Alignments of RNA structures.
Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, Helene Touzet

To cite this version:
Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, Helene Touzet. Alignments of RNA structures. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Institute of Electrical and Electronics Engineers, 2010, 7 (2), pp.309-322. <10.1109/TCBB.2008.28>. <hal-00506348>
Alignments of RNA structures

Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, and Hélène Touzet

Abstract—We describe a theoretical unifying framework to express comparison of RNA structures, which we call alignment hierarchy. This framework relies on the definition of common supersequences for arc-annotated sequences, and encompasses main existing models for RNA structure comparison based on trees and arc-annotated sequences with a variety of edit operations. It also gives rise to edit models that have not been studied yet. We provide a thorough analysis of the alignment hierarchy, including a new polynomial time algorithm and an NP-completeness proof. The polynomial time algorithm involves biologically relevant evolutionary operations, such as pairing or unpairing nucleotides. It has been implemented in a software, called gardenia that is available at the web server http://bioinfo.lifl.fr/RNA/gardenia.

Index Terms—Computational biology, RNA structures, arc-annotated sequences, NP-hardness, edit distance, algorithm

I. INTRODUCTION

Non-coding RNA genes are now known to play essential roles in the cell, and comparison of RNA molecules has attracted a lot of interest recently. Broadly, one can distinguish two combinatorial models for RNA structures: macroscopic representations, with two-interval graphs [8], [28], and microscopic representations with arc-annotated sequences [11]. We focus here on arc-annotated sequences. In this formalism, an RNA molecule is represented as a raw sequence of nucleotides provided with related additional information in the form of arcs connecting pairs of positions. The set of arcs constitutes the secondary and the tertiary structures of the molecule. It determines the way the sequence folds into a three-dimensional space.

When it comes to compare arc-annotated sequences, four main paradigms have been proposed so far: tree edit distance [9], [10], [21], [27], [29], tree alignment [18], longest common arc-preserving subsequence [11], [17], [22], and general edit distance [5], [16]. In this paper, we introduce a unifying framework to address the problem of arc-annotated sequence comparison that is based on the definition of the common arc-annotated supersequence. This framework has several instances depending on the definition of the embedding involved in the notion of supersequence, and the type of the supersequence (NESTED, CROSSING or UNLIMITED). It gives rise to a hierarchy of problems, that we call the alignment hierarchy in reference to the tree alignment of [18]. We show that this hierarchy brings together all four previously mentioned comparison models for arc-annotated sequences. It also leads to the introduction of new comparison models. In particular, we propose in Sections IV-B and V a polynomial time algorithm for the problem of comparing two NESTED arc-annotated sequences with a full set of edit operations (including arc-altering and arc-breaking), whereas these edit operations yield a non polynomial time algorithm in the edit distance scheme. We provide a full analysis of the complexity of this algorithm in the worst case and in average in paragraphs V-C and V-D, and demonstrate its applicability on RNA sequences in paragraph V-E. We also present a NP-completeness result that gives some new insight on the hardness of the comparison of NESTED arc-annotated sequences with arc-altering and arc-breaking operations (Section IV-C), refining previous results of [5], [22]. This leads to an almost exhaustive study of the alignment hierarchy.

II. EDITION MODELS FOR ARC-ANNOTATED SEQUENCES

Given a finite alphabet $\Sigma$, an arc-annotated sequence is defined by a pair $(S, P)$, where $S$ is a string of $\Sigma^*$ and $P$ is a set of arcs connecting pairs of characters of $S$. Arcs correspond to hydrogen interactions between bases. In reference to RNA structures, characters are called bases. Bases with no incident arc are called single bases. As usually done in the study of arc-annotated sequences, we distinguish four levels of arc structure, originally proposed by Evans in [11]:

- **UNLIMITED (UNLIM)** – no restriction at all,
- **CROSSING (CROS)** – there is no base incident to more than one arc,
- **NESTED (NEST)** – there is no base incident to more than one arc and no arcs are crossing,
- **PLAIN** – there is no arc.

Since we focus here on structure comparison, we do not consider PLAIN sequences, which do not carry any structural information. In the remaining of this paper, we shall only deal with sequences of type NESTED, CROSSING and UNLIMITED.

In order to compare two arc-annotated sequences, we consider the set of edit operations and their associated costs introduced in [23], and classify it into two groups:

- **Substitution operations**, inducing renaming of bases in the arc-annotated sequence: base-match ($w_m : \Sigma^2 \rightarrow R$), base-mismatch ($w_m : \Sigma^2 \rightarrow \overline{R}$), arc-match ($w_{am} : \Sigma^4 \rightarrow R$), arc-mismatch ($w_{am} : \Sigma^4 \rightarrow \overline{R}$).
- **Deletion operations**, inducing deletion of bases and/or of arcs:

  - base-deletion ($w_d : \Sigma \rightarrow R$)
  - arc-breaking ($w_b : \Sigma^4 \rightarrow \overline{R}$)
  - arc-removing ($w_r : \Sigma^2 \rightarrow R$)
  - arc-altering ($w_a : \Sigma^3 \rightarrow R$) or

G. Blin is with the Institut Gaspard Monge, UMR 8049 CNRS – Université Paris-Est. A. Denise and C. Herrbach are with the LRI, UMR 8623 CNRS – Université Paris-Sud. S. Dulucq is with the Laboratoire d’Informatique de Paris Est (LIFL), UMR 8000 CNRS – Université Paris-Est Créteil. H. Touzet (corresponding author) is with the LIPI, UMR 8022 CNRS – Université Paris 1. A preliminary version of some of this work appears as an extended abstract in the Proceedings of the SPIRE 2006 conference [6].
Although this is not explicit in the notation, the cost of any deletion operation depends on the values of the affected bases. The definition in [23] is slightly different: deletion operations are defined in such a way that they cannot change the value of the remaining bases. For example, changing a G → C base pair into two single bases G and A needs two operations: at first an arc-breaking, then a base-mismatch from C to A. Here, we choose to allow them to change the bases incident to the arc. Hence, in the above example, only one arc-breaking has to be done. It can be easily seen the two models are equivalent, by changing the costs of the operations.

This set of operations allows us to define three edit models:

I : all substitution operations, base-deletions and arc-removals are allowed,

II : the operations of model I and arc-alterations are allowed,

III : the operations of model II and arc-breakings are allowed.

Given two arc-annotated sequences u and v, and K in {I,II,III} a K-edit script from u to v refers to a series of non-oriented operations of the model K transforming u into v. The cost of a K-edit script from u to v, denoted cost(u, v, K), is the sum of the costs of each operation involved in the K-edit script. We define the K-edit distance between u and v as the minimum cost of a K-edit script from u to v. Finding this K-edit distance is called the Edit(u, v, K) problem. For each model K ∈ {I,II,III}, we also define the ordering relation ≤K: if u can be obtained from v by a series of deletion and substitution operations of the model K, then u ≤K v.

Provided with these notations, we propose to extend the notion of subsequence to arc-annotated sequences as follows.

Definition 1 (K-subsequence) Given two arc-annotated sequences u and v, and an edit model K ∈ {I,II,III}, u is said to be a K-subsequence of v if, and only if, u ≤K v.

Given three arc-annotated sequences u, v and w such that w ≤K u and w ≤K v, w is said to be a common K-subsequence of u and v. We define the cost of a common K-subsequence w of u and v as the minimum sum of operation costs needed to transform u into w and v into w: cost(u, w, K) + cost(v, w, K).

When dealing with plain sequences, it is well-known that each edit script can be associated with a common subsequence of the same cost. This property is still valid with K-edit scripts on arc-annotated sequences.

Lemma 1 Given two arc-annotated sequences u and v, and an edit model K ∈ {I,II,III}, solving the Edit(u, v, K) problem is equivalent to finding a common K-subsequence w of u and v of minimal cost.

