots, n\} \). For the sake of clarity, we always denote the vertices of linear graphs by integers. In case of linear graphs, we write an edge between vertices \( i \) and \( j \), \( i < j \), as the pair \( (i, j) \). Two edges of a graph are called independent if they do not share a vertex. A linear graph \( G \) is called edge-independent if it is composed of independent edges, i.e., \( G \) is a matching.

Of particular interest are the relations between independent edges [31]. Let \( e = (i, j) \) and \( e' = (i', j') \) be two independent edges in a linear graph \( G \). We write \( (i) \ e < e' \) if \( i < j \) and \( i' < j' \), as
Let $G \wedge, h$.

A nested linear graph is called a (if it contains no branching edges. Rephrased in terms of Dyck words, a nested linear graph $G$ is flat if it can be written as

$$G = (a^h b^h) (a^h b^h) \ldots (a^h b^h)$$

for some positive integers $h_1, h_2, \ldots, h_k$. A nested linear graph is called a (w, h)-level graph if it can be written as $G = (a^h b^h)^w$ for some positive integers $h$ and $w$. We now define the notion of occurrence of one linear graph in another. Let $G_1$ and $G_2$ be two linear graphs. The graph $G_1$ is said to occur in $G_2$ if one can obtain $G_1$ from $G_2$ (regardless of precise vertex labels) by a sequence of edge and vertex deletions. The Max-NLS problem is defined formally as follows (see Fig.1 for an illustration).

**Max-NLS**

**Instance:** A family of $n$ linear graphs $G = \{G_1, G_2, \ldots, G_n\}$.

**Solution:** A nested linear graph $G_{sol}$ that occurs in each $G_i \in G$.

**Measure:** The size of the common nested linear graph $G_{sol}$, i.e., $|E(G_{sol})|$.

The Max-NLS problem - in its natural decision version - is NP-complete even if $G$ is composed of edge-independent linear graphs [10] and is approximable within ratio $O(\log^2 k)$, where $k$ is the size of the solution, i.e., $k = |E(G_{sol})|$. We briefly review some related results. Lozano and Valiente gave a dynamic programming algorithm for finding a largest nested linear graph that occurs in two nested linear graphs [25] (see also [34]). In a totally different context, Felser et al. [15] considered the matching problem regardless of precise pattern definition and proved that given a linear graph $G$ of size $m$, a maximum size nested subgraph of $G$ can be found in $O(m^3)$ time. The general problem of finding a maximum size edge-independent subgraph of $G$ is the well-known maximum matching problem [11]. On the other hand, given two linear graphs $G_1$ and $G_2$, finding a maximum size edge-independent linear graph that occurs both in $G_1$ and $G_2$ is an NP-complete problem [31]. Of particular importance is the following theorem which states that one can find a given nested linear graph in a linear graph in polynomial-time (see also [31]).

**Theorem 1** ([20]). Given a linear graph $G_1$ of size $m_1$ and a nested linear graph $G_2$ of size $m_2$, one can find an occurrence of $G_2$ in $G_1$, if it exists, in $O(m_2 \log m_2 + m_1 m_2)$ time.

Clearly, it follows from the above theorem that finding a maximum size nested subgraph in a set of linear graphs $G$ is polynomial-time solvable if $G$ contains at least one graph of logarithmic size with
Fig. 1. Shown here is an occurrence of the nested linear graph $G_{\text{sol}} = a^2 b^2 (ab)^2$ in $G = \{G_1, G_2, G_3\}$. The Max-NLS problem asks to find the largest nested linear graph $G_{\text{sol}}$ that occurs in each $G_i \in G$.

3 Finding a maximum size nested linear graph in a linear graph

Felsner et al considered in [15] the matching problem regardless of a precise pattern definition. In this context, they introduced the concept of circle trapezoid graphs (CT-graphs), a class of graphs that contains trapezoid graphs, circle graphs and circular-arc graphs as subclasses, and proved that, given a CT-graph $G$ with $m = \#E(G)$, a maximum size nested subgraph of $G$ can be found in $O(m^2)$ time. In this brief section, we improve that result for linear graphs by giving a simple dynamic programming algorithm for finding a maximum size nested subgraph of a linear graph in $O(n^2 + nm)$ time and $O(n^2)$ space, where $n = \#V(G)$.

Let $G$ be the input linear graph with $V(G) = \{1, 2, \ldots, n\}$, and assume $\#E(G) = m$. For each pair $(i, j)$ with $1 \leq i < j \leq n$, let $G[i, j]$ denote the subgraph of $G$ induced by the vertices $\{i, i+1, \ldots, j\}$, i.e., the graph obtained from $G$ by removing all vertices $k$ with either $k < i$ or $k > j$. For each pair $(i, j)$ with $1 \leq i \leq j \leq n$, let $\text{opt}[i, j]$ denote the maximum size of a nested subgraph of the linear graph $G[i, j]$. It is convenient to have the following - at first odd - convention: if $j < i$ then $G[i, j]$ stands for the empty graph and $\text{opt}[i, j] = 0$. Notice also that $\text{opt}[i, i] = 0$ for all $1 \leq i \leq n$. Hence our aim is to compute $\text{opt}[1, n]$. We will compute $\text{opt}[i, j]$ - viewed as a dynamic
programming table - in the following order (row by row and column by column):

$$
\begin{align*}
\text{opt}[n,n] \\
\text{opt}[n-1,n-1] & \text{opt}[n-1,n] \\
\text{opt}[n-2,n-2] & \text{opt}[n-2,n-1] & \text{opt}[n-2,n] \\
& \vdots & \vdots & \ddots \\
\text{opt}[1,1] & \text{opt}[1,2] & \ldots & \ldots & \ldots & \ldots & \text{opt}[1,n]
\end{align*}
$$

Concerning the elements in the first column, they are of the form $\text{opt}[i,i]$, and hence their value is set to 0. Let us now show how, for a generic $i$ with $1 \leq i < n$, the elements of row $i+1$, that is the elements $\text{opt}[n-i,n-i], \text{opt}[n-i,n-i+1], \ldots, \text{opt}[n-i,n]$ can be computed one after the other in total $O(m+n)$ time if we know the elements of all previous rows.

Let us thus concentrate on computing the generic element $\text{opt}[n-i,j]$ with $j > n-i$. The maximum nested subgraph of $G[n-i,j]$ either does not take vertex $j$, in which case $\text{opt}[n-i,j] = \text{opt}[n-i,j-1]$, or it takes an edge $(k,j) \in E(G)$ with $k < j$. We denote by $N^-(j)$ the set of those edges $(k,j) \in E(G)$ with $k < j$. It should now be clear that

$$
\text{opt}[n-i,j] = \max \left\{ \text{opt}[n-i,j-1], \max[1+\text{opt}[n-i,k-1]+\text{opt}[k+1,j-1] : k \in N^-(j) \land k \geq n-i] \right\}.
$$

Hence, the value of $\text{opt}[n-i,j]$ is computed by taking the maximum of at most $\#N^-(j) + 1$ terms, where each term can be computed in $O(1)$ time. Thus, the value of $\text{opt}[n-i,j]$ is computed in $O(\#N^-(j))$ time. Therefore, the total time to compute the whole row $i+1$ is $\sum_{j=n-i}^{m} O(\#N^-(j) + 1) = O(m+n)$ time since the edge sets $N^-(j)$ are disjoint subsets of $E(G)$. Then, it follows that $\text{opt}[1,n]$ is computed in $O(n^2 + mn)$ time.

