Approximation of RNA Multiple Structural Alignment*

Marcin Kubica[1], Romeo Rizzi[2], Stéphane Vialette[3], and Tomasz Waleń[1]

[1] Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warszawa, Poland
{kubica, walen}@mimuw.edu.pl
[2] Dipartimento di Matematica ed Informatica (DIMI),
Università di Udine, Via delle Scienze 208, I-33100 Udine, Italy
Romeo.Rizzi@dimi.uniud.it
[3] 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 [7] the problem of finding the largest nested linear graph that occurs in a set $\mathcal{G}$ of linear graphs, the so-called Max-NLS problem. This problem generalizes both the *longest common subsequence* problem and the *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 a fixed number (k) of linear graphs. Also, we strongly strengthen the result of [7] by proving that the problem is **NP**-complete even if $\mathcal{G}$ 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 [7] by showing that the Max-NLS problem is approximable within ratio $O(\log m_{opt})$ in $O(kn^2)$ running time, where m_{opt} is the size of an optimal solution. We also present $\mathcal{O}(1)$-approximation of Max-NLS problem running in $\mathcal{O}(kn)$ time for restricted linear graphs. In particular, for ncRNA derived linear graphs, an $\frac{1}{4}$-approximation is presented.

1 Introduction

Non-coding RNA, unlike regular genes, are not translated to proteins, but perform a variety of catalytic, structural and regulatory functions; transfer RNA, ribosomal RNA and spliceosomal RNA are textbook examples. The RNA sequences consist of four kinds of nucleotides: A (adenine), C (cytosine), G (guanine) and U (uracil). RNA sequences tend to fold, forming secondary and tertiary

* This research was partially supported by the Polish Scientific Research Committee (KBN) under grant GR-1946 and by the French-Italian Galileo Project PAI 08484VH.

M. Lewenstein and G. Valiente (Eds.): CPM 2006, LNCS 4009, pp. 212–223, 2006.
© Springer-Verlag Berlin Heidelberg 2006

structures, stabilized by bonds between nucleotides. Three kinds of such bonds are most frequent: A–U, G–C and U–G.

The structural stability and function of ncRNA genes are largely determined by the formation of stable secondary structures through complementary bases (see [18] for a detailed introduction to RNA secondary structures). Much research work has been done on the structural comparison of ncRNA sequences [2, 3, 4, 7, 13, 14]. Davydov and Batzoglou proposed in [7] a new model for structural alignment of multiple ncRNA sequences, based on finding the largest common secondary structure. The problem of computing the largest common secondary structure for the given set of k ncRNA sequences can be solved using the stochastic context-free grammars, in $O(n^{3k})$ time complexity [12].

In this paper, we focus on the problem of secondary structure alignment for multiple ncRNA sequences. We follow a general model presented in [7], where ncRNA sequences are modelled by linear graphs and the common secondary structure is modelled using the maximum common nested linear subgraphs (Max-NLS problem). Unfortunately this problem is NP-complete, but can be approximated within $O(\log^2 n)$ factor in $O(k \cdot n^5)$ running time [7]. The authors proposed to approximate Max-NLS with maximum level linear subgraphs (Max-LLS problem).

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 a fixed number of linear graphs. Section 5 strongly refines the hardness result of [7] by giving a tight description of the borderline between **NP**-completeness and **P**. We present in Section 6 a faster algorithm for the Max-NLS problem and in Section 7 we improve the approximation ratio of [7]. Finally, Section 8 deals with the Max-NLS problem, for restricted linear graphs. 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 $\mathbf{V}(G)$ the set of vertices and by $\mathbf{E}(G)$ the set of edges. The *order* and the *size* of G stand for $|\mathbf{V}(G)|$ and $|\mathbf{E}(G)|$, respectively. A *linear graph* of order n is a vertex-labeled graph where each vertex is labeled by a distinct integer from $\{1, 2, \ldots, n\}$. 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 [17]. Let $e = (i, j)$ and $e' = (i', j')$ be two independent edges in a linear graph G. We write (i) $e < e'$ if $i < j < i' < j'$, (ii) $e \sqsubset e'$ if $i' < i < j < j'$, and (iii) $e \between e'$ if $i < i' < j < j'$. Two edges e and e' are said to be R-comparable for some $\mathrm{R} \in \{<, \sqsubset, \between\}$ if $e \,\mathrm{R}\, e'$ or $e' \,\mathrm{R}\, e$. Observe, that any two independent edges are R-comparable for some $\mathrm{R} \in \{<, \sqsubset, \between\}$. An edge-independent linear graph

G is called an *$\mathcal{R}$-comparable linear graph* for some non-empty $\mathcal{R} \subseteq \{<, \sqsubset, \between\}$ if any two distinct edges in G are R-comparable for some $\mathrm{R} \in \mathcal{R}$. Let G be a linear graph. The *width* (resp. *height*) of G is the size of a maximum cardinality $\{<\}$-comparable (resp. $\{\sqsubset\}$-comparable) subset of $\mathbf{E}(G)$.