Proof: Let w be a common K-subsequence of u and v. By definition, we have w ≤K u and w ≤K v. Therefore, there exist two series of operations of the model K that respectively transform u into w and v into w. It is straightforward to verify that these operations induce an edit script whose cost equals cost(u, w, K) + cost(v, w, K). Thus the edit distance is lower than or equal to the cost of w.

(⇒) Conversely, let M be a K-edit script from u to v of cost α. We show that there exists a common K-subsequence whose cost is lower than or equal to α. According to the parsimony principle, each position of u or v is affected by at most one deletion operation in M. If not, M is not optimal and can be simplified so as to eliminate redundant operations. Now, since each position of u or v is concerned by at most one operation, we are ensured that there are no conflicting pairs, i.e. pairs that share a common base which is concerned by two operations on its adjacent arcs. It follows that the script may be modified in such a way that all deletion rules on u apply before any deletion rule on v. A common K-subsequence w of u and v can then be obtained by applying to u all the operations of the reordered K-edit script appearing before the first deletion rule on v. The cost of w is lower than or equal to α.

We now turn to a novel paradigm to compare arc-annotated sequences, simply considering K-supersequences instead of K-subsequences. We shall see that this alternative point of view is a fruitful perspective and that it brings new insights on arc-annotated sequence comparison.

Definition 2 (K-supersequence) Given two arc-annotated sequences u and v, and an edit model K ∈ {I,II,III}, u is said to be a K-supersequence of v if, and only if, v ≤K u.

In a similar way as for common subsequences, given three arc-annotated sequences u, v and w, w is a common K-supersequence of u and v if u ≤K w and v ≤K w. The cost of w is defined as cost(u, w, K) + cost(w, v, K).

We saw in Lemma 1 that Edit problems amount to finding optimal subsequences. We prove that each Edit problem can also reduce to finding an optimal supersequence.

Lemma 2 Given two arc-annotated sequences u and v, and an edit model K ∈ {I,II,III}, there exists a common K-supersequence of u and v of cost α iff there exists a common K-supersequence of u and v of the same cost.

Proof: (⇒) Let u = (S, P), v = (T, Q) and w = (R, U) be three arc-annotated sequences such that w is a common K-subsequence of u and v. For each position i of R, let φ(i, R, S) (resp. φ(i, R, T)) denote the position of the character in S (resp. T) from which the character R[i] is obtained. We build a K-supersequence x = (V, W) of u and v as follows:

V = S_1 T_1 R[1] S_2 T_2 R[2] ... S_n T_n R[n] S_{n+1} T_{n+1}
W = \{ψ_u(i), ψ_v(j) : (i, j) ∈ P\}
∪ \{ψ_u(i), ψ_v(j) : (i, j) ∈ Q\}

where n is the length of R and S_i (resp. T_i) denotes S[φ(i - 1, R, S) + 1...φ(i, R, S) - 1] (resp. T[φ(i - 1, R, T) + 1...φ(i, R, T) - 1]). By construction, we have φ(0, R, S) = φ(0, R, T) = 0 and φ(n + 1, R, S) (resp. φ(n + 1, R, T)) is the last position of S (resp. T). ψ_u (resp. ψ_v) is an application that associates to each base of S (resp. T) the corresponding base in V. By construction, x is indeed a common supersequence of u and v. We now turn to prove that its cost is α. First, note that
The purpose of this paper is to study exhaustively the alignment hierarchy and confront it to known results for existing comparison models for arc-annotated sequences.

What is the number of different instances in the hierarchy? Since \( \text{ALIGN}(A, B, K) \rightarrow C \) is equivalent to \( \text{ALIGN}(B, A, K) \rightarrow C \), we can always assume that \( B \subseteq A \). Moreover, in order for the problem to be meaningful, we impose \( A \subseteq C \). Therefore, the hierarchy contains thirty distinct entries, ten for each edit model, when considering all relevant possibilities for \( A, B, C \) and \( K \).

The first noticeable result is that the ALIGN hierarchy includes all instances of the edit distance problem, as stated in Theorem 1. This is a straightforward consequence of Lemma 1 and Lemma 2.

**Theorem 1** Given two types \( A, B \) in \{\text{NEST}, \text{CROS}, \text{UNLIM}\} and an edit model \( K \in \{I, II, III\} \), the \( \text{EDIT}(A, B, K) \) and \( \text{ALIGN}(A, B, K) \rightarrow \text{UNLIM} \) problems are equivalent.

The three next sections are devoted to the study of the alignment hierarchy for each edit model \( K \in \{I, II, III\} \).

**III. ORDERED TREES AND EDIT MODEL I**

Comparing arc-annotated sequences of NESTED types when considering the edit model I amounts to comparing ordered trees. Each pair of connected bases corresponds to an internal node, and each single base corresponds to a leaf. In this model, considering supersequence of UNLIMITED type is meaningless as stated in Lemmas 3 and 4.

**Lemma 3** Given two types \( A, B \) in \{\text{NEST}, \text{CROS}\}, the \( \text{ALIGN}(A, B, I) \rightarrow \text{UNLIM} \) and \( \text{ALIGN}(A, B, I) \rightarrow \text{CROS} \) problems are equivalent.

**Proof:** Only arc-altering and arc-breaking operations (which are prohibited in this edit model) can create multiple arcs incident to a single character – which is the only property that arc-annotated sequences of CROS and UNLIMITED types do not have in common.

**Lemma 4** Given a type \( B \) in \{\text{NEST}, \text{CROS}\}, the \( \text{ALIGN}(\text{UNLIM}, B, I) \rightarrow \text{UNLIM} \) problem has the same complexity as \( \text{ALIGN}(\text{CROS}, B, I) \rightarrow \text{CROS} \).

**Proof:** Since this edit model does not allow for arc-altering or arc-breaking operations, all multiple incident arcs should be deleted with an arc-removing operation, which can be done in linear time. So the UNLIMITED arc-annotated sequence is rewritten into a CROS arc-annotated sequence. Conclusion stems from Lemma 3.

Together with Theorem 1, these two lemmas imply that nine out of ten entries of the model I are equivalent or reduce to EDIT problems. The only problem that does not reduce to an edit problem is \( \text{ALIGN}(\text{NEST}, \text{NEST}, I) \rightarrow \text{NEST} \), which fully corresponds to the ordered tree alignment, introduced by Jiang et al. in [18]. Therefore, the ALIGN hierarchy is completely solved for the edit model I, as summed up in Table I.
TABLE I
ALIGNMENT HIERARCHY FOR THE EDIT MODEL I.

<table>
<thead>
<tr>
<th>A × B → C</th>
<th>Edit</th>
<th>model 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEST × NEST → NEST</td>
<td>O(n^2) – Jiang [18]</td>
<td></td>
</tr>
<tr>
<td>NEST × NEST → CROS</td>
<td>O(n^3 log(n)) – Klein [21]</td>
<td></td>
</tr>
<tr>
<td>CROS × NEST → NEST</td>
<td>O(n^3 log(n)) – Ma [24]</td>
<td></td>
</tr>
<tr>
<td>CROS × CROS → NEST</td>
<td>NP-complete – Ma [24]</td>
<td></td>
</tr>
<tr>
<td>CROS × CROSS → UNLIM</td>
<td>NP-complete – Ma [24]</td>
<td></td>
</tr>
<tr>
<td>UNLIM × NEST → UNLIM</td>
<td>O(n^3 log(n)) – Lemma 4</td>
<td></td>
</tr>
<tr>
<td>UNLIM × CROS → UNLIM</td>
<td>NP-complete – Ma [24]</td>
<td></td>
</tr>
</tbody>
</table>

According to Lemma 3, the ten problems of the hierarchy reduce to seven distinct instances. We indicate encytes that can also be formulated as edit problems with × in the second column (see Theorem I). Complexity results are indicated for two arc-annotated sequences u and v s.t. max(|u|, |v|) = n.