### 4 A Polynomial-time algorithm for fixed $\#G$

According to [20], given a linear graph $G_1$ of size $m_1$ and a nested linear graph $G_2$ of size $m_2$, an occurrence of $G_2$ in $G_1$ can be found in $O(m_2 \log m_2 + m_1 m_2)$ time. Valiente proposed in [25] a dynamic programming algorithm for finding a largest nested linear graph that occurs in two nested linear graphs (see also [34]). In this section, we give a $O(m^2 \log m^2 + m \log n \log m^2)$ time dynamic programming algorithm, where $m = \max\{\#E(G_i) : G_i \in \mathcal{G}\}$, for finding a largest nested linear graph that occurs in a fixed number of linear graphs, thereby proving that the Max-NLS problem is polynomial-time solvable for fixed $\#G$.

We need some additional definitions and notations. We use the notions of trapezoid diagrams and $d$-trapezoid diagrams introduced in [9] and [16], respectively. Assume $d$ is a non-negative integer and let $L^1, L^2, \ldots, L^{d+1}$ be $d+1$ parallel lines indexed by their ordering in the plane. A graph $G$ is called a $d$-trapezoid graph [15, 6] if there exist families of intervals $T_u = \{I^i_u = [l^i_u, r^i_u] : l^i_u, r^i_u \in L^i, 1 \leq i \leq d+1\}, u \in V(G)$, satisfying $\{u, v\} \in E(G)$ if and only if $Q_u \cap Q_v \neq \emptyset$, where $Q_x$ denotes the closed polygon $(l^1_x, l^2_x, \ldots, l^{d+1}_x, r^1_x, \ldots, r^d_x)$. We refer to the family $T_G = \{T_u : u \in V(G)\}$ to as the $d$-trapezoid diagram of $G$. Note that 0-trapezoid graphs are precisely interval graphs [17], 1-trapezoid graphs are the usual trapezoid graphs [9], and $d$-trapezoid graphs are comparability graphs of posets with interval dimension at most $d + 1$ [16].

Based on a geometric representation of 1-trapezoid graphs by boxes in the plane, Felsner et al. [15] designed an optimal $O(n \log n)$ algorithm for finding a maximum weighted independent set on
such graphs. Of particular importance, they proved that the ideas behind the weighted independent set for trapezoid graphs carry over to higher dimension leading to a $O(n \log^d n)$ time algorithm for $d$-trapezoid graphs of order $n$. This has been improved in [1] to $O(n \log^{d-1} \log \log n)$ time. We now turn to defining an irreflexive, transitive and anti-symmetric relation $\sqsubset$ on $\mathcal{T}_G$. Let $\mathcal{G}$ be a $d$-trapezoid graph and $\mathcal{T}_G = \{T_u : u \in V(G)\}$ be the corresponding $d$-trapezoid diagram. Let $T_u = \{I_u = [l_u^i, r_u^i] : l_u^i, r_u^i \in L^i, 1 \leq i \leq d+1\}$ and $T_v = \{I_v = [l_v^i, r_v^i] : l_v^i, r_v^i \in L^i, 1 \leq i \leq d+1\}$ be two $d$-trapezoids of $\mathcal{T}_G$. We say that the $d$-trapezoid $T_u$ is strictly contained in $T_v$, written $T_u \sqsubset T_v$, if $l_u^i < l_v^i$ and $r_u^i < r_v^i$ for all $1 \leq i \leq d+1$.

Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ be an instance of the Max-NLS problem. We associate to $\mathcal{G}$ an $(n-1)$-trapezoid diagram as follows. For each $1 \leq i \leq n$, the graph $G_i$ is associated to Line $L^i$, and for each $(x, y) \in E(G_i)$ we define an interval $I_{x,y}^i = [x, y]$ on line $L^i$. We denote by $\mathcal{I}_{G_i}$ the set of all intervals on $L^i$ that are associated to the graph $G_i$, i.e., $\mathcal{I}_{G_i} = \{I_{x,y}^i : (x, y) \in E(G_i)\}$. The $(n-1)$-trapezoid diagram induced by $\mathcal{G}$, written $\mathcal{T}[\mathcal{G}]$, is defined as follows:

$$\forall I_1 \in \mathcal{I}_{G_1}, \forall I_2 \in \mathcal{I}_{G_2}, \ldots, \forall I_n \in \mathcal{I}_{G_n}, \{I_1, I_2, \ldots, I_n\} \in \mathcal{T}[\mathcal{G}].$$

Clearly, $\#\mathcal{T}[\mathcal{G}] = \prod_{G_i \in \mathcal{G}} \#E(G_i)$. Having disposed of these preliminaries, we now turn to presenting our algorithm, referred hereafter to as Algorithm nested-linear-subgraph, for finding a largest nested linear graph $G$ that occurs in a family of linear graphs $\mathcal{G}$. The basic idea is to associate to each $(n-1)$-trapezoid $T \in \mathcal{T}[\mathcal{G}]$ a weight $\omega(T)$ denoting the maximum size of a nested linear graph that occurs in the family of linear graphs induced by $T$ and all $(n-1)$-trapezoids strictly included in $T$. This is done in turn by dynamic programming according to a linear extension of $\mathcal{T}[\mathcal{G}]$. We need the following subroutine: Given an $(n-1)$-trapezoid diagram $T$ and a function $\omega : \mathcal{T}[\mathcal{G}] \rightarrow \mathbb{N}^+$, we refer to the algorithm for finding a maximum weighted disjoint subset of $T$ (in terms of disjoint induced closed polygons) as max-weighted-independent-set($T$, $\omega$) [15, 1]. A more schematic description of Algorithm nested-linear-subgraph($\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$) is given in Figure 2. The following lemma is crucial for correctness of the algorithm.

**Lemma 1.** Let $e_1 \in E(G_1), e_2 \in E(G_2), \ldots, e_n \in E(G_n)$, and $T$ be the corresponding $(n-1)$-trapezoid in $\mathcal{T}[\mathcal{G}]$. After having processed $T$ in the main loop (Lines 7-12) of Algorithm nested-linear-subgraph($\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$), $\omega(T)$ is the size of a maximum size nested linear graph that occurs in each edge-induced subgraph $G_i[E_i], G_i \in \mathcal{G}$, where $E_i = \{e_i\} \cup \{e : e \in E(G_i) \land e \sqsubset e_i\}$.

**Proposition 1.** The Max-NLS problem is solvable in $O(m^2 n \log^{n-2} m n \log \log m^n)$ time, where $n = \#\mathcal{G}$ and $m = \max \{\#E(G_i) : G_i \in \mathcal{G}\}$.

We have thus proved that the Max-NLS problem is polynomial-time solvable for fixed $\mathcal{G}$. This result gains in interest if we compare Proposition 1 to the related LAPCS problem restricted to two nested arc-annotated sequences, i.e., the LAPCS(Nested, Nested) problem, restricted to unary alphabet, which has been proved to be NP-complete in [24] (only two nested arc-annotated sequences, and hence two linear graphs here).

### 5 Hardness results

It is proved in [10] that the Max-NLS problem is NP-complete even when restricted to edge-independent linear graphs. We sharply strengthen this result by showing that the problem is hard
even for flat linear graphs of height at most 2. Observe that our result gives a precise borderline between tractable and intractable instances of the Max-NLS problem (the problem is indeed trivially polynomial-time solvable for nested linear graphs of height at most 1: find the stability of $\#G$ associated interval graphs and return the maximum stability found). We also note that this result justifies the need for a second step in the approximation ratio analysis in [10] (the problem is indeed hard for flat graphs).