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 (or G_1 is called a *subgraph* of G_2) if one can obtain G_1 from G_2 (regardless of precise vertex labels) by a sequence of edge and vertex deletions. More formally, the deletion of vertex i consists in (1) the deletion of all the edges incident to vertex i, (2) the deletion of vertex i and (3) the relabeling of all vertices $j > i$ to $j - 1$.

For the purpose of ncRNA multiple structural alignment, convenient graphs are needed [7]. A linear graph is *non-crossing* if it does not contain two edges e and e' such that $e \between e'$. A $\{<, \sqsubset\}$-comparable linear graph is also called a *nested linear graph*. A $\{\sqsubset\}$-comparable subgraph of a linear graph G is called a *nested loop*. Let (i, j) be the outermost edge of a nested loop in G. Value $j - i$ is called the *diameter* of the nested loop.

Nested linear graphs can be viewed as collections of trees. Each edge in the nested linear graph corresponds to a node in a tree — e is the parent of e' iff $e' \sqsubset e$ and there is no such edge e'' that $e' \sqsubset e'' \sqsubset e$. Each $\sqsubset$-maximal edge in the nested linear graph is a root of some tree.

To shorten notation, a nested linear graph G of size n can be represented by a Dyck word of semi-length n over the alphabet $\mathcal{A} = \{a, b\}$. By abuse of notation, we continue to write G for the corresponding Dyck word. A nested linear graph G is *flat* if it can be written as $G = a^{h_1} b^{h_1} \, a^{h_2} b^{h_2} \, \ldots \, a^{h_k} b^{h_k}$ for some positive integers $h_1, h_2, \ldots, h_k$. A flat linear graph is called *level* if it can be written as $G = (a^h b^h)^w$ for some positive integers h and w.

For a given linear graph G we can consider its level subgraphs of given height and maximum width. The relation between the height and the maximum width of level subgraphs is called *level signature* of the linear graph. More formally, level signature of G is a function $s : \mathbb{N} \to \mathbb{N}$ such, that: (i) $s(h)$ is the maximum width of a level subgraph of G with height h; (ii) if G has no level subgraph of height h, then $s(h) = 0$.

Fig. 1. Maximum level subgraphs of G with height 2 (on the left), and height 3 (on the right). The level signature of the graph is: $s(1) = 5$, $s(2) = 4$, $s(3) = 3$, $s(4) = 0$.

Genomic sequences can be naturally viewed as linear graphs. A sequence of nucleotides $S = (a_1, a_2, \ldots, a_n)$ corresponds to a linear graph whose vertices are the nucleotides and there is an edge between two nucleotides iff there can be a bond between them. In this paper we also investigate more abstract correspondence. Let $S = (a_1, a_2, \ldots, a_n)$ be a sequence over some fixed finite alphabet Σ, and let $\xi \subseteq \Sigma^2$ be a fixed **symmetrical** relation. A linear

graph *derived* from S, denoted as $\mathbf{G}_\xi(S)$, is such a linear graph of order n, in which there is an edge (p, q) iff $p \neq q$ and $a_p \, \xi \, a_q$. Throughout this paper we assume that Σ and ξ are fixed. We will call linear graphs derived from sequences over Σ, *restricted linear graphs* (RLG). Clearly, for $\Sigma_{\mathsf{RNA}} = \{\mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C}\}$ and $\xi_{\mathsf{RNA}} = \{(\mathrm{A}, \mathrm{U}), (\mathrm{U}, \mathrm{A}), (\mathrm{U}, \mathrm{G}), (\mathrm{G}, \mathrm{U}), (\mathrm{G}, \mathrm{C}), (\mathrm{C}, \mathrm{G})\}$ we get RLG modelling ncRNA sequences.

Let $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ be a set of linear graphs. The Max-NLS problem is to find a maximum size common nested linear subgraph of $G_i \in \mathcal{G}$. We denote it by Max-NLS$(G_1, \ldots, G_k)$. Nested linear subgraphs and Max-NLS problem have been introduced in [7] to represent possible structural alignments of ncRNA. Nested linear subgraphs correspond to different structural alignments of a sequence of nucleotides, and the edges of a nested linear subgraph correspond to bonds stabilising the structure of ncRNA, as shown in fig. 2.