IV. THE EDIT MODEL II

A. Some correspondences with the LAPCS problem

As introduced by Evans in [11], the LONGEST ARC-PRESERVING COMMON SUBSEQUENCE problem (LAPCS for short) is defined as follows: given two arc-annotated sequences u and v, find the longest – in terms of sequence length – common arc-preserved subsequence w of u and v such that an arc (i, j) in w can only be obtained from both an arc in u and an arc in v (i.e. arc-preserving). We prove hereafter that the LAPCS problem is a specific case of the common subsequence problem when considering the edit model II, namely the EDIT(A, B, II) problem, provided that the score system for edit operations is correctly chosen. The cost of a base-deletion or of an arc-altering is 1, the cost of an arc-removing is 2, and substitutions are prohibited, with arbitrary high costs.

Theorem 2 Let u, v, w be three arc-annotated sequences. The sequence w is a longest arc-preserving common subsequence of u and v iff w ≤II v and w ≤II u.

Proof: The proof relies on the following property: Let u' = (S, P) and v' = (T, Q) be two arc-annotated sequences. We have u' ≤II v' if S is a common arc-preserving subsequence of T considering the implicit mapping – denoted M – from u' to v' induced by u' ≤II v'.

⇒ The proof is by induction on the number of edit rules necessary to reduce v into u. All deletion rules of the edit model II (base-deletion, arc-removing and arc-altering) clearly have the arc-preservation property.

⇐ The proof is by induction on the difference of lengths between S and T. If S and T have the same length, we have S = T and the condition on arc preservation yields P = Q. If T is longer than S, then let i be the first position in T such that for any position j in S the pair (i, j) does not belong to M. It is enough to show that there exists an arc-annotated sequence w = (U, R) such that u ≤II w on the one hand, U is longer than S, U is a subsequence of T with arc-preservation property on the other hand. Then the induction hypothesis will allow us to conclude that w ≤II v, which implies u ≤II v by transitivity of ≤II.

We have to consider several cases according to the status of T[i] for the construction of w. We note S' (resp. S'') the image of T[1...i-1] in S (resp. T[i+1...|T|]) in the mapping M. By construction, we have S'S'' = S.

− T[i] is a single base: w is defined by U = T[1...i-1] ∪ T[i+1...|T|] and R = Q. We have w ≤II v since w is derived from v by a base-deletion of T[i]. The arc-preservation property between u and w still holds. So the induction hypothesis implies u ≤II v.

In the other cases, T[i] is a paired base. Let k be the position of its partner (i.e. (i, k) ∈ Q).

− If there exists a position l in S such that (k, l) belongs to M: According to the arc preservation property for u and v, S[l] is a single base. We define U = T[1...i-1] ∪ T[i+1...|T|] and R = Q − {(i, k)}. We have w ≤II v since w is derived from v by an arc-altering on T[i] and T[k]. The arc-preservation property between u and w still holds. So the recurrence hypothesis implies u ≤II v.

− k is not mapped to any position in S with M: We define w as the arc-annotated sequence obtained from v by application of an arc-removing operation on (i, k). The arc-preservation property between u and w still holds. So the recurrence hypothesis implies u ≤II v.

This theorem combined with Theorem 1 allows us to derive several cases of the alignment hierarchy for the edit model II from results of the LAPCS literature. All known results are summed up in Table II. LAPCS(NEST, NEST, NESTED) corresponds to ALIGN(NEST, NEST, II) → UNLIM, is known to be NP-complete, and so do LAPCS(CROSSING, NESTED), LAPCS(CROSSING, CROSSING), LAPCS(UNLIM, NESTED), LAPCS(UNLIM, CROSSING) and LAPCS(UNLIM, UNLIM). It remains four problems: ALIGN(NEST, NEST, II) → {NEST, CROS} and ALIGN(CROS, {NEST, CROS}, II) → CROS. The first two problems can be seen as refinements of LAPCS(NESTED, NESTED). We solve them in the next two sections, and show that the first one is polynomial, whereas the second one is NP-complete. It follows that ALIGN(CROS, NEST, II) → CROS and ALIGN(CROS, CROS, II) → CROS are also NP-complete.

B. ALIGN(NEST, NEST, II) → NESTED is polynomial

The first result that we present for the edit model II is concerned with the alignment of two NESTED sequences. This is indeed a consequence of the more general algorithm proposed for model III in Theorem 6 and Table IV (Section V).

Theorem 3 ALIGN(NESTED, NESTED, II) → NESTED is polynomial.

This result is somehow unexpected since the associate edit problem EDIT(NESTED, NESTED, II) is NP-complete. It shows that prohibiting crossing arcs in the superstructure is sufficient to make the problem polynomial.
We indicate problems that can be formulated as edit distance problems in the second column. In these cases, known results stem from the LACPS problem (Theorems 1 and 2). Other problems are specific to the ALIGN hierarchy and are introduced in this paper. Complexity results are indicated for two arc-annotated sequences \( u \) and \( v \) s.t. \( \max(|u|, |v|) = n \).

C. ALIGN(NESTED, NESTED, II) \( \rightarrow \) CROSSING is NP-hard

We show in this section that relaxing the constraint on crossing arcs in the common supersequence makes the problem difficult, even if we do not allow multiple incidents arcs in the supersequence as in LAPCS(NESTED, NESTED).

**Theorem 4** ALIGN(NESTED, NESTED, II) \( \rightarrow \) CROSSING is NP-complete.

The decision problem is defined formally as follows.

**Input:** Two arc-annotated sequences \( u \) and \( v \) of NESTED type and an integer \( \ell \).

**Question:** Can one find an arc-annotated sequence \( w \) of CROSSING type which is a common II-supersequence of \( u \) and \( v \) of cost lower than or equal to \( \ell \) ?

We initially notice that this problem is in \( \text{NP} \) since given three arc-annotated sequences \( u \), \( v \) and \( w \) one can check polynomially if (1) \( w \) is of CROSSING type, (2) \( w \) is a common II-supersequence of \( u \) and \( v \), and (3) the cost of \( w \) is lower than or equal to \( \ell \). In order to prove that it is \( \text{NP} \)-complete, we propose a polynomial reduction from the \( \text{NP} \)-complete problem MIS-3P [4]. The MIS-3P problem is also used in the polynomial reduction of the \( \text{NP} \)-complete proof of LAPCS(NESTED, NESTED) [22].

**Input:** A cubic planar bridgeless connected graph \( G = (V, E) \) and an integer \( k \).

**Question:** Is there an independent set of vertices of \( G \) – i.e. a set \( V' \subseteq V \) such that no two vertices of \( V' \) are connected by an edge in \( E \) – of cardinality greater than or equal to \( k \) ?

A graph \( G = (V, E) \) is said to be a cubic planar bridgeless connected graph if any vertex of \( V \) is of degree three (cubic), \( G \) can be drawn in the plane in such a way that no two edges of \( E \) cross (planar), and there are at least two paths – with no edge in common – connecting any pair of vertices of \( V \) (bridgeless connected).

The idea of the proof is to encode a cubic planar bridgeless connected graph by two arc-annotated sequences. The construction uses first a two-page book embedding.

**Theorem 5** (Bernhart and al. [3]) One can always find, in polynomial time, a two-page book embedding of a cubic planar bridgeless connected graph with the following additional property: on each page, any vertex has a non-null degree.

A two-page book embedding of a graph \( G \) is a linear ordering of the vertices of \( G \) along a line and an assignment of the edges of \( G \) to the two half-planes delimited by the line – called the pages – so that no two edges assigned to the same page cross. For convenience, we will refer to the page above (resp. below) the line as the top-page (resp. bottom-page).