Our result is a three-step procedure. We begin by proving the \textsf{NP}-hardness of a satisfiability-like problem, namely the \textsf{Convenient 3–Sat} problem. Next, we use that result for proving that a new list problem, \textit{i.e.}, the \textsf{Longest Common Sublist} problem, is \textsf{NP}-complete. Finally, we give a polynomial-time reduction from the \textsf{Longest Common Sublist} problem to prove that the \textsf{Max-NLS} problem is \textsf{NP}-complete even when restricted to simple instances solely composed of flat linear graphs of height at most 2.

We thus begin by introducing the \textsf{Convenient 3–Sat} problem. We are given as input an ordered list $\mathcal{C} = C_1, C_2, \ldots, C_m$ of clauses making a 3-Sat formula $\phi$ on a set of variables $X = \{x_1, \ldots, x_n\}$. Each clause $C_j$ contains 3 different literals, where a literal is either a variable in $X$ or its negation. We assume each variable $x_i \in X$ has $\sigma_i$ occurrences $x_i^1, \ldots, x_i^{\sigma_i}$, and let $S = \bigcup_{i=1}^n \{x_i^1, \ldots, x_i^{\sigma_i}\}$ be the set of all the occurrences in $\mathcal{C}$. Moreover, for every $1 \leq i \leq n$ and every $1 \leq j \leq \sigma_i$ let $p(i,j)$ be the \textit{pedex} in $\{1,2,\ldots,m\}$ such that clause $C_{p(i,j)}$ contains the occurrence $x_i^j$ of variable $x_i$. It is assumed that $p(i,1) < p(i,2) < \ldots < p(i,\sigma_i)$ for every $1 \leq i \leq n$. A subset $S' \subseteq S$ is called \textit{consequent} if, for every $1 \leq i \leq n$ and every $1 \leq j \leq \sigma_i - 1$, the occurrence $x_i^{j+1}$ belongs to $S'$ whenever the occurrence $x_i^j$ belongs to $S'$. Given a consequent subset $S' \subseteq S$, we denote by $\mathcal{C}(S')$ the set of those clauses of $\mathcal{C}$ which contain at least one literal involving either a positive occurrence in $S'$ or a negated occurrence not in $S'$. We are now in position to formally define the \textsf{Convenient 3-Sat} decision problem.

```
Algorithm nested-linear-subgraph($G = \{G_1, G_2, \ldots, G_n\}$)

Data : A family $G = \{G_1, G_2, \ldots, G_n\}$ of $n$ linear graphs.
Result : The maximum size of a nested linear graph that occurs in each $G_i \in G$.

begin
  1. Compute the $(n-1)$-trapezoid diagram $T[G]$ induced by $G$.
  2. Compute any linear extension $\Phi$ of $(T[G], \sqsubseteq)$.
  3. foreach $T \in T[G]$ do
     4.      $\omega(T) := 0$.
     5.      foreach $T' \in T[G]$ with respect to $\Phi$ do
     6.             $T' := \{T' : T' \sqsubseteq T\}$.
     7.             $T'' := \max$-weighted-independent-set$(T', \omega)$.
     8.             $\omega(T) := 1 + \sum_{T'' \subseteq T'} \omega(T'')$.
     9.         end
  10. $T^* := \max$-weighted-independent-set$(T[G], \omega)$.
  11. return $\sum_{T \subseteq T^*} \omega(T)$.
end
```

Fig. 2. Algorithm $\text{nested-linear-subgraph}(G = \{G_1, G_2, \ldots, G_n\})$ for finding the maximum size of a nested linear graph $G$ that occurs in a family of linear graphs $G$. 
Convenient 3-Sat

Instance: An ordered list $C = C_1, C_2, \ldots, C_m$ of clauses making a 3-Sat formula $B$ on a set of variables $X = \{x_1, \ldots, x_n\}$.

Question: Is there a consequent subset $S'$ of $S_C$ such that $C(S') = C$?

Informally, we are trying to “satisfy all clauses” where we are partially allowed to change the boolean value of variables when scanning formula $B$ from left to right. The only constraint we must obey is that, once a variable has been true in some occurrence, it cannot become false in later occurrences.

Proposition 2. The Convenient 3-Sat problem is NP-complete.

Having disposed of this preliminary step, we now turn to introducing our second problem. We need new notations. When $L$ is a list of integers, we denote by $\text{len}(L)$ the length of $L$ and by $L[i]$ the value of the $i$-th integer in $L$, $1 \leq i \leq \text{len}(L)$. A sublist of $L$ is any list obtained from $L$ by dropping some of the elements in $L$. Clearly, $L$ admits $2^{\text{len}(L)}$ sublists. We can now state formally the Longest Common Sublist decision problem.

Longest Common Sublist

Instance: Lists of positive integers $L_1, L_2, \ldots, L_n$, and a positive integer $k$.

Question: Are there lists $\tilde{L}_1, \tilde{L}_2, \ldots, \tilde{L}_n$, where $\tilde{L}_i$ is a sublist of $L_i$, $1 \leq i \leq n$, and $\text{len}(\tilde{L}_1) = \text{len}(\tilde{L}_2) = \ldots = \text{len}(\tilde{L}_n) = \ell$, such that $\sum_{1 \leq j \leq \ell} \min\{\tilde{L}_i[j] : 1 \leq j \leq n\} \geq k$?

We remark that the above problem admits a polynomial-time algorithm both when $m$ is bounded by a constant (there is dynamic programming solution) and when the lengths of the strings the in input are bounded by a constant (by simple guess of the single contributions to the objective function value).

We now prove the result that the Longest Common Sublist problem is NP-complete even if all integer values in the lists are either 1’s or 2’s. We present a reduction from the above introduced Convenient 3-Sat problem to the Longest Common Sublist problem. Assume thus given a generic instance of the Convenient 3-Sat problem, that is, an ordered list $C = C_1, C_2, \ldots, C_m$ of clauses making a 3-SAT formula $\phi$ on a set of variables $X = \{x_1, x_2, \ldots, x_n\}$. To such an instance of the Convenient 3-Sat problem, we associate an instance of the Longest Common Sublist problem - in which all numbers are either 1 or 2 - as follows. First, for every $C_j \in C$, we impose an arbitrary order on the 3 literals of $C_j$, and then denote by $C_{j1}$ (resp., $C_{j2}$ and $C_{j3}$) the first (resp., second and third) literal of $C_j$. Let $L_0$ be the following list of $50m + 5$ integers in $\{1, 2\}$.

$$L_0 = \left( \prod_{j=1}^{m} 2^5 1^{10} (212121) 1^{29} \right) 2^5.$$  

Besides this one list $L_0$, called the ruler, our instance of the Longest Common Sublist problem contains a list $L_{n+1}$, called the selector. The selector is the following list of $54m + 5$ integers in $\{1, 2\}$.

$$L_{n+1} = \left( \prod_{j=1}^{m} 2^5 1^{14} 2 1^{34} \right) 2^5.$$
Besides the ruler and the selector, our instance of the Longest Common Sublist problem contains $n$ other lists $L_1, L_2, \ldots, L_n$, where list $L_i$ represents variable $x_i$ of $X$. For each $1 \leq i \leq n$, list $L_i$ contains $50m + 6$ integers in $\{1, 2\}$ and is defined as

$$L_i = \left( \prod_{j=1}^{m} 2^{6j^{10}s(i, j)} \right)^{129},$$

where

$$s(i, j) = \begin{cases} 222222 & \text{if } C_j \text{ contains no occurrence of } x_i, \\ 122222 & \text{if } C_j \text{ is a positive occurrence of } x_i, \\ 212222 & \text{if } C_j \text{ is a negative occurrence of } x_i, \\ 221222 & \text{if } C_j \text{ is a positive occurrence of } x_i, \\ 222122 & \text{if } C_j \text{ is a negative occurrence of } x_i, \\ 222212 & \text{if } C_j \text{ is a positive occurrence of } x_i, \\ 222221 & \text{if } C_j \text{ is a negative occurrence of } x_i. \end{cases}$$

Lemma 2. Assume the instance of the Convenient 3-Sat problem admits a consequent subset $S' \subseteq S$ such that $C(S') = C$. Then, there exists sublists $\tilde{L}_0, \tilde{L}_1, \ldots, \tilde{L}_n$, $\tilde{L}_{n+1}$ with $\tilde{L}_j$ sublist of $L_j$ for $0 \leq j \leq n + 1$ and $\text{len}(\tilde{L}_0) = \text{len}(\tilde{L}_1) = \cdots = \text{len}(\tilde{L}_{n+1})$, and such that $\sum_{j=1}^{\text{len}(\tilde{L}_1)} \min_{0 \leq i \leq n+1} \tilde{L}_i[j] \geq 55m + 5 + m$.