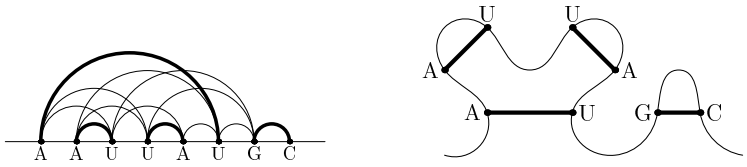

Fig. 2. Example of linear graph for sequence AAUUAUGC, its Max-NLS and corresponding bonds between nucleotides

The Max-LLS problem is to find a maximum size common level linear subgraph of $G_1, \ldots, G_k$ (denoted by Max-LLS$(G_1, \ldots, G_k)$). Level linear subgraphs have been used to approximate Max-NLS problem in [7]. The MNL(G) problem is to find a maximum size nested loop in G. Nested loops and the MNL problem have been used to model ncRNA structural alignments in [4]. The authors also present $\mathcal{O}(n^2)$ algorithm for computing MNL.

3 Finding a Maximum Size Nested Linear Graph in a Linear Graph

Felsner *et al.* considered in [8] 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 $m = |\mathbf{E}(G)|$, a maximum size nested subgraph of G can be found in $\mathcal{O}(m^2)$ time. On the other hand the Max-NLS(G) can be computed in $\mathcal{O}(n^3)$ time [16], where $n = |\mathbf{V}(G)|$.

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 $\mathcal{O}(n^2 + nm)$ time and $\mathcal{O}(n^2)$ space. 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 $i, \ldots, j$.

We denote by $N^-(j)$ the set of vertices $N^-(j) = \{q : q < j \wedge (q,j) \in \mathbf{E}(G)\}$. For $1 \leq i < j \leq n$, let $\mathsf{opt}[i,j]$ denote the maximum size of a nested subgraph of the linear graph $G_{i,j}$ (for $i \geq j$ we have $\mathsf{opt}[i,j] = 0$). For $1 \leq i < j \leq n$, $\mathsf{opt}[i,j]$ can be obtained using the following formula:

$$\mathsf{opt}[i,j] = \max \begin{cases} \mathsf{opt}[i,j-1], \\ 1 + \mathsf{opt}[i,q-1] + \mathsf{opt}[q+1,j-1] : q \in N^-(j) \ \wedge \ q \geq i \end{cases}$$

The maximum nested subgraph of $G_{i,j}$ either does not not take vertex j, in which case $\mathsf{opt}[i,j] = \mathsf{opt}[i,j-1]$, or it takes an edge $(q,j) \in \mathbf{E}(G)$ with $i \leq q < j$.

Using dynamic programming, the array opt can be computed in $O(n^2 + nm)$ time, since computation of each $\mathsf{opt}[i,j]$ requires $O(|N^-(j)|)$ steps.

4 A Polynomial-Time Algorithm for Fixed $|\mathcal{G}|$

According to [11], 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 $\mathcal{O}(m_2 \log m_2 + m_1 m_2)$ time. Valiente proposed in [15] a dynamic programing algorithm for finding the largest nested linear graph that occurs in two nested linear graphs (see also [19]). In this section, we give a $\mathcal{O}(m^{2k} \log^{k-2} m^k \log\log m^k)$ time dynamic programming algorithm, where $m = \max\{|\mathbf{E}(G_i)| : G_i \in \mathcal{G}\}$ and $k = |\mathcal{G}|$, for finding the largest nested linear graph that occurs in a fixed number of linear graphs.

We need some additional definitions and notations. We use the notions of *trapezoid diagrams* and *d-trapezoid diagrams* introduced in [6] and [9], 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* [8,5] if there exist families of intervals $T_u = \{I_u^i = [l_u^i, r_u^i] : l_u^i, r_u^i \in L^i, 1 \leq i \leq d+1\}$, $u \in \mathbf{V}(G)$, satisfying $\{u,v\} \in \mathbf{E}(G)$ if and only if $Q_u \cap Q_v \neq \emptyset$, where Q_x denotes the closed polygon $(l_x^1, l_x^2, \ldots, l_x^{d+1}, r_x^{d+1}, r_x^d, \ldots, r_x^1)$. We refer to the family $\mathcal{T}_G = \{T_u : u \in \mathbf{V}(G)\}$ to as the *d-trapezoid diagram* of G. Note that 0-trapezoid graphs are precisely interval graphs [10], 1-trapezoid graphs are the usual trapezoid graphs [6], and d-trapezoid graphs are comparability graphs of posets with interval dimension at most $d+1$ [9].