Given a two-page book embedding, we construct two arc-annotated sequences of NESTED type \( u = (S, P) \) and \( v = (T, Q) \) on the three-letters alphabet \( \{a, b, \#\} \). The underlying raw sequences \( S \) and \( T \) are defined as follows:

\[
S = \#^n S_1 \#^n S_2 \ldots \#^n S_n \\
T = \#^n T_1 \#^n T_2 \ldots \#^n T_n
\]

where \( n \) is the number of vertices of the initial graph, and for each \( 1 \leq i \leq n \), \( S_i \) (resp. \( T_i \)) is a segment \( abaa \) if the degree of the vertex \( v_i \in V \) in the top-page (resp. bottom-page) equals two, a segment \( aaab \) otherwise.

Now that the sequences \( S \) and \( T \) are defined, we have to copy the arc configuration of the top-page (resp. bottom-page) on \( S \) (resp. \( T \)). Each edge \( (v_i, v_j) \) of the top-page is represented by an arc in \( P \). More precisely, this arc connects a base \( a \) of \( S_i \) and a base \( a \) of \( S_j \). We proceed in a similar way for each edge of the bottom-page by adding, for each one, an arc in \( Q \). Moreover, we impose that when a vertex \( v_i \) is of degree two on the top-page (resp. bottom-page), the two corresponding arcs in \( P \) (resp. \( Q \)) are incident to the rightmost two bases \( a \) of the segment \( S_i \) (resp. \( T_i \)). And, consequently, we impose that, when a vertex \( v_i \) is of degree one on the top-page (resp. bottom-page), the corresponding arc in \( P \) (resp. \( Q \)) is incident to the leftmost base \( a \) of the segment \( S_i \) (resp. \( T_i \)). It is easy to check that it is always possible to reproduce on
Fig. 2. Example of an align-construction for the proof of Theorem 4. The graph (a) is a cubic planar bridgeless connected graph of 6 vertices. The graph (b) is a two-page book embedding of the graph (a) such that, on each page, any vertex has a non-null degree. (c) The two arc-annotated sequences of NESTED type obtained from the graph (a) by an align-construction.

u and v the non-crossing edge configuration of each page. An example of such a construction is given in Figure 2. The size of u and v is quadratic in n: the length of S and T is n(n+4) and the total number of arcs is 3n. In the following, we will refer to any such construction as an align-construction.

It remains to define parameter values for edit operations. We set the score system as follows: \( w_d(b) = 2 \), \( w_d(#) = 6 \), \( w_d(a) = 1 \), \( w_r(a,a) = 1 \). \( w_r(a,a) = 2 \). As a matter of fact, the proofs of Lemmas 5, 6, 7, 8 are still valid with any combination of parameters that fulfills these two inequalities: \( 3w_u(a,a,a) + 2w_d(b) = 3w_r(a,a,a) + 3w_d(a) \) and \( w_r(a,a) + 3w_d(a) < w_u(a,a,a) + 2w_d(b) \).

We first show that for any such pair of arc-annotated sequences with the given score system, there exists a “canonical” optimal common II-supersequence whose form is easy to characterize. This is the purpose of the two following Lemmas.

**Lemma 5** Let u and v be two arc-annotated sequences of NESTED type obtained by an align-construction for an initial graph of n vertices. There exists an optimal common II-supersequence \( w = (U,R) \) such that \( U \) is of the form \( #^nU_1\ldots#^nU_n \), where for each \( i \in 1..n \), \( U_i = aabaaba \) or \( U_i = baaba \).

**Proof:** It is easy to verify that \((#^n aabaaba)^n\) is a common II-supersequence whose cost is lower than or equal to \( n(3w_u(a,a,a) + 3w_d(a)) = 6n \). This observation ensures that any optimal supersequence is of the form \( U = #^nU_1\ldots#^nU_n \), where \( U_i \in \{a,b\}^* \). Indeed, assume that an optimal supersequence contains more than \( n^2 \) occurrences of \# symbol. This implies that the supersequence contains one extra stretch of \( n \) occurrences of \#, which will give rise to \( n \) base deletions of \#. Therefore the associated cost is at least \( nw_d(#)^n = 6n \).

By construction, each \( U_i \) is a supersequence of aab and baab. There are five candidate strings: aabaaba, baaba, baabaab, baabaaba and baabaaba (all other sequences are equivalent). We show that any optimal supersequence cannot contain any \( U_i \) of the three last kinds.

Assume there exists \( i \in 1..n \) such that \( U_i = baaba \). We suppose w.l.o.g. that \( S_i = aab \) and \( T_i = baab \). The construction of \( u \) and \( v \) ensures that there is no \( j \) such that there exists an arc connecting both \( S_i \) and \( S_j \) in \( u \), and \( T_i \) and \( T_j \) in \( v \). Therefore three arcs are incident from \( U_i \). Let \( j \) (resp. \( k \) and \( l \)) be the position of the pairing partner of the first \( a \) of \( S_i \) in \( U \) (resp. the second and third \( a \) of \( T_i \) in \( U \)). There are five cases to consider (see Figure 3). The main argument that is common to all cases is that replacing \( U_i \) with aabaaba does not increase the cost of the alignment.

**Case 1:** \( U[j] \) does not appear in \( T \), and \( U[k], U[l] \) do not appear in \( S \)

**Case 2:** \( U[j] \) does not appear in \( T \), and one of \( U[k] \) or \( U[l] \) appears in \( S \)

**Case 3:** \( U[j] \) appears in \( T \), and \( U[k], U[l] \) do not appear in \( S \)

**Case 4:** \( U[j] \) appears in \( T \), and one of \( U[k] \) or \( U[l] \) appears in \( S \)

![Fig. 3. Four first cases for the replacement of \( U_i \) when \( U_i = baaba \) in Proof of Lemma 5](attachment:image.png)
and a base-deletion of $a$, whereas $T_1$ is derived from $aaabaaa$ by an arc-removing and two base-deletions of $a$. The total cost is $2w_r(a,a) + w_u(a,a) + 3w_d(a) = 8.5$.

3. $U[j]$ appears in $T$, and $U[k]$, $U[l]$ do not appear in $S$. On the one hand, $S_i$ is derived from $U_i$ by an arc-altering, an arc-removing and a base-deletion of $b$, and $T_i$ is derived from $U_i$ by an arc-altering and a base-deletion of $b$. The corresponding cost is $w_r(a,a) + 2w_u(a,a) + 2w_d(b) = 9$. On the other hand, $S_i$ is derived from $aaabaa$ by two arc-removings and a base-deletion of $a$, whereas $T_i$ is derived from $aaabaa$ by an arc-altering and two base-deletions of $a$. The total cost is $2w_r(a,a) + w_u(a,a) + 3w_d(a) = 8.5$.

4. $U[j]$ appears in $T$, and one of $U[k]$ or $U[l]$ appears in $S$. On the one hand, $S_i$ is derived from $U_i$ by two arc-removings, and a base-deletion of $b$, whereas $T_i$ is derived from $U_i$ by an arc-altering and a base-deletion of $b$. The corresponding cost is $3w_u(a,a) + 2w_d(b) = 8.5$. On the other hand, $S_i$ is derived from $aaabaa$ by an arc-altering, an arc-removing and a base-deletion of $a$, and $T_i$ is derived from $U_i$ by an arc-altering and two base deletions of $a$. The cost is $w_r(a,a) + 2w_u(a,a) + 3w_d(a) = 8$.

5. $U[k]$ and $U[l]$ both appear in $S$: this last case is impossible, since it would imply that $aaab$ is derived from $baaab$ without any operation of arc-altering.