Lemma 3. Assume there exist sublists $\bar{L}_0, \bar{L}_1, \ldots, \bar{L}_n$, $\bar{L}_{n+1}$ with $\bar{L}_j$ sublist of $L_j$ for $0 \leq j \leq n + 1$ and $\text{len}(\bar{L}_0) = \text{len}(\bar{L}_1) = \cdots = \text{len}(\bar{L}_{n+1})$, and such that $\sum_{j=1}^{\text{len}(\bar{L}_1)} \min_{0 \leq i \leq n+1} \bar{L}_i[j] \geq 55m + 5 + m$. Then the instance of the Convenient 3-Sat problem admits a consequent subset $S'$ of $S$ such that $C(S') = C$.

On the basis of the reduction described above and Lemma 2 and 3, we can now state the following proposition.

Proposition 3. The Longest Common Sublist problem is NP-complete even if all integer values in the lists are either 1’s or 2’s.

Most of the interest in the Longest Common Sublist problem stems from the following proposition.

Proposition 4. The Max-NLS problem for flat linear graphs of height at most 2 is NP-complete.

Proof (of Proposition 4). Let an instance of the Longest Common Sublist problem - where all integer values in the lists are either 1’s or 2’s - be given by $n$ lists of positive integers $L_1, L_2, \ldots, L_n$, and by a positive integer $k$. We construct a corresponding set $G$ of $n$ linear graphs as follows (we abbreviate in a natural way a nested linear graph $G$ of size $n$ to a Dyck word of semi-length $n$ over the alphabet $A = \{a, b\}$). For each list $L_i$, $1 \leq i \leq n$, we add the nested linear graph $G_i$ defined by

$$G_i = a^{L_i[1]}b^{L_i[1]}a^{L_i[2]}b^{L_i[2]} \cdots a^{L_i[\text{len}(L_i)]}b^{L_i[\text{len}(L_i)]}$$

to $G$. It is easily seen that $G$ is composed of flat linear graphs of height at most 2 since all integer values in the lists are either 1’s or 2’s, and that our construction can be carried on in polynomial-time.
Suppose that there exist lists \( \tilde{L}_1, \tilde{L}_2, \ldots, \tilde{L}_n \), where \( \tilde{L}_i \) is a sublist of \( L_i, 1 \leq i \leq n \), and \( \text{len}(\tilde{L}_1) = \text{len}(\tilde{L}_2) = \ldots = \text{len}(\tilde{L}_n) = \ell \), such that \( \sum_{1 \leq i \leq \ell} \min \{ \tilde{L}_i[j] : 1 \leq i \leq n \} \geq k \). Let \( \tilde{L} \) be the list of length \( \ell \) defined by \( \tilde{L}[j] = \min \{ \tilde{L}_i[j] : 1 \leq i \leq n \} \) for \( 1 \leq j \leq \ell \). Now, consider the flat linear graph \( G_{\text{sol}} \) defined by:

\[
G_{\text{sol}} = a^{\tilde{L}[1]} b^{\tilde{L}[1]} a^{\tilde{L}[2]} b^{\tilde{L}[2]} \ldots a^{\tilde{L}[\ell]} b^{\tilde{L}[\ell]}
\]

It can be easily verified that \( G_{\text{sol}} \) is a flat linear graph of size \( \sum_{1 \leq j \leq \ell} \min \{ \tilde{L}_i[j] : 1 \leq i \leq n \} \geq k \) that occurs in each \( G_i \in \mathcal{G} \).

Conversely, suppose that there exists a linear graph \( G_{\text{sol}} \) of size at least \( k \) that occurs in each \( G_i \in \mathcal{G} \). Since \( \mathcal{G} \) is composed of flat linear graphs, \( G_{\text{sol}} \) is a flat linear graph. Therefore, \( G_{\text{sol}} \) can be written \( G_{\text{sol}} = a^{z_1} b^{z_1} a^{z_2} b^{z_2} \ldots a^{z_\ell} b^{z_\ell} \) for some non-negative integers \( z_1, z_2, \ldots, z_\ell \), and \( z_1 + z_2 + \ldots + z_\ell \geq k \). Consider the list of positive integers \( \tilde{L} \) of length \( \ell \) defined by \( \tilde{L}[j] = z_j \) for \( 1 \leq j \leq \ell \). Clearly \( \sum_{1 \leq j \leq \ell} \tilde{L}[j] = z_1 + z_2 + \ldots + z_\ell \geq k \). By construction, for all \( 1 \leq i \leq n \), there exists a sublist \( \tilde{L}_i \) of \( L_i \) of length \( \ell \) such that \( \tilde{L}_i[j] \geq \tilde{L}[j] \).

Here below we offer a reformulation of Proposition 4 that may be of independent interest.

**Proposition 5.** Finding the largest common homeomorphic subtree in a set of ordered rooted trees of height at most 3 is an NP-complete problem.

### 6 Approximation algorithms

Using a nice two step algorithm, Davydov and Batzoglou proved in [10] that the Max-NLS problem is approximable within ratio \( O(\log^2 k) \), where \( k \) is the size of the optimal common nested structure. This short section is devoted to further approximation analysis of the Max-NLS problem. We improve the approximation ratio of [10] by showing that the Max-NLS problem is approximable within ratio \( [\log k] + 1 \), where \( k \) is the size of the optimal common nested structure. Also, we give two non-trivial approximation results for the Max-NLS problem (the proof of Proposition 7 uses a standard argument).

Let \( \mathcal{G} = \{ G_1, G_2, \ldots, G_n \} \) be an arbitrary instance of the Max-NLS problem. Let \( G_{\text{opt}} \) denote a minimum size nested linear graph that occurs in each input graph \( G_i \in \mathcal{G} \) and write \( m_{\text{opt}} = \#E(G_{\text{opt}}) \). Also, let \( G' \) be the largest \((w, h)\)-level graph that occurs in each input graph \( G_i \in \mathcal{G} \) and write \( m' = \#E(G') \). For \( i = 1, 2, \ldots, m_{\text{opt}} \), let \( m_i \) be the number of edges of \( G_{\text{opt}} \) of height \( i \). For one, we have \( \sum_{i=1}^{m_{\text{opt}}} m_i = m_{\text{opt}} \). For another, \( i \cdot m_i \leq m' \) since \( i \cdot m_i \) is the size of a feasible solution, i.e., a \((m_i, i)\)-level graph, that occurs in \( G_{\text{opt}} \). Then it follows that

\[
m_{\text{opt}} = \sum_{i=1}^{m_{\text{opt}}} m_i \leq \sum_{i=1}^{m_{\text{opt}}} \frac{m'}{i} = m' \sum_{i=1}^{m_{\text{opt}}} \frac{1}{i} \leq m'([\log m_{\text{opt}}] + 1)
\]

It is proved in [10] that one can check in polynomial-time whether a given \((w, h)\)-level graph occurs in a (general) linear graph. Since, one has to check only a polynomial number of \((w, h)\)-level graphs, we have thus proved the following.