Based on a geometric representation of 1-trapezoid graphs by boxes in the plane, Felsner *et al.* [8] designed an optimal $\mathcal{O}(n \log n)$ 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 $\mathcal{O}(n \log^d n)$ time algorithm for d-trapezoid graphs of order n. This has been improved in [1] to $\mathcal{O}(n \log^{d-1} n \log\log n)$ time. We now turn to defining an irreflexive, transitive and anti-symmetric relation $\sqsubset$ on $\mathcal{T}_G$. Let G be a d-trapezoid graph and $\mathcal{T}_G = \{T_u : u \in \mathbf{V}(G)\}$ be the corresponding d-trapezoid diagram. Let $T_u = \{I_u^i = [l_u^i, r_u^i] : l_u^i, r_u^i \in L^i, 1 \leq i \leq d+1\}$ and $T_v = \{\mathcal{I}_v^i = [l_v^i, r_v^i] : l_v^i, r_v^i \in L^i, 1 \leq i \leq d+1\}$ 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 $l_v^i < l_u^i$ and $r_u^i < r_v^i$ for all $1 \leq i \leq d+1$.

Let $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ be an instance of the Max-NLS problem. We associate to $\mathcal{G}$ an $(k-1)$-trapezoid diagram as follows. For each $1 \leq i \leq k$, the graph G_i is associated to Line L^i, and for each $(x, y) \in \mathbf{E}(G_i)$ we define an interval $I^i_{x,y} = [x, y]$ 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.*, $\mathcal{I}_{G_i} = \{I^i_{x,y} : (x, y) \in \mathbf{E}(G_i)\}$. The $(k-1)$-*trapezoid diagram induced by* $\mathcal{G}$, written $\mathcal{T}[\mathcal{G}]$, is defined as follows:

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

Clearly, $|\mathcal{T}[\mathcal{G}]| = \prod_{G_i \in \mathcal{G}} |\mathbf{E}(G_i)|$. Having disposed of these preliminaries, we now turn to presenting our algorithm, referred hereafter to as Algorithm nested-linear-subgraph, for finding the largest nested linear graph G that occurs in a family of linear graphs $\mathcal{G}$. The basic idea is to associate to each $(k-1)$-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 $(k-1)$-trapezoids strictly included in T. This is done in turn by dynamic programming according to a linear extension of $(\mathcal{T}[\mathcal{G}], \sqsubset)$. We need the following subroutine: Given an $(n-1)$-trapezoid diagram $\mathcal{T}$ and a function $\omega : \mathcal{T} \to \mathbb{N}^+$, we refer to the algorithm for finding a maximum weighted disjoint subset of $\mathcal{T}$ (in terms of disjoint induced closed polygons) as max-weighted-independent-set$(\mathcal{T}, \omega)$ [8, 1]. A more schematic description of Algorithm nested-linear-subgraph$(\mathcal{G} = \{G_1, G_2, \ldots, G_k\})$ is given below:

```
1  begin
2  |  Compute the (k − 1)-trapezoid diagram T[G] induced by G
3  |  Compute any linear extension Φ of (T[G], ⊏)
4  |  foreach T ∈ T[G] do ω(T) := 0
5  |  foreach T ∈ T[G] with respect to Φ do
6  |  |  T' := {T' : T' ⊏ T}.
7  |  |  T'' := max-weighted-independent-set(T', ω)
8  |  |  ω(T) := 1 + Σ_{T''∈T''} ω(T'')
9  |  end
10 |  T* := max-weighted-independent-set(T[G], ω)
11 |  return Σ_{T∈T*} ω(T)
12 end
```

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

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 [13] (only two nested arc-annotated sequences, and hence two linear graphs here).

5 Hardness Results

It is proved in [7] 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 $|\mathcal{G}|$ associated interval graphs and return the maximum stability found). Our result is a two-step procedure. We begin by proving the **NP**-hardness of a new list problem, *i.e.*, the Longest Common Sublist problem. Next, we give a polynomial-time reduction from the Longest Common Sublist problem to prove that the Max-NLS problem is **NP**-complete even when restricted to simple instances solely composed of flat linear graphs of height at most 2.

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

Longest Common Sublist

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

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

Proposition 2. *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 3. *The* Max-NLS *problem for flat linear graphs of height at most* 2 *is* **NP***-complete.*