The reasoning is similar for $baaab$ and $baaab$.

**Lemma 6** Let $u$ and $v$ be two arc-annotated sequences of Nested type obtained by an align-construction. In any optimal common II-supersequence $w = (U,R)$ of $u$ and $v$, if there is an arc in $R$ connecting a base of the segment $U_i$ and a base of the segment $U_j$, then $U_i$ and $U_j$ cannot both be of the form $baaab$.

**Proof:** By contradiction, let us assume that there exists such an arc for a given $1 \leq i \leq n$ and a given $1 \leq j \leq n$, $U_i$ and $U_j$ being both of type $baaab$, this arc will induce either an arc-breaking between $w$ and $u$, or an arc-breaking between $w$ and $v$. Since we are considering the edit model II, this operation is forbidden. This leads to a contradiction.

These lemmas allow us to express the cost of an optimal Nested supersequence between two arc-annotated sequences obtained with the align-construction.

**Lemma 7** Let $u$ and $v$ be two arc-annotated sequences of Nested type obtained by an align-construction. The cost of any optimal common II-supersequence $w$ is $3pw_u(a,a,a) + 3\left(\frac{p}{2} - p\right)w_r(a,a) + 3(n - p)w_d(a) + 2pw_d(b)$, where $p$ is the number of segments of $w$ of type $baaab$.

**Proof:** By construction, the supersequence $w$ contains $3\frac{p}{2}$ arcs, three arcs being incident to a base from each segment $U_i$. Lemma 6 ensures that there is no arc between two segments of type $baaab$. So there are $3p$ arcs connecting a segment of type $baaab$ with a segment of type $aaabaa$, and $3\frac{p}{2} - 3p$ arcs connecting two segments $aaabaa$. As mentioned before, each arc of the supersequence is present only in one of the two sequences $u$ and $v$. So each arc of $w$ is affected by a deletion operation. Moreover, an arc between two segments of type $aaabaa$ gives rise to an arc-removing, whereas an arc between a segment $baaab$ and a segment $aaabaa$ gives rise to an arc-altering. It follows that the total cost of deletion operations on arcs is $3pw_u(a,a,a) + 3\left(\frac{p}{2} - p\right)w_r(a,a)$.

As for the single bases, each segment $aaabaa$ produces three base-deletions of $a$, and each segment $baaab$ produces two base-deletions of $b$. It follows that the global cost is $3pw_u(a,a,a) + 3\left(\frac{p}{2} - p\right)w_r(a,a) + 3(n - p)w_d(a) + 2pw_d(b)$.

The following Lemma concludes the proof of Theorem 4.

**Lemma 8** A cubic planar bridgeless connected graph $G = (V,E)$ admits an independent set of vertices of cardinality greater than or equal to $k$ if, and only if, there exists an arc-annotated sequence $w$ of Crossing type that is a common II-supersequence of $u$ and $v$ of cost lower than or equal to $\ell = 3kw_u(a,a,a) + 3\left(\frac{p}{2} - k\right)w_r(a,a) + 3(n - k)w_d(a) + 2kw_d(b) = 3|V'|w_u(a,a,a) + 3\left(\frac{p}{2} - k\right)w_r(a,a) + 3(n - k)w_d(a) + 2kw_d(b)$. By Lemma 7, the cost of the alignment induced by $w$ is $3|V'|w_u(a,a,a) + 3\left(\frac{p}{2} - |V'|\right)w_r(a,a) + 3(n - |V'|)w_d(a) + 2|V'|w_d(b)$. Since by hypothesis $|V'| \geq k$, this cost is majorized by $3kw_u(a,a,a) + 3\left(\frac{p}{2} - k\right)w_r(a,a) + 3(n - k)w_d(a) + 2kw_d(b)$, which equals $\ell$.

(⇐) Let Lemma 5, there exists an optimal supersequence $w = (U,R)$ of cost lower than or equal to $\ell$ that is composed of $n$ stretches of $\#^n$ and of segments $aaabaa$ and $baaab$. Let $V'$ be the set of vertices of $G$ defined by $\{v_i \in V; U_i = baab\}$. By Lemma 7, the cost of the initial alignment is $3|V'|w_u(a,a,a) + 3\left(\frac{p}{2} - |V'|\right)w_r(a,a) + 3(n - |V'|)w_d(a) + 2|V'|w_d(b)$. Since by hypothesis this score is lower than or equal to $\ell$ and $w_r > w_u$, we obtain $k \leq |V'|$.

One can remark that the arc-annotated sequences of the NP-completeness proof are not conform to the representation of an RNA molecule. One can modify the encoding of the two-page book embedding in order to get sequences that are more realistic: the alphabet is $\{A, U, C, G\}$ and all arcs correspond to Watson-Crick pairings ($A$ is paired with $U$, and $C$ with $G$). To achieve this goal, we modify the definition of $u$ and $v$ in the following way: replace $\#^n$ with twelve occurrences of $C$, $b$ with $GGGGGG$ and $a$ with $AU$ ($AU$ is self-complementary). Each edge in the two-page book embedding now corresponds to two arcs between $AU$ and $AU$. Figure 4 shows this new representation for the example of Figure 2.

**V. General edit distance and edit model II**

The edit model III contains all edit operations introduced by Jiang et al. in the general edit distance problem [16]. So we can derive several complexity results for the alignment hierarchy from known results on the general edit distance [5], [16] with Theorem 1. As illustrated in Table III, the complexity of $\text{ALIGN}(\text{NEST}, \text{NEST}, \text{III}) \rightarrow \{\text{NEST}, \text{CROSS}\}$...
and of ALIGN(CROS, {NEST, CROS}, III) → CROS only is still to elucidate. We solve ALIGN(NEST, NEST, III) → NEST.

**Theorem 6** ALIGN(NEST, NEST, III) → NEST is polynomial.

The proof of the Theorem follows from Theorem 7 in paragraph V-B and Theorem 8 of paragraph V-C. We first need some notations for the representation of NEST arc-annotated sequences.

**A. Notations**

We write α(f) a NEST arc-annotated sequence, or equivalently a tree, that is composed of a root α and a subforest f. A NEST arc-annotated sequence is defined recursively by concatenating a tree and an arc-annotated sequence. Let ◦ be a binary operator that concatenates two arc-annotated sequences. α(u) ◦ v denotes the arc-annotated sequence composed by an arc α spanning the arc-annotated sequence u, concatenated to the arc-annotated sequence v. Let b ∈ Σ. b ◦ u denotes the arc-annotated sequence composed by the single base b concatenated to the arc-annotated sequence u.

**B. Algorithm**

We saw in Section III that the ALIGN(NEST, NEST, I) → NEST problem is polynomial, since it is equivalent to ordered tree alignment such as proposed in [18]. We show here that the construction scheme for the edit model I can be extended to edit models II and III by adding supplementary rules for the arc-altering and arc-breaking operations. All rules concerning substitutions, base-deletions and arc-removings are identical.

In Table IV, we state the recurrences which enable to compute the alignment score of two sequences, denoted A1. The common supersequence is built from right to left. Each step of the algorithm adds a component in the supersequence – one single base or two bases connected by an arc – that is selected so as to minimize the cost of the alignment. Several particular cases are needed for the arc-breaking and arc-altering operations. We consider five cases depending on the form of the pair of arc-annotated sequences to align, that determines which edition rules to apply. Arc-altering operation creates an arc in the common supersequence. So it only should be considered when at least one of the two sequences begins with a base incident to an arc. Arc-breaking operation requires that one sequence begins with an arc, and the other one begins with a single base. The implementation uses on dynamic programming. An optimal supersequence is recovered from A1 by trace back.