**Proposition 6.** The Max-NLS problem is approximable within ratio \( [\log k] + 1 \), where \( k \) is the size of an optimal solution.

We give below two approximations ratios (the second focuses on moderate height linear graphs).
Proposition 7. The Max-NLS problem is approximable within ratio $\mathcal{O}(\frac{m}{\log m})$, where $m$ is the minimum size of a linear graph in $G$, i.e., $m = \min\{ \#E(G_i) : G_i \in \mathcal{G} \}$.

Proposition 8. The Max-NLS problem for linear graphs of height at most $h$ is approximable within ratio $h$.

It follows from Proposition 8 that, for fixed $h$, the Max-NLS problem for linear graphs of height at most $h$ is in APX. Of particular importance, we do not know, however, whether the problem is complete for APX.

7 Parameterized complexity

One of the latest approaches to attack computational intractability is to study parameterized complexity [12]. For many hard problems, the seemingly unavoidable combinatorial explosion can be restricted to a “small part” of the input, the parameter, so that the problems can be solved in polynomial time when the parameter is fixed. The parameterized problems that have algorithms of $f(k) \cdot n^{O(1)}$ time complexity are called fixed-parameter tractable, where $k$ is the parameter and $f$ can be an arbitrary function depending only on $k$, and $n$ denotes the overall input size. We designate the class of fixed-parameter tractable problems FPT. Parameterized complexity also describes a hierarchy of parameterized complexity classes $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[P]$ for which there are many natural hard or complete problems. In the last decade, parameterized complexity has proved to be extremely useful in computational molecular biology, see for example [4, 20, 2]. The best general reference here is [12].

We investigate in this section the parameterized complexity of the Max-NLS problem. We show that the parameterized version of the problem has complexity varying from FPT to $W[t]$-hard for all $t \geq 1$ depending on the parameter and the structure of the linear graphs. The emphasis here is on the combination of several parameters: the crossing structure, the height and the depth. More precisely, we prove that the Max-NLS problem is fixed-parameter tractable when parameterized by the length of the requested common nested linear graph. When the parameter is the number of linear graphs, we show that the problem is $W[1]$-hard for (i) nested linear graphs of height at most 3 and for (ii) edge-independent linear graphs of height at most 3 and depth at most 2, and $W[t]$-hard for all $t \geq 1$ for (i') edge-independent linear graphs of height at most 3, for (ii') non-crossing linear graphs of height at most 3 and for (iii') nested linear graphs of height at most 4.

Proposition 9. The Max-NLS problem is solvable in $O(4^k n (k \log k + mk))$ time, and hence is fixed-parameter tractable when parameterized by the size $k$ of the common nested subgraph.

Proposition 9 has to be compared with the classical Longest Common Subsequence parameterized by the length of the common subsequence which belongs to $W[P]$ and is hard for $W[2]$ for unbounded alphabet [5, 12].


It remains open, however, whether the Max-NLS problem for nested linear graphs of height at most 3 is complete for $W[1]$. Nevertheless, in the light of Proposition 10, we could naturally ask what would happen if we drop the nested or edge-independent assumption. As shown in the following, dropping either of these assumptions seems to result in an increase in complexity (Propositions 11 and 13).
Proposition 11. The Max-NLS problem for edge-independent linear graphs of height at most 3 is $\mathsf{W}[t]$-hard for all $t \geq 1$ when parameterized by $n = \#G$.

We note that we are only able to prove Proposition 11 for unbounded depth graphs. We stress in the following proposition that bounding the depth in addition to the height does not seem however to result in an increase in complexity (compare Propositions 10, 11 and 12).

Proposition 12. The Max-NLS problem for edge-independent linear graphs of height at most 3 and depth at most 2 is $\mathsf{W}[1]$-hard when parameterized by $n = \#G$.

We now turn to dropping the assumption of edge independence.

Proposition 13. The Max-NLS problem for non-crossing linear graphs of height at most 3 and depth at most 1 is $\mathsf{W}[t]$-hard for all $t \geq 1$ when parameterized by $n = \#G$.

We observe that, in contrast to Proposition 11, we are able to prove that Proposition 12 holds true even for linear graphs of depth 1. This raises the important question of whether the Max-NLS problem for edge-independent linear graphs of height at most 3 and depth at most 2 is complete for $\mathsf{W}[1]$. Finally, dropping the assumption of linear graphs of height at most 3 seems to result in an increase in complexity even when restricted to nested linear graphs.

Proposition 14. The Max-NLS problem for nested linear graphs of height at most 4 is $\mathsf{W}[t]$-hard for all $t \geq 1$ when parameterized by $n = \#G$.

8 Conclusion

In the context of non-coding RNA multiple structural alignment, we considered the problem of finding a largest nested linear graph in a set $G$ of linear graphs. We described a polynomial-time algorithm for fixed $\#G$, and gave approximability results. From a parameterized complexity point of view, we showed that the problem has complexity varying from $\mathsf{FPT}$ to $\mathsf{W}[t]$-hard for all $t \geq 1$ depending on the parameter and the structure of $G$.

In conclusion, we mention some interesting directions for future works. The parameterized complexity and the approximation of the Max-NLS problem for linear graphs of height at most 2 are completely unexplored. It would also be of interest to know whether the Max-NLS problem for linear graphs of bounded height is complete for $\mathsf{APX}$. More generally, the approximation aspect of the Max-NLS problem has to be improved. Finally, we do believe that investigating generalizations of the Max-NLS problem involving more complex structures is of particular importance from both a theoretical and a practical computational biology point of view (bi-secondary structures seem to be an interesting starting point).

References

Appendix (Program committee version only)

Proof (of Proposition 1). Correctness of Algorithm \texttt{nested-linear-subgraph}(\mathcal{G} = \{G_1, G_2, \ldots, G_n\}) follows from Lemma 1. Having disposed of this preliminary step, we now turn to proving the time complexity. Clearly, \#T[\mathcal{G}] \leq \prod_{G_i \in \mathcal{G}} \#E(G_i) \leq (\max\{\#E(G_i) : G_i \in \mathcal{G}\})^n = m^n. Hence, computing a linear extension of \((T[\mathcal{G}], \emptyset)\) can be done in \(O(m^{2n})\) time using classical algorithms, e.g., a depth first search algorithm \cite{8}. We now consider one iteration (Lines 7-11) of Algorithm \texttt{-Sat}_{\mathcal{G}}. Let \(\tilde{\mathcal{G}}\) be a linear extension of \((T[\mathcal{G}], \emptyset)\). According to Theorem 1, this takes \(O(n \log n)\) time, where \(\tilde{\mathcal{G}}\) is a linear extension of \((T[\mathcal{G}], \emptyset)\). Clearly, \(\mathcal{G}\) is satisfiable if and only if \(\tilde{\mathcal{G}}\) is satisfiable. Therefore, for every \(i = 1, 2, \ldots, m\), we introduce the clause \(C_i = \tilde{C}_i\). Denote by \(S = \cup_{i=1}^{n} \{x_1^i, x_2^i, \ldots, x_n^i\}\) the set of occurrences of the new instance the \texttt{Convenient 3-Sat} problem.