Proof. 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 $L_1, L_2, \ldots, L_k$, and by a positive integer m. We construct a corresponding set $\mathcal{G}$ of k linear graphs as follows (we abbreviate in a natural way a nested linear graph G of size m to a Dyck word of semi-length m over the alphabet $\mathcal{A} = \{a, b\}$). For each list L_i, $1 \leq i \leq k$, we add to $\mathcal{G}$ the nested linear graph G_i defined by $G_i = a^{L_i[1]} b^{L_i[1]}\ a^{L_i[2]} b^{L_i[2]}\ \ldots\ a^{L_i[\mathsf{len}(L_i)]} b^{L_i[\mathsf{len}(L_i)]}$.. It is easily seen that $\mathcal{G}$ is composed of k 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 $\tilde{L}_1, \tilde{L}_2, \ldots, \tilde{L}_k$, where $\tilde{L}_i$ is a sublist of L_i, $1 \leq i \leq k$, and $\mathsf{len}(\tilde{L}_1) = \mathsf{len}(\tilde{L}_2) = \ldots = \mathsf{len}(\tilde{L}_k) = \ell$, such that $\sum_{1 \leq j \leq \ell} \min\{\tilde{L}_i[j] : 1 \leq i \leq k\} \geq m$. Let $\tilde{L}$ be the list of length ℓ defined by $\tilde{L}[j] = \min\{\tilde{L}_i[j] : 1 \leq i \leq k\}$ for $1 \leq j \leq \ell$. Now, consider the flat linear graph G_{sol} defined by $G_{\mathrm{sol}} = a^{\tilde{L}[1]}b^{\tilde{L}[1]}\ a^{\tilde{L}[2]}b^{\tilde{L}[2]}\ \ldots\ a^{\tilde{L}[\ell]}b^{\tilde{L}[\ell]}$. It can be easily verified that G_{sol} is a flat linear graph of size $\sum_{1 \leq j \leq \ell} \min\{\tilde{L}_i[j] : 1 \leq i \leq n\} \geq m$ that occurs in each $G_i \in \mathcal{G}$.

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

Here below we offer a reformulation of Proposition 3 that may be of independent interest.

Proposition 4. *Finding the largest common homeomorphic subtree in a set of ordered rooted trees of height at most* 3 *is an* **NP***-complete problem.*

6 Max-LLS Problem

Max-LLS problem was defined in [7], where it was used to approximate Max-NLS. The algorithm proposed in [7] to solve Max-NLS problem for k linear graphs of order n has $\mathcal{O}(k \cdot n^5)$ time complexity. In this section, we present an algorithm solving this problem in $\mathcal{O}(k \cdot n^2)$ time. For this and the following section, let $G_1, \ldots, G_k$ be given linear graphs of order n. First, for each G_i we compute its level signature. Then we compute minimum of k level signatures. This minimum gives us the shape of Max-LLS$(G_1, G_2, \ldots, G_k)$. Knowing its shape, we can then reconstruct it.

For each G_i its level signature is computed in three steps. First, we compute an array MNL_i describing sizes of maximum nested loops for all fragments of G_i. Let $G'_{i,p,q}$ be a linear subgraph of G_i induced by vertices $p, \ldots, q$. For $1 \leq p < q \leq n$, $\mathsf{MNL}_i[p,q]$ is the size of $\mathsf{MNL}(G'_{i,p,q})$ (for $p \geq q$ we have $\mathsf{MNL}_i[p,q] = 0$). If $(p,q) \in \mathbf{E}(G_i)$ then $\mathsf{MNL}_i[p,q] = \max(\mathsf{MNL}_i[p{+}1,q],\ \mathsf{MNL}_i[p,q{-}1], \mathsf{MNL}_i[p{+}1,q{-}1]{+}1)$, otherwise $\mathsf{MNL}_i[p,q] = \max(\mathsf{MNL}_i[p+1,q],\ \mathsf{MNL}_i[p,q-1])$. Clearly, MNL_i can be computed in $\mathcal{O}(n^2)$ time using dynamic programming.

Next, we compute an array[4] NLW_i containing minimum diameter of nested loops of given size starting at given position. For $1 \leq p \leq n$ and $1 \leq h \leq \frac{n}{2}$, $\mathsf{NLW}_i[p,h]$ is the minimum such integer, that $v_{i,p}, \ldots, v_{i,p+\mathsf{NLW}_i[p,h]}$ contains a nested loop of size h (or 0 if such an integer does not exist).

[4] We assume that all arrays are implicitly initialized with zeros.

```
for h = 1 to ⌊n/2⌋ do
    for p = n − 1 downto 1 do
        if NLW_i[p+1,h] > 0 then NLW_i[p,h] = NLW_i[p+1,h] + 1
        else if MNL_i[p,n] ≥ h then NLW_i[p,h] = n − p;
        while NLW_i[p,h] > 1 ∧ MNL_i[p, p + NLW_i[p,h] − 1] ≥ h do
            NLW_i[p,h] = NLW_i[p,h] − 1
        end
    end
end
```

Let us have a closer look at the inner `while` loop. Please note that, for given p and h, if $\mathsf{NLW}_i[p,h] > 0$ and $\mathsf{NLW}_i[p+1,h] > 0$ then the `while` loop performed $\mathsf{NLW}_i[p+1,h] - \mathsf{NLW}_i[p,h] + 1$ iterations. If $\mathsf{NLW}_i[p,h] > 0$ and $\mathsf{NLW}_i[p+1,h] = 0$ then the `while` loop performed $n - p - \mathsf{NLW}_i[p,h]$ iterations. Otherwise $\mathsf{NLW}_i[p,h] = 0$ and the `while` loop performed no iterations. Hence, the total number of iterations made by this loop for given h is not greater than $n - 1 - \mathsf{NLW}_i(1,h) \leq n$. Therefore, the running time of this step is $\mathcal{O}(n^2)$.