**Theorem 7** The algorithm of Table IV solves the ALIGN(NEST, NEST, III) → NEST problem.

**Proof:** We show that at each step of the algorithm, A1(f, g) is the cost of an optimal NESTED supersequence of f and g, for any pair of subforests f and g. The algorithm contains five possible cases. Case 1 is a subcase of case 2, case 4 is a symmetric case of case 3, and case 5 is obvious. So we examine cases 2 and 3 in full details. Throughout the proof, S(f, g) denotes an optimal common supersequence of f and g.

**Case 2.** Let S = S(α(u) ◦ v, β(w) ◦ x), and let (i, j) be the positions of α in S (i < j), (k, l) be the positions of β in S (k < l).

- If i = k: This configuration corresponds to an arc-match or an arc-mismatch between α and β. Indeed, since the supersequence S does not allow for multiple incident arcs, we necessarily have j = l. So the arc α is transformed into β by an arc-match or an arc-mismatch operation. S is obtained as β(S(u, w)) ◦ S(v, x). The resulting cost is wam(α, β) + A1(u, w) + A1(v, x).
- If i < k, then either j < k, or l < j. Other values for j are prohibited because it would induce crossing or multiple arcs in the supersequence S.

- If j < k and i < k: This configuration corresponds to an arc-removing of α. Indeed, k is the first position in S corresponding to a base present in β(w) ◦ x. So α and u have no counterpart in β(w) ◦ x, and S is obtained as α(u) ◦ S(v, β(w) ◦ x). The resulting cost is wam(α, β) + A1(u, w) + A1(v, β(w) + x).
- If i < k and j < k: we have to look further at the position indexed by j. If it is aligned with a single base of b of x, the arc α is affected by an arc-altering operation. If not, the arc α is affected by an arc-removing operation. In the first case, let y and z such that β(w) ◦ x = y zobz. S is obtained as α(S(y, u)) ◦ S(v, z). The resulting cost is wam(α, β) + A1(u, y) + A1(v, z).
- If k < j, then this configuration is exactly equivalent to i < k when we exchange α(u) ◦ v and β(w) ◦ x.

**Case 3.** Let S = S(b ◦ v, β(w) ◦ x), and let i be the position of b in S, (k, l) be the positions of β in S (k < l).

- If i = k and l is aligned with some position of b ◦ v: This configuration corresponds to an arc-breaking of β. Let b2 be the base of v occurring at position l in the supersequence S. b2 is necessarily a single base, since multiple incident arcs are prohibited in S. Furthermore, there exists no arc in v spanning b2, otherwise we would have crossing arcs in S. So there are two (possibly empty) subtrees y and z such that v = y ◦ b2 ◦ z. The optimal supersequence S is obtained as β(S(y, u)) ◦ S(z, x). The resulting cost is wam(β, β) + A1(y, u) + A1(z, x).
- If i = k and l does not correspond to any position in b ◦ v: This configuration corresponds to an arc-altering of β, whose S’ base is aligned with b. Let y be subtree of v starting at position i + 1 and ending at position k in S, and let z be the largest subtree of v starting at position k + 1 in S. Since S does not contain any two crossing arcs, there are no arcs between y and z in v. It follows that S is obtained as β(S(y, u)) ◦ S(z, x). The resulting cost is wam(β, β) + A1(y, u) + A1(z, x).
- If i < k: This configuration corresponds to a base-deletion of b. The supersequence S begins with a single base, that corresponds to b in b ◦ v and that has no counterpart in β(w) ◦ x.
The remaining part of $S$ is obtained as $S(v, \beta(w) \circ x)$. The total resulting cost is $w_v(b) + \Lambda_1(v, \beta(w) \circ x)$.

- If $k < i$ and $l$ is aligned with some position of $b \circ v$: This configuration corresponds to an arc-altering of $\beta$, with $3'$ matching base. Let $b_2$ be the base of $v$ that occurs in position $l$ in $S$. $b_2$ is necessarily a single base, since multiple incident arcs are prohibited in the supersequence. Furthermore, there exists no arc in $v$ spanning $b_2$, otherwise we would have crossing arcs in the supersequence. So there are two (possibly empty) subforests $y$ and $z$ such that $b \circ v = y \circ b_2 \circ z$. $S$ is obtained as $\beta(S(y, w)) \circ S(z, x)$. The resulting cost is $w_v(b_2) + \Lambda_1(y, w) + \Lambda_1(z, x)$.

- If $k < i$ and $l$ does not correspond to any position in $b \circ v$: This configuration corresponds to an arc-removing of $\beta$. Let $y$ be the largest subforest of $b \circ v$ ending at position $l - 1$ in $S$, and let $z$ be the largest subforest of $b \circ v$ starting at position $l + 1$. We have $b \circ v = y \circ z$, $y$ is aligned with $w$ and $z$ with $x$ in $S$. Since $S$ does not contain any crossing arcs, there are no arcs from $y$ to $z$. $S$ is obtained as $\beta(S(y, w)) \circ S(z, x)$. The resulting cost is $w_v(\beta) + \Lambda_1(y, w) + \Lambda_1(z, x)$.

C. Worst-case complexity

Let us state some definitions and notations. Let $f$ be a forest. We denote $n_f$ its number of nodes, $\ell_f$ its number of leaves, and $w_f$ its width, which is the number of concatenated trees it contains. Given a node $v$ of $f$, the degree of $v$, denoted $d_v$, is its number of children. Let $g$ be a subforest of $f$. $g$ is said to be a closed subforest if it contains consecutive sibling trees, i.e., trees whose root nodes are consecutive siblings. A complete subforest is a closed forest containing all the subtrees that have the same parent. A suffix subforest is a closed forest that contains the rightmost tree of a complete subforest. We write $S_f$ for the set of suffix subforests and subtrees of $f$, and $C_f$ for the set of complete subforests.

**Lemma 9** Let $f$ and $g$ be two forests. The pairs of forests appearing in the dynamic programming decomposition of algorithm of Table IV are exactly those of $S_f \times C_g \cup C_f \times S_g$.

**Proof:** The proof is by induction on the sizes of $f$ and $g$. Like in proof of Theorem 7, we treat cases 2 and 3, which are representative of other cases.

**Case 2:** If $(\alpha(u) \circ \beta(w) \circ x)$ belongs to $S_f \times C_g$, then $(u, w)$, $(u, y)$, $(v, x)$, $(v, z)$, $(z, x)$, and $(y, w)$ in $S_f \times C_g$, and $(y, w)$ is in $C_f \times S_g$. Similarly, if $(\alpha(u) \circ \beta(w) \circ x)$ belongs to $C_f \times S_g$, then $(u, w)$, $(y, w)$, $(v, x)$, $(v, z)$, $(z, x)$, $(y, v)$ are in $C_f \times S_g$, and $(u, y)$ is in $S_f \times C_g$.

**Case 3:** If $(b \circ v, \beta(w) \circ x)$ belongs to $S_f \times C_g$, then $(v, w \circ x)$, $(z, x)$, and $(z, x)$ are in $S_f \times C_g$, and $(y, w)$ is in $C_f \times S_g$. If $(b \circ b, \beta(w) \circ x)$ belongs to $C_f \times S_g$, then $(v, \beta(w) \circ x)$, $(z, x)$, and $(y, w)$ are in $C_f \times S_g$ too.

This lemma shows that the set of pairs of subforests appearing in the dynamic programming decomposition is the same as for the usual tree alignment algorithm [19]. We now determine the exact number of elementary operations involved in the computation.