Assume \(\tilde{B}\) is satisfiable and let \(\phi : X \mapsto \{\text{true, false}\}\) be a truth assignment such that each clause in \(\tilde{C}\) evaluates to \text{true} under \(\phi\). Consider the subset \(S'\) of \(S\) made of all those \(x_j^i\) (with \(j = 1, \ldots, \sigma_i\)) such that \(\phi(x_j^i) = \text{true}\). Clearly, \(S'\) is consequent since, for each variable \(x_i\), either \(S'\) contains all occurrences of \(x_i\) or none. Moreover, \(C(S') = \tilde{C}\). Indeed, for \(t = 0, 1, \ldots, n\) and for \(j = 1, 2, \ldots, m\), we introduce the clause \(C_{j+t} = \tilde{C}_j\). Therefore, for every \(i = 1, 2, \ldots, m\), \(S_i = \tilde{S}_i\) is \(t\)-standard. Consider the truth assignment \(\phi : X \mapsto \{\text{true, false}\}\) such that \(\phi(\tilde{x}_i) = \text{true}\) if all the occurrences of \(x_i\) within clauses \(C_{i+1}, C_{i+2}, \ldots, C_{i+t}\) are in \(S'\) and \(\phi(\tilde{x}_i) = \text{false}\) if no such occurrence of \(x_i\) is in \(S'\). Since \(C(S') = \tilde{C}\), each one of the clauses \(C_{i+1}, C_{i+2}, \ldots, C_{i+t}\) belongs to \(\tilde{C}\) and hence each one of these clauses evaluates to true under \(\phi\). Therefore, for \(j = 1, 2, \ldots, m\), \(C_j = \tilde{C}_j\) evaluates to true under \(\phi\). \(\square\)

Proof (of Proposition 2). The \texttt{Convenient 3-Sat} problem is easily seen to be in \texttt{NP}. We present a reduction from the \texttt{3-Sat} problem. Assume given an instance \(\tilde{B} = (X, \tilde{C})\) of the \texttt{3-Sat} problem, where \(X = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}\) and \(\tilde{C} = \{C_1, C_2, \ldots, C_m\}\). To \(\tilde{B} = (X, \tilde{C})\) we associate an instance of the \texttt{Convenient 3-Sat} problem obtained as follows. First, take \(\{x_1, x_2, \ldots, x_n\}\) as your set of boolean variables \(X\). Next, for \(t = 0, 1, \ldots, n\), we introduce the clause \(C_{t+1} = \tilde{C}_t\). Conversely, suppose that the reduction described above admits a consequent subset \(S'\) of \(S\) such that \(C(S') = \tilde{C}\). For any given \(t = 0, 1, \ldots, n\), we say that \(S'\) is \(t\)-standard if, for every \(i = 1, 2, \ldots, n\), either all or none of the occurrences of \(x_i\) within clauses \(C_{i+1}, C_{i+2}, \ldots, C_{i+t}\) are in \(S'\). Since \(S'\) is consequent, then there exists a \(t\) such that \(S'\) is \(t\)-standard. Consider the truth assignment \(\phi : X \mapsto \{\text{true, false}\}\) such that \(\phi(\tilde{x}_i) = \text{true}\) if all the occurrences of \(x_i\) within clauses \(C_{i+1}, C_{i+2}, \ldots, C_{i+t}\) are in \(S'\) and \(\phi(\tilde{x}_i) = \text{false}\) if no such occurrence of \(x_i\) is in \(S'\). Since \(C(S') = \tilde{C}\), each one of the clauses \(C_{i+1}, C_{i+2}, \ldots, C_{i+t}\) belongs to \(\tilde{C}\) and hence each one of these clauses evaluates to true under \(\phi\). Therefore, for \(j = 1, 2, \ldots, m\), \(C_j = \tilde{C}_j\) evaluates to true under \(\phi\). \(\square\)

Proof (of Proposition 7). Let \(G_i \in \mathcal{G}\) be a minimum size linear graph in \(\mathcal{G}\). Assume \(\#E(G_i) = m\). Divide the edge set \(E(G_i)\) into \(p = \frac{m}{q}\) subsets \(E_1^q, E_2^q, \ldots, E_p^q\), each containing \(q\) edges. For each edge subset \(E_l^q\), \(1 \leq l \leq p\), check all subsets to see if they induce a common nested linear graph in \(\mathcal{G}\). According to Theorem 1, this takes \(O(n(m' \log m' + m' q)2^n)\) time where \(m'\) is the maximum size of a linear graph in \(\mathcal{G}\). Thus picking \(q = O(\log m)\) yields a procedure that takes polynomial-time, i.e., \(O(nm' \log m')\) time.
The success of this straightforward algorithm relies on the fact that removing edges from a common nested linear graph result in another common nested linear graph. Let \( G \) be a largest common nested linear graph in \( G \). By the pigeonhole principle, at least one of the edge subset \( E_\ell, 1 \leq \ell \leq p \), must contain at least \( \frac{q}{m|D|} \) of the edges of an occurrence of \( G \) in \( G_i \). Thus letting these edge subset be our common nested linear graph, our approximation algorithm finds a common nested linear graph \( G' \) satisfying \( \#E(G') \geq \frac{q}{m} \#E(G) \). This is an approximation ratio of \( O(\frac{m}{\log m}) \).

\[ \square \]

Proof (Of Lemma 2). Let \( \tilde{L}_0 := L_0 \). Since \( C(S') = C \), then, for every \( j = 1, 2, \ldots, m \), clause \( C_j \) contains a literal which is either a positive occurrence in \( S' \) or a negative occurrence not in \( S' \); we hence define \( t(j) \) so that \( C_{t(j)} \) is such a literal. We can now define \( \tilde{L}_{n+1} \) as follows.

\[
\tilde{L}_{n+1} = \left( \prod_{j=1}^{m} 2^{5}1^{10} T(j) 1^{29} \right) 2^5.
\]

where

\[
T(j) = \begin{cases} 
211111 & \text{if } t(j) = 1, \\
112111 & \text{if } t(j) = 2, \\
11121 & \text{if } t(j) = 3.
\end{cases}
\]

(2)

Finally, for \( i = 1, 2, \ldots, n \), let

\[
\tilde{L}_i = \left( \prod_{j=1}^{m} 2^{6}1^{9} s(i, j) 1^{29} \right) 2^5
\]

if no occurrence of \( x_i \) belongs to \( S' \). Otherwise, where \( pos(i) \in \{1, \ldots, m\} \) is such that \( C_i \) contains the first occurrence of \( x_i \) which belongs to \( S' \), then let

\[
\tilde{L}_i = \left( \prod_{j=1}^{pos(i)-1} 2^{6}1^{9} s(i, j) 1^{29} \right) 2^{6}1^{8} s(i, j) 1^{29} \left( \prod_{j=pos(i)+1}^{m} 2^{6}1^{9} s(i, j) 1^{29} \right) 2^6.
\]

The reader is invited to check that

\[
\sum_{j=1}^{\text{len}(L_1)} \min_{i \in \{0, 1, \ldots, n+1\}} \tilde{L}_i[j] \geq 55m + 5 + m.
\]

\[ \square \]