Finally, for $1 \leq p \leq n$, we compute as $\mathsf{SIG}_i[p,h]$ the level signature of $G'_{i,p,n}$ for height h. It can be done in $\mathcal{O}(n^2)$ time, using dynamic programming and the following formula:

$$\mathsf{SIG}_i[p,h] = \begin{cases} \mathsf{SIG}_i[p + \mathsf{NLW}_i[p,h] + 1, h] + 1, & \text{if } \mathsf{NLW}_i[p,h] > 0 \\ 0 & \text{otherwise} \end{cases}$$

Obviously, the level signature of G_i for height h is in $\mathsf{SIG}_i[1,h]$ and the total time complexity of all three steps is $\mathcal{O}(n^2)$. The level signature SIG of common level subgraphs of $G_1, G_2, \ldots, G_k$ equals $\mathsf{SIG}[h] = \min_{i=1,\ldots,k} \mathsf{SIG}_i[1,h]$, and the size of the maximum common level subgraph is equal $\max_{h=1,\ldots,\lfloor \frac{n}{2} \rfloor} h \cdot \mathsf{SIG}[h]$. The actual maximum common level subgraph can be reconstructed basing on its height and from arrays SIG_i, NLW_i and MNL_i, in $\mathcal{O}(kn)$ time.

7 Approximation of Max-NLS

In [7], the authors prove that the optimal solution for Max-LLS problem gives $\mathcal{O}(\log^2 m_{opt})$-approximation for Max-NLS problem (where m_{opt} is the size of the optimal solution for Max-NLS). The proof from [7] consists of two steps: first Max-NLS problem is reduced to the problem of finding a maximum flat linear subgraph, and then it is further reduced to Max-LLS. In this section, we show how to reduce Max-NLS problem to Max-LLS directly, achieving $\mathcal{O}(\log m_{opt})$ approximation ratio. Please note, that we study here exactly the same approximation, but we prove a better approximation ratio.

Theorem 1. *Let $G_1, \ldots, G_k$ be given linear graphs of order n, m_{opt} and h_{opt} be respectively the size and the height of* Max-NLS$(G_1, \ldots, G_k)$*, and l be the size of* Max-LLS$(G_1, \ldots, G_k)$*. Then we have: $m_{opt} \leq \Theta(\log h_{opt}) \cdot l \leq \Theta(\log m_{opt}) \cdot l$.*

Proof. Let T be a collection of trees representing Max-NLS$(G_1, \ldots, G_k)$. Obviously, T contains m_{opt} nodes. For each such node v, by $l(v)$ we will denote the height of the subtree rooted in v, and by $\mathsf{path}(v)$ we will denote a fixed path of length $l(v)$ starting in v and ending in some leaf of the subtree rooted in v. For all leaves we have $l(v) = 1$. Please note, that $\mathsf{path}(v)$ represents a nested loop of $l(v)$ edges. By $L(h)$ we will denote the set of nodes in T of height h, $L(h) = \{v : l(v) = h\}$. Clearly, $\sum_{i=1}^{h_{opt}} |L(i)| = m_{opt}$. We also define $S(h) = \{\mathsf{path}(v) : v \in L(h)\}$. Let us observe that $S(h)$ represents a set of nested loops that form a level linear subgraph with width $|L(h)|$, height h and size $s(h) = h \cdot |L(h)|$. Let h_{max} be such a height, for which $s(h)$ is maximum. Clearly, $s(h_{\mathsf{max}}) \leq l$. For any $i = 1, \ldots, h_{opt}$, we have $|L(i)| \leq \frac{s(h_{\mathsf{max}})}{i} \leq \frac{l}{i}$.

$$m_{opt} = \sum_{i=1}^{h_{opt}} |L(i)| \leq \sum_{i=1}^{h_{opt}} \frac{l}{i} = l \cdot \sum_{1}^{h_{opt}} \frac{1}{i} \leq \Theta(\log h_{opt}) \cdot l \leq \Theta(\log m_{opt}) \cdot l$$

□

This bound is asymptotically tight. The family of trees defined in [7] gives $\Theta(\log m_{opt})$ approximation factor.