**Lemma 10** Let $A$ be a tree.

1) the cardinality of $S_A$ is $n_A + \ell_A - 1$, and
\[ \sum_{f \in S_A} w_f = \ell_A + \sum_{v \in A} \frac{(d_v + 1)}{2} \]

2) the cardinality of $C_A$ is $\sum_{v \in A} \frac{(d_v + 1)^2}{2}$, and
\[ \sum_{f \in C_A} w_f = \sum_{v \in A} \frac{(d_v + 2)}{3} \]

**Lemma 11** Let $A$ and $B$ be two trees. The number of operations necessary to compute $\Lambda_1(A, B)$ is proportional to
\[ \left( \sum_{v \in B} \frac{(d_v + 1)^2}{2} \right) \left( \ell_A + \sum_{v \in A} \frac{(d_v + 1)}{2} \right) + (n_A + \ell_A - 1) \sum_{v \in B} \frac{(d_v + 1)}{2} + (n_B + \ell_B - 1) \sum_{v \in A} \frac{(d_v + 1)}{2} \]

**Proof:** For each pair of subforests $(f, g) \in A \times B$, the number of operations needed to compute $\Lambda_1(f, g)$ is majorized by $5w_f + w_g$. From Lemma 9, the total number of operations needed to compute $\Lambda_1(A, B)$ is
\[ 5 \sum_{(f, g) \in S_A \times C_B \cup C_A \times S_B} w_f + w_g \]
which is
\[ 5|C_B| \sum_{f \in S_A} w_f + |S_A| \sum_{g \in C_B} w_g + |S_B| \sum_{f \in C_A} w_f + |C_A| \sum_{g \in S_B} w_g \]
Applying Lemma 10 gives the result.

Now we can state the worst-case complexity of the algorithm.

**Theorem 8** Let $A$ and $B$ be two trees whose maximum degree is respectively $d_A$ and $d_B$. Then the number of scores to be computed is $O(n_A n_B (d_A + d_B))$ and the number of operations needed to compute them is $O(n_A n_B (d_A + d_B)^2)$.

**Proof:** From Lemma 10 we get for each tree $T$
\[ |S_T| \leq 2n_T, \quad |C_T| \leq \frac{4(n_T + 2)}{3} \]
\[ \sum_{f \in S_T} w_f \leq 2n_T d_T, \quad \sum_{f \in C_T} w_f \leq \frac{n_T d_T}{2} \]
Putting this in Lemma 9 and Lemma 11 gives the result.

Hence the worst-case complexity of the algorithm is in $O(n^3)$, which concludes the proof of Theorem 6.

D. Average-case complexity

We experimentally estimated the average complexity of the algorithm by randomly generating large trees. Thanks to the GenRGen software [25], 1000 trees of each size $n = 50, 150, 200, 250, \ldots, 2000$ were generated uniformly and randomly, giving 500 pairs of random trees for each size. Then the number of operations needed by the algorithm was computed for each pair, according to Lemma 11, and its mean value was computed within each of the 41 different sizes (including size 0). Results are given in the graph of Figure 6.
Fig. 4. RNA-like arc-annotated sequences for the example of Figure 2.

1. \( A_1(\alpha(u), \beta(u)) = \min \left\{ \omega_{\text{mm}}(\alpha, \beta) + A_1(u, w) - \text{arc-(mis)match between } \alpha \text{ and } \beta \right\} \)
   \[ \begin{align*}
   \omega_{\text{mm}}(\alpha, \beta) &= \min \{ A_1(y, w) + A_1(z, \nu) | y \circ z = \alpha(u) \circ v \} - \text{arc-removing of } \beta \\
   \omega_{\nu}(\alpha) &= A_1(u, y) + A_1(\nu, z) | y \circ z = \beta(u) \circ v \} - \text{arc-removing of } \alpha
   \end{align*} \]

2. \( A_1(\alpha(u) \circ v, \beta(u) \circ x) = \min \left\{ \omega_{\text{mm}}(\alpha, \beta) + A_1(u, w) + A_1(v, x) - \text{arc-(mis)match between } \alpha \text{ and } \beta \right\} \)
   \[ \begin{align*}
   \omega_{\text{mm}}(\alpha, \beta) &= \min \{ A_1(y, w) + A_1(z, \nu) | y \circ z = \alpha(u) \circ v \} - \text{arc-removing of } \beta \\
   \omega_{\nu}(\alpha) &= A_1(u, y) + A_1(\nu, z) | y \circ z = \beta(u) \circ v \} - \text{arc-removing of } \alpha
   \end{align*} \]

3. \( A_1(b \circ v, \beta(u) \circ x) = \min \left\{ \omega_{\text{mm}}(b, \nu) + A_1(u, w) + A_1(v, x) - \text{arc-(mis)match between } \alpha \text{ and } \beta \right\} \)
   \[ \begin{align*}
   \omega_{\text{mm}}(b, \nu) &= \min \{ A_1(y, w) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \beta \\
   \omega_{\nu}(\beta, b) &= A_1(u, y) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \alpha
   \end{align*} \]

4. \( A_1(\alpha(u) \circ v, b \circ x) = \min \left\{ \omega_{\text{mm}}(b, \nu) + A_1(u, w) + A_1(v, x) - \text{arc-(mis)match between } \alpha \text{ and } \beta \right\} \)
   \[ \begin{align*}
   \omega_{\text{mm}}(b, \nu) &= \min \{ A_1(y, w) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \beta \\
   \omega_{\nu}(\beta, b) &= A_1(u, y) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \alpha
   \end{align*} \]

5. \( A_1(b \circ v, b \circ x) = \min \left\{ \omega_{\text{mm}}(b, \nu) + A_1(u, w) + A_1(v, x) - \text{arc-(mis)match between } \alpha \text{ and } \beta \right\} \)
   \[ \begin{align*}
   \omega_{\text{mm}}(b, \nu) &= \min \{ A_1(y, w) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \beta \\
   \omega_{\nu}(\beta, b) &= A_1(u, y) + A_1(\nu, z) | y \circ z = b \circ v \} - \text{arc-removing of } \alpha
   \end{align*} \]

TABLE IV
Algorithm for the Align(Nested, Nested, II) \( \rightarrow \) Nested and Align(Nested, Nested, III) \( \rightarrow \) Nested problems.

This table shows the recurrence relations for the Align(Nested, Nested, II) \( \rightarrow \) Nested problem (Theorem 6). \( A_1 \) denotes the alignment score, that is the optimal cost for the pair of subforests. \( y \) and \( z \) are (possibly empty) closed subforests. Recurrence relations when at least one arc-annotated sequence is empty, with length 0, are omitted. In this case, the mapping is composed by a series of deletion operations applied on the non-empty arc-annotated sequence. Since edit model II is a sub-model of edit model III, this algorithm also solves the Align(Nested, Nested, II) \( \rightarrow \) Nested problem. For that, it is enough to remove all rules concerning arc-breaking operations.

tRNA ALA \( \rightarrow \) tRNA LEU

\[
\begin{align*}
\text{tRNA ALA / tRNA LEU}
\end{align*}
\]

\[
\begin{align*}
\text{tRNA ALA / tRNA LEU}
\end{align*}
\]

D. desulfuricans RNase P \( \rightarrow \) tRNA/C. jejuni RNase P RNA

\[
\begin{align*}
\text{D. desulfuricans RNase P RNA / C. jejuni RNase P RNA}
\end{align*}
\]

Fig. 5. Gardenia alignments: two tRNA genes from E. coli (above), and D. desulfuricans and C. jejuni RNase P RNAs (below). * indicates conserved positions. Alignments obtained with RNAforester are identical.
We carried out two interpolation methods on these data: polynomial interpolation and least squares (with the Maple function CurveFitting[Interactive]). We made the hypothesis that the complexity would be between \(O(n^2)\) and \(O(n^4)\), and would possibly contain a \((\log n)^k\) factor. Our results strongly suggest that the average complexity is in \(\theta(n^2)\). Indeed, the far best fit is got with \(f(n) = 22.09717440n^2 - 67.224600n\), computed by polynomial interpolation on three experimental values. The maximum relative error between the values of this function and the 48 remaining experimental values is less than \(6.10^{-3}\). Intuitively, this result seems natural since the average degree of an inner node in a random tree is less than 2. Indeed, the number of trees of size \(n + 1\) is the Catalan number \(C_n = \frac{1}{n+1} \binom{2n}{n}\) and the number of trees of size \(n + 1\) having \(k\) leaves is the Narayana number \(N(n,k) = \frac{1}{n-k} \binom{n}{k} \binom{n}{k-1}\). The average number of leaves in a random tree is \(\sum_k N(n,k)/C_n = \frac{n+2}{2}\). It remains \(\frac{n}{n+1}\) inner nodes in average for \(n\) edges, hence the result.