Proof (Of Lemma 3). Assume

\[
\sum_{j=1}^{\text{len}(L_1)} \min_{i \in \{0, 1, \ldots, n+1\}} \tilde{L}_i[j] \geq 56m + 5.
\]
Since \( \text{len}(L_0) = 50m + 5 \) and since the number of 2’s contained in list \( L_{n+1} \) is \( 6m \), then it follows both that \( L_0 = L_0 \) and that no one of the 2’s contained in \( L_{n+1} \) has been dropped in producing \( \tilde{L}_{n+1} \). For every \( i = 1, 2, \ldots, n \), \( \text{len}(\tilde{L}_i) = \text{len}(L_0) = \text{len}(L_0) = \text{len}(L_i) - 1 \), hence list \( \tilde{L}_i \) has been obtained from list \( L_i \) dropping just one element. And clearly, we can always assume the dropped element is a 1, which allows us to simplify the analysis of the cases here below. Let \( \text{pos}(i) \) be the position in list \( L_i \) of this one 1 element which has been removed in obtaining list \( \tilde{L}_i \). We now define a consistent subset \( S' \) of \( S \) by specifying, for every \( i = 1, 2, \ldots, n \), those occurrences among \( x_1^i, \ldots, x_{\text{len}(i)}^i \) which belong to \( S' \). Indeed, we put \( x_i^j \) in \( S' \) if and only if \( x_i^j \) occurs in a clause \( C_t \) with \( t \geq 1 + \frac{\text{pos}(i)-21}{40} \). Clearly, under this definition, \( S' \) is certainly consistent. Moreover, \( C(S') = C \).

Let \( \sigma_i \) be a consistent subset \( S' \) of \( S \) by specifying, for every \( i = 1, 2, \ldots, n \), \( \text{Height}(\sigma_i) \) which belong to \( S' \). Indeed, for every \( C_t \in C \), we can observe that the \( 6j \)-th 2 element in the selector \( L_{n+1} \) must have been paired with either the \( 8j \)-th or the \( (8j - 1) \)-th or the \( (8j - 2) \)-th 2 element in the ruler \( L_0 \). And all lists \( \tilde{L}_i \) must present a 2 in this position. Hence either the last, or the second last, or the first literal in \( C_j \) is either a positive occurrence in \( S' \) or a negative occurrence not in \( S' \).

\[ \text{Proof (of Proposition 8).} \]

Let \( G^* \) be a maximum size common nested linear graph of \( \mathcal{G} \). Let \( G' \) be a \((W, H)\)-level graph where \( W = \min\{\text{Width}(G_i) : G_i \in \mathcal{G}\} \) and \( H = 1 \). According to [31], computing \( W \) can be done in \( O(nm \log m) \) time where \( n = |\mathcal{G}| \) and \( m \) is the maximum size of a linear graph in \( \mathcal{G} \). Clearly, \( G' \) is a common nested linear graph of \( \mathcal{G} \). Thus, letting \( G' \) be our common nested linear graph, our approximation algorithm finds a solution satisfying

\[
\frac{\#E(G^*)}{\#E(G')} = \frac{\min\{\text{Width}(G_i) : G_i \in \mathcal{G}\}}{\max\{\text{Height}(G_i) : G_i \in \mathcal{G}\} \cdot \min\{\text{Width}(G_i) : G_i \in \mathcal{G}\}} \leq \frac{\text{max}\{\text{Height}(G_i) : G_i \in \mathcal{G}\}}{\text{min}\{\text{Width}(G_i) : G_i \in \mathcal{G}\}} = \max\{\text{Height}(G_i) : G_i \in \mathcal{G}\}
\]

This is an approximation ratio of \( \max\{\text{Height}(G_i) : G_i \in \mathcal{G}\} = h \).

\[ \text{Proof (of Proposition 9).} \]

The algorithm is by direct enumeration. The number of Dyck words of length \( 2k \), \( k > 0 \), is given by the Catalan number \( C_k = \frac{1}{k+1} \binom{2k}{k} \). Asymptotically, the Catalan numbers grow as \( C_k \sim 4^k / k \sqrt{k \pi} \) [18]. Combining this with Theorem 1 yield the desired result.

\[ \text{Proof (of Proposition 10).} \]

We describe a parameterized reduction from the Longest Common Subsequence problem for binary alphabet parameterized by the number of strings, which is known to be \( W[1] \)-hard [27]. Let an instance of the Longest Common Subsequence problem for binary alphabet be given by a set of strings \( S = \{s_1, s_2, \ldots, s_n\} \) over the binary alphabet \( A = \{0, 1\} \) and a positive integer \( k \) denoting the size of the desired common subsequence. We describe how to generate a set of \( n + 1 \) nested linear graphs \( \mathcal{G} \), each of height at most 3, such that \( S \) has a common subsequence of length \( k \) if and only if \( \mathcal{G} \) has a common nested subgraph of size \( 3k \).

Since all linear graphs are nested, we may assume that they are given in the form of Dyck words over the alphabet \( \{a, b\} \). Let \( H_0, H_1 \) and \( H \) be three nested linear graphs defined by \( H_0 = \)}
a \( (ab)^2 \) b, \( H_1 = a^3b^3 \) and \( H = a^2 \ (ab)^2 \ b^2 \). Observe that both \( H_0 \) and \( H_1 \) occur in \( H \). For each string \( s_i = s_{i[1]} s_{i[2]} \ldots s_{i[m_i]} \) of \( S \) of length \( m_i \), add to \( \mathcal{G} \) the nested linear graph \( G_i = H_{s_{i[1]}} \circ H_{s_{i[2]}} \circ \ldots \circ H_{s_{i[m_i]}} \). Finally, add to \( \mathcal{G} \) the target nested linear graph \( G_0 = H_0 \). It is a simple matter to check that each graph of \( \mathcal{G} \) is a nested linear graph of height at most 3.

Suppose that \( S \) has a common subsequence \( u = u[1] u[2] \ldots u[k] \) of length \( k \). Consider the nested linear graph \( G_{\text{sol}} = H_{u[1]} \circ H_{u[2]} \circ \ldots \circ H_{u[k]} \). Clearly, \( G_{\text{sol}} \) has size \( 3k \). Furthermore, it may be easily verified that \( G_{\text{sol}} \) occurs in each nested linear graph \( G_i \in \mathcal{G} \).

Conversely, suppose that \( \mathcal{G} \) has a common nested linear graph \( G_{\text{sol}} \) of size \( 3k \). First, observe that \( H \) does not occur in any linear graph \( G_i \), \( 1 \leq i \leq n \). But \( G_0 = H_k \). Then it follows that \( G_{\text{sol}} \) is of the form \( G_{\text{sol}} = H_{i_1} \circ H_{i_2} \circ \ldots \circ H_{i_k} \) where \( i_j = 0 \) or \( i_j = 1 \) for \( 1 \leq j \leq k \), since both \( H_0 \) and \( H_1 \) have size 3. Hence the string \( i_1 \ i_2 \ldots i_k \) is a common subsequence of \( S \) of length \( k \).

**Proof (of Proposition 11).** We reduce from the Longest Common Subsequence problem parameterized by the number of strings, which is known to be \( \mathsf{W}[t] \)-hard for all \( t \geq 1 \) [5]. Let an instance of the Longest Common Subsequence problem be given by a set of string \( S = \{s_1, s_2, \ldots, s_n\} \) over alphabet \( \mathcal{A} \) and a positive integer \( k \) denoting the size of the desired common subsequence. For simplicity, we shall assume \( \mathcal{A} = \{1, 2, \ldots, c\} \), \( c > 1 \). We describe how to generate a set of \( n + 1 \) edge-independent linear graphs \( \mathcal{G} = \{G_0, G_1, \ldots, G_n\} \) of height at most 3 such that \( S \) has a common subsequence of length \( k \) if and only if \( \mathcal{G} \) has a common nested linear graph of size \( k(c+2) \).