8 Max-NLS Problem for Restricted Linear Graphs

In this section we deal with Max-NLS problem, for restricted linear graphs. We will show for this problem an $\mathcal{O}(1)$-approximation algorithm running in $\mathcal{O}(kn)$ time. For this section, let $S = (a_1, \ldots, a_n) \in \Sigma^n$ be a given sequence of characters, and $\#_p(X)$ denote the number of characters p in sequence X.

For $p, q \in \Sigma$, by $\mathsf{MNL}_{(p,q)}(S)$ we will denote MNL of a subgraph of $\mathbf{G}_\xi(S)$ containing only such edges (i, j), that $a_i = p$ and $a_j = q$. In other words, we focus only on edges whose left endpoints are at characters p and right endpoints are at characters q. By $\mathsf{MNL}_\xi(S)$ we will denote the maximum nested loop among all $\mathsf{MNL}_{(p,q)}(S)$ for $(p, q) \in \xi$.

Theorem 2. $\mathsf{MNL}_\xi(S)$ *can be computed in* $\mathcal{O}(n)$ *time.*

Proof. The following two arrays: $l[i, p] = \#_p(a_1, \ldots, a_i)$, $r[i, p] = \#_p(a_i, \ldots, a_n)$ (for $1 \leq i \leq n$, $p \in \Sigma$) can be computed in $\mathcal{O}(n)$ time. The $|\mathbf{E}(\mathsf{MNL}_\xi(S))|$ can be obtained using the following formula:

$$|\mathbf{E}(\mathsf{MNL}_\xi(S))| = \max_{i \in 1, \ldots, n-1,\ (p,q) \in \xi} \min(l[i, p], r[i + 1, q]).$$

Since the size of ξ is $\mathcal{O}(1)$, the total running time of this algorithm is $\mathcal{O}(n)$. □

Lemma 1. *Let* $(p, q) \in \xi$. *If* $\#_p(S) \geq l$ *and* $\#_q(S) \geq l$ *then* $|\mathbf{E}(\mathsf{MNL}_\xi(S))| \geq \frac{l}{2}$.

Proof. One can note, that there exists a $1 \leq i \leq n-1$ such that $\#_p(a_1, \ldots, a_i) = \#_q(a_{i+1}, \ldots, a_n) = c$. There are two cases:

- If $c \geq \frac{l}{2}$, then obviously $|\mathbf{E}(\mathsf{MNL}_{(p,q)})| \geq c \geq \frac{l}{2}$.

– If $c < \frac{l}{2}$, then we have $\#_q(a_1, \ldots, a_i) \geq l - c$ and $\#_p(a_{i+1}, \ldots, a_n) \geq l - c$. Hence $|\mathbf{E}(\mathsf{MNL}_{(q,p)})| \geq l - c \geq \frac{l}{2}$. □

Theorem 3. *Let d be the number of such unordered pairs $\{p, q\}$, that $(p, q) \in \xi$. For given k sequences $S_i \in \Sigma^n$, $\mathsf{Max\text{-}NLS}(\mathbf{G}_\xi(S_1), \ldots, \mathbf{G}_\xi(S_k))$ can be approximated using MNL_ξ within factor $\frac{1}{2d}$ and in $\mathcal{O}(kn)$ time complexity.*

Proof. For each S_i we can calculate $\mathsf{MNL}_\xi(S_i)$ in $\mathcal{O}(n)$ time. Then we choose such $1 \leq j \leq k$ for which $|\mathbf{E}(\mathsf{MNL}_\xi(S_j))|$ is minimum and $\mathsf{MNL}_\xi(S_j)$ is our approximation. Clearly $\mathsf{MNL}_\xi(S_j)$ is a common subgraph of $\mathbf{G}_\xi(S_1), \ldots, \mathbf{G}_\xi(S_k)$, and the overall complexity of this algorithm is $\mathcal{O}(kn)$.

Let $S_j = (a_1, \ldots, a_n)$ and let $e = |\mathbf{E}(\mathsf{Max\text{-}NLS}(\mathbf{G}_\xi(S_j)))|$. It is enough to show, that $|\mathbf{E}(\mathsf{MNL}_\xi(S_j))| \geq \frac{e}{2d}$. We label each edge (r, s) in $\mathsf{Max\text{-}NLS}(\mathbf{G}_\xi(S_j))$ with an unordered pair $\{a_r, a_s\}$. Since there are at most d different labels on e edges, there exists such a pair of characters $(p, q) \in \xi$, that there are at least $\frac{e}{d}$ edges labeled with $\{p, q\}$. Hence $\#_p(S_j) \geq \frac{e}{d}$ and $\#_q(S_j) \geq \frac{e}{d}$. Using lemma 1, we have that $|\mathbf{E}(\mathsf{MNL}_\xi(S_j))| \geq \frac{e}{2d}$. □

It also proves that the approach presented in [7], where general linear graphs were used to model structural alignment of ncRNA, is too abstract. The set of nucleotides is definitely finite. Taking this into account can lead to faster algorithms or better approximation ratios.