**E. Application to RNA comparison**

From a historical perspective, RNA secondary structures, corresponding to \textsc{nested} sequences, were first encoded by \textsc{nest}ed trees [24, 26, 30] provided with edit operations of model I. Figure 7 gives such an example. The main limitation of model I is that the evolutionary operations are not expressive. Indeed, there are some basic modifications in RNA structures that cannot be directly translated into a tree operation. For example, when comparing two RNA structures, it often happens that two nucleotides are paired in one structure and get unpaired in the other one. A likely explanation is that one of the two nucleotides has been mutated, so that they can be paired in the first structure but not in the second one. In model I, no single operation can represent this simple evolutionary event: this should be done by deleting the base pair, then inserting two new nucleotides.

Models II and III are more suitable for RNA structure comparison, since they allow for arc-altering and arc-breaking operations. As mentioned in Sections IV and V, the algorithmic complexity of the edition problem of two arc-annotated sequences is a pitfall, since the \textsc{edit} (\textsc{nested}, \textsc{nest}ed) problem is \textsc{np}-complete. To circumvent this difficulty, some authors have developed sequence-oriented algorithms [2]. The comparison is basically done on the nucleic sequence while trying to incorporate information on arcs in the comparison process. Another line of work is focused on tree comparison by partitioning the structures into macroscopic modules, such as stems [13] or multiloops [1].

A thorough review of recently published tools for RNA comparison is beyond the scope of this paper. We will not compare our algorithm to all state-of-the-art programs. We discuss here in further detail two program tools that address the comparison problem at the same level as us: they allow for arc-altering and arc-breaking operations and do not partition structures into macroscopic modules. In [16], Jiang et al. considered the \textsc{edit} (\textsc{crossing}, \textsc{nest}ed) problem (corresponding to \textsc{align} (\textsc{crossing}, \textsc{nest}ed, \textsc{ill}) \rightarrow \textsc{unlimited} in our hierarchy) and proposed a polynomial time algorithm for a restricted score scheme. Only arc-match, arc-mismatch and arc-breaking operations are explicitly required concerning arcs. Every arc-altering operation is treated as an arc-breaking operation plus a base deletion, and every arc-removing operation is treated as an arc-breaking plus two base deletions. This approach has some limitations. The first one comes from the edit model itself. The permissivity of the comparison model authorizes alignments with several arcs incident from the same position in the superstructure. Figure 8 shows such an example. Some base-pairings that seem unrelated may be associated and matched in the alignment. The restriction on the score scheme also changes the nature of the arc-removing operation. Deleting a base-pairing is no longer treated as a single evolutionary event, but as a series of three independent evolutionary events. It can lead to non relevant edit scripts, such as depicted in Figure 9.

The other program is the widely-distributed RNAforester software [14], that is part of the Vienna package [15]. The
algorithm is based on the tree alignment of [18]. The authors use a clever tree-based representation of RNA structures to incorporate arc-altering and arc-breaking operations. Each pair of nucleotides is encoded by three nodes: an inner one, called a P-node, which represents the arc and two leaves that represent the nucleotides (see Figure 7). Thus, the arc-breaking operation consists in deleting the P-node, and the arc-altering operation consists in deleting the P-node and one of its children. It means that the encoding suffers from the same restriction in terms of relations between the cost of edit operations as the previous approach. The introduction of P-nodes has also a hidden impact on the conformation of the supersequence. It authorizes to mix up base pairings, as described in Figure 8 in an unexpected way. Figure 10 shows the alignment supertree for sequences 1 and 2 of Figure 8. The supertree is based on the tree alignment of [18]. The authors have briefly discussed how to apply it to the problem of RNA secondary structure comparison. The method shows that there exists no polynomial time algorithm for arbitrary values of parameters (such as usual dynamic programming algorithms do). We, thus, conjecture that both ALIGN(NEST,NEST,II) → CROS and ALIGN(CROS,*,III) → CROS problems are NP-complete.

In the last section, we have also provided a polynomial time algorithm to compare arc-annotated sequences of NESTED type with arc-altering and arc-breaking operations, whereas when considering other models, the problem is NP-complete.

We have briefly discussed how to apply it to the problem of RNA secondary structure comparison. The method shows promising results in comparison with other existing programs that address the comparison problem at the level of individual bases and base pairings.

VI. CONCLUSION

In this paper, we have proposed and studied a new framework for comparing arc-annotated sequences, namely the alignment hierarchy. We think that this study is relevant both from theoretical and practical perspectives. We gave a new NP-completeness result, that enhances understanding of the complexity of arc-annotated sequences comparison. This result sheds a new light on the border between tractability and untractability when dealing with arc-annotated sequences—especially of CROSSING type. These results, combined with the ones derived from EDIT and LAPCS comparison models, have almost filled the complexity table of the alignment hierarchy.

As illustrated in Table II, there still exist some open questions for the model III. But we can notice that the edit model III reduces to the edit model II when the cost of any arc-breaking is arbitrary high. As a consequence, the NP-completeness of ALIGN(NEST,NEST,II) → CROS and of ALIGN(CROS,*,II) → CROS shows that there exists no polynomial time algorithm for arbitrary values of parameters (such as usual dynamic programming algorithms do). We, thus, conjecture that both ALIGN(NEST,NEST,III) → CROS and ALIGN(CROS,*,III) → CROS problems are NP-complete.

In the last section, we have also provided a polynomial time algorithm to compare arc-annotated sequences of NESTED type with arc-altering and arc-breaking operations, whereas when considering other models, the problem is NP-complete.

We have briefly discussed how to apply it to the problem of RNA secondary structure comparison. The method shows promising results in comparison with other existing programs that address the comparison problem at the level of individual bases and base pairings.

REFERENCES


<table>
<thead>
<tr>
<th>tRNAs (80nt) – [14]</th>
<th>gardenia</th>
<th>RNAforester</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. coli</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RNase P RNAs (350 nt) – [7]</td>
<td>0.23</td>
<td>6</td>
</tr>
<tr>
<td>D. desulfuricans and C. jejuni</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>IRES RFAM 00549 (600nt) – [12]</td>
<td>2.3</td>
<td>25</td>
</tr>
<tr>
<td>16S RNAs (1500 nt) – [20]</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>B. subtilis and A. pernix</td>
<td>0.4</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig. 9. Constraint on edit operation weights. This figure shows two alternative alignments for the same pair of RNA structures. The first alignment corresponds to the application of a single edit operation: one arc-removing. The second alignment results from three distinct evolutionary operations: one arc-breaking and two base-deletions. For comparison models with a restricted score scheme, such as [16] or [14], these two alignments are equivalent, since they have the same score, whereas the first alignment is more relevant.

Fig. 10. P-node supertree for sequences 1 and 2 of Figure 8. The supertree induces the same optimal alignment as the ALIGN(NEST,NEST) → UNLIM scheme in figure 8. Positions marked with * in the alignment are pointed with arrows in the supertree.

Fig. 11. Comparison of execution times for Gardenia and RNAforester. All times are in seconds, on a bi-processor 3Ghz, 6GB RAM.