We start by defining gadgets. For each \( 1 \leq i \leq c \), define the nested linear graph \( H_i \) as follows:

\[
V(H_i) = \{1, 2, \ldots, 2c + 4\} \\
E(H_i) = \{(j, j + 1) : j \in \{3, 5, \ldots, 2i + 1\}\} \cup \{(j, j + 1) : j \in \{2i + 4, 2i + 6, \ldots, 2c + 2\}\} \cup \{(1, 2i + 3), \{2, 2c + 4\}\}
\]

Also, define the edge-independent linear graph \( H \) as follows:

\[
V(H_0) = \{1, 2, \ldots, 4c + 2\} \\
E(H_0) = \{(j, j + 1) : j \in \{c + 2, c + 5, \ldots, 4c - 1\}\} \cup \{(j, 3(j - 1) + 6) : j \in \{2, 3, \ldots, c + 1\}\} \cup \{(1, 4c + 2)\}
\]

For the sake of clarity, we introduce the temporary notation \( \circ \) for concatenating linear graphs. Let \( G_1 \) and \( G_2 \) be two linear graphs with \( V(G_1) = \{1, 2, \ldots, n_1\} \) and \( V(G_2) = \{1, 2, \ldots, n_2\} \). The linear graph \( G_1 \circ G_2 \) is defined by

\[
V(G_1 \circ G_2) = \{1 + 2 + \ldots n_1 + n_2\} \\
E(G_1 \circ G_2) = E(G_1) \cup \{(i + n_1, j + n_1) : (i, j) \in E(G_2)\}
\]

We now turn defining the linear graphs of \( \mathcal{G} \). For each string \( s_i = s_{i[1]} s_{i[2]} \ldots s_{i[m_i]} \) of \( S \) of length \( m_i \), add to \( \mathcal{G} \) the nested linear graph \( G_i = H_{s_{i[1]}} \circ H_{s_{i[2]}} \circ \ldots \circ H_{s_{i[m_i]}} \). Finally, add to \( \mathcal{G} \) the target nested linear graph \( G_0 = H_0 \circ H_0 \circ \ldots \circ H_0 \), \( k \) times. It is a simple matter to check that each graph of \( \mathcal{G} \) is an edge-independent linear graph of height at most 3.
Suppose first that \( S \) has a common subsequence \( i_1 i_2 \ldots i_k \) of length \( k \). Consider the nested linear graph \( G_{\text{sol}} = H_{i_1} \odot H_{i_2} \odot \ldots \odot H_{i_k} \). Clearly, \( G_{\text{sol}} \) has size \( k(c+2) \). Furthermore, it may be easily verified that \( G_{\text{sol}} \) occurs in each \( G_i \in \mathcal{G} \).

Conversely, suppose that \( \mathcal{G} \) has a common nested linear graph \( G_{\text{sol}} \) of size \( k(2c+4) \). First observe that \( H_0 \) has size \( (4c+2) \) and is not nested. Furthermore, since \( H_0 \) has depth \( c \), one has to remove at least \( c-1 \) edges from \( H_0 \) to obtain a nested linear subgraph. But \( G_0 = H_0 \odot H_0 \odot \ldots \odot H_0 \), \( k \) times. Then, it follows that \( G_{\text{sol}} \) is of the form \( G_{\text{sol}} = H_{i_1} \odot H_{i_2} \odot \ldots \odot H_{i_k} \), and hence the string \( i_1 i_2 \ldots i_k \) is a common subsequence of \( S \) of length \( k \).

\[ \square \]

**Proof (of Proposition 12).** The proof is analogous to the proof of Proposition 11. It suffices to use the Binary Alphabet Longest Common Subsequence problem parameterized by the number of strings, which is known to be \( \textbf{W}[1]\)-hard [27], as a starting problem.

\[ \square \]

**Proof (of Proposition 13).** Again, the proof is analogous to the proof of Proposition 11. It suffices to consider the following linear graph \( H_0 \):

\[
\begin{align*}
V(H_0) &= \{1, 2, \ldots, 3c+3\} \\
E(H_0) &= \{(j, j+1) : j \in \{3, 6, \ldots, 3c\}\} \cup \\
&\quad \{(2, j) : j \in \{5, 8, \ldots, 3c+2\}\} \cup \\
&\quad \{(1, 3c+3)\}
\end{align*}
\]

\[ \square \]

**Proof (of Proposition 14).** We describe a reduction from the Longest Common Subsequence problem parameterized by the number of strings, which is known to be \( \textbf{W}[t]\)-hard for all \( t \geq 1 \) [5]. Let an instance of the Longest Common Subsequence problem be given by a set of string \( S = \{s_1, s_2, \ldots, s_n\} \) over alphabet \( A = \{1, 2, \ldots, c\} \) and a positive integer \( k \) denoting the size of the desired common subsequence. We describe how to generate a set of \( n+1 \) nested linear graphs \( \mathcal{G} = \{G_0, G_1, \ldots, G_n\} \) of height at most 4 such that \( S \) has a common subsequence of length \( k \) if and only if \( \mathcal{G} \) has a common nested linear graph \( k(2c+1) \).

Since all linear graphs are nested, we may assume that they are given in the form of Dyck words over the alphabet \( \{a, b\} \). For each \( i \in A = \{1, 2, \ldots, c\} \), consider the nested linear graph \( H_i \) defined by \( G_i = a^2(a^2b^2)^{c-1}ab(a^2b^2)^{c-1}b^2 \). Furthermore, define the nested linear graph \( H_0 \) as follows:

\[
H_0 = a^2(a^2b^2)^{c-1}b^2.
\]

Now, corresponding to each each string \( s_i \in S \) of length \( m_i \), construct the nested linear graph \( G_i = H_{s_i[1]} H_{s_i[2]} \ldots H_{s_i[m_i]} \). Finally, add to \( \mathcal{G} \) the nested linear graph \( G_0 = H_0^k \). It is easily seen that \( \mathcal{G} \) is composed of nested linear graphs of height at most 4.

Suppose that \( S \) has a common subsequence \( i_1 i_2 \ldots i_k \) of length \( k \). Consider the nested linear graph \( G_{\text{sol}} = H_{i_1} H_{i_2} \ldots H_{i_k} \). Clearly, \( G_{\text{sol}} \) has size \( k(2c+1) \). Furthermore, it may be easily verified that \( G_{\text{sol}} \) occurs in each \( G_i \in \mathcal{G} \).

Conversely, suppose that \( \mathcal{G} \) has a common nested linear graph \( G_{\text{sol}} \) of size \( k(2c+1) \). First, observe that \( H_0 \) does not occur in any linear graph \( G_i \in \mathcal{G} \). Furthermore, \( G_0 \) has size \( k(2c+2) \).
Thus $G_{\text{sol}}$ is of the form $G_{\text{sol}} = H_{i_1}H_{i_2}\ldots H_{i_k}$, where each $i_k \in A$ for $1 \leq i \leq k$. But, $a(a^2b^2)c\ b$ does not occur in any $H_i$, $1 \leq i \leq c$, and hence no $H_i$, $1 \leq i \leq k$, could be obtained from $H_0$ by deleting the first or the second occurrence of $a$ (together with the corresponding $b$). Then it follows that there exists an ordered collection $(i_1, i_2, \ldots, i_k)$, $1 \leq i_j \leq c$ for $1 \leq j \leq k$, such that $G_{\text{sol}} = H_{i_1}H_{i_2}\ldots H_{i_k}$, where each $i_k \in A$ for $1 \leq i \leq k$. Since $G_{\text{sol}}$ occurs in each $G_i \in \mathcal{G}$, we thus conclude that $i_1i_2\ldots i_k$ is a common subsequence of $S$ of length $k$. \qed