For the ncRNA sequences, from the theorem 3 we can achieve an approximation ratio $\frac{1}{6}$ (since there are three possible kinds of unordered bonds in ξ_{RNA}). However, the lower bound on the approximation ratio can be improved, using properties of relation ξ_{RNA}, to $\frac{1}{4}$.

Proposition 5. *For given k sequences $S_i \in \Sigma^n_{\mathsf{RNA}}$, $\mathsf{Max\text{-}NLS}(\mathbf{G}_{\xi_{\mathsf{RNA}}}(S_1), \ldots, \mathbf{G}_{\xi_{\mathsf{RNA}}}(S_k))$ can be approximated using MNL within factor $\frac{1}{4}$.*

9 Conclusion

In this paper, we have investigated the problem of structural alignment of multiple ncRNA sequences. The problem have been modeled in the graph theoretical framework, where ncRNA correspond to linear graphs and their structural alignments correspond to nested linear subgraphs. We described a polynomial-time algorithm for fixed k, and gave improved approximability results, $\mathcal{O}(\log n)$-approximation algorithm running in $\mathcal{O}(kn^2)$ time for arbitrary linear graphs and $\mathcal{O}(1)$-approximation of Max-NLS problem running in $\mathcal{O}(kn)$ time for restricted linear graphs. In particular, for ncRNA derived linear graphs, an $\frac{1}{4}$-approximation is presented.

In conclusion, we mention some interesting directions for future works. First, the approximation aspect of the Max-NLS problem has to be improved. In particular, is the Max-NLS problem approximable to within some constant? Second, 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

1. M.I. Abouelhoda and E. Ohlebusch:. Multiple genome alignment: Chaining algorithms revisited. In *Proc. of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 2676 of *LNCS*, pages 1–16, 2003.
2. V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. Number 937, pages 1–16. Springer-Verlag, Berlin, 1995.
3. Vineet Bafna, Haixu Tang, and Shaojie Zhang. Consensus folding of unaligned rna sequences revisited. *RECOMB*, 2005.
4. Sergey Bereg and Binhai Zhu. RNA multiple structural alignment with longest common subsequences. *COCOON*, 3595:32–41, 2005.
5. H.L. Bodlaender, T. Kloks, D. Kratsch, and H. Müller. Treewidth and minimum fill-in on d-trapezoid graphs. *Journal of Graph Algorithms and Applications*, 2(5): 1–23, 1998.
6. I. Dagan, M.C. Golumbic, and R.Y. Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21:35–46, 1988.
7. E. Davydov and S. Batzoglou. A computational model for RNA multiple structural alignment. In *Proc. of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, LNCS, pages 254–269. Springer-Verlag, 2004.
8. S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations: Geometry and algorithms. *Discrete Applied Math.*, 74:13–32, 1997.
9. C. Flotow. on powers of m-trapezoid graphs. *Discrete Applied Mathematics*, 63(2):187–192, 1995.
10. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.
11. J. Gramm, J. Guo, and R. Niedermeier. Pattern matching for arc-annotated sequences. In *Proc. of the the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2556 of *LNCS*, pages 182–193, 2002.
12. Ian Holmes and Gerald M. Rubin. Pairwise RNA structure comparison with stochastic context-free grammars. In *Pacific Symposium on Biocomputing*, pages 163–174, 2002.
13. G. Lin, Z-Z. Chen, T. Jiang, and J. Wen. The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65(3):465–480, 2002. Special issue on computational biology.
14. Jianghui Liu, Jason TL Wang, Jun Hu, and Bin Tian. A method for aligning RNA secondary structures and its application to RNA motif detection. *BMC Bioinformatics*, 6(89), 2005.
15. A. Lozano and G. Valiente. On the maximum common embedded subtree problem for ordered trees. In C. Iliopoulos and T Lecroq, editors, *String Algorithmics*, chapter 7. King's College London Publications, 2004.
16. R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithms for loop matching. *SIAM Journal of Applied Mathematics*, 35(1):68–82, 1978.
17. S. Vialette. On the computational complexity of 2-interval pattern matching. *Theoretical Computer Science*, 312(2-3):223–249, 2004.
18. M.S. Waterman. *Introduction to computational biology - Maps, sequences and genomes*. Chapman and Hall, London, 1995.
19. K. Zhang and D. Shacha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal of computing*, 18(6):1245–1262, 1989.