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.

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 LaBri,UMR 5800 CNRS – Université Bordeaux I. H. Touzet (corresponding author) is with the LIFL, UMR 8022 CNRS – Université Lille 1 and INRIA. A preliminary version of some of this work appears as an extended abstract in the Proceedings of the SPIRE 2006 conference [6].

II. Edition models for arc-annotated sequences

Given a finite alphabet Σ, an arc-annotated sequence is defined by a pair (S, P), where S is a string of Σ^* 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.

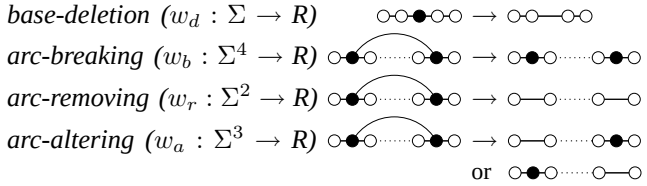
Since we focus here on structure comparison, we do not consider PLAIN sequences, which do not carry any structural information. In the remaining of this paper, we shall only deal with sequences of type NESTED, CROSSING and UNLIMITED.

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* ($w_m : \Sigma^2 \to$ R), *base-mismatch* ($w_m : \Sigma^2 \to$ R), *arc-match* ($w_{am} : \Sigma^4 \to$ R), *arc-mismatch* ($w_{am} : \Sigma^4 \to$ R).

• Deletion operations, inducing deletion of bases and/or of arcs:

base-deletion ($w_d : \Sigma \to R$)
arc-breaking ($w_b : \Sigma^4 \to R$)
arc-removing ($w_r : \Sigma^2 \to R$)
arc-altering ($w_a : \Sigma^3 \to R$)



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-removings are allowed,
- **II** : the operations of model I and arc-alterings 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 $\texttt{cost}(u, v, K)$, 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 $\textsc{Edit}(u, v, K)$ problem. For each model $K \in \{\text{I}, \text{II}, \text{III}\}$, we also define the *ordering relation* $\trianglelefteq_K$: if u can be obtained from v by a series of deletion and substitution operations of the model K, then $u \trianglelefteq_K v$. Provided with these notations, we propose to extend the notion of subsequence on plain sequences to arc-annotated sequences as follows.

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

Given three arc-annotated sequences u, v and w such that $w \trianglelefteq_K u$ and $w \trianglelefteq_K v$, 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: $\texttt{cost}(u, w, K) + \texttt{cost}(v, w, K)$.

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 $K \in \{\text{I}, \text{II}, \text{III}\}$, solving the $\textsc{Edit}(u, v, K)$ problem is equivalent to finding a common K-subsequence w of u and v of minimal cost.*

Proof: ($\Rightarrow$) Let w be a common K-subsequence of u and v. By definition, we have $w \trianglelefteq_K u$ and $w \trianglelefteq_K v$. 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 $\texttt{cost}(u, w, K) + \texttt{cost}(v, w, K)$. Thus the edit distance is lower than or equal to the cost of w.

($\Leftarrow$) 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 $K \in \{\text{I}, \text{II}, \text{III}\}$, u is said to be a K-supersequence of v if, and only if, $v \trianglelefteq_K u$.*

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 $u \trianglelefteq_K w$ and $v \trianglelefteq_K w$. The cost of w is defined as $\texttt{cost}(w, u, K) + \texttt{cost}(w, v, K)$.

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 $K \in \{\text{I}, \text{II}, \text{III}\}$, there exists a common K-subsequence of u and v of cost α iff there exists a common K-supersequence of u and v of the same cost.*

Proof: ($\Rightarrow$) 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 $\phi(i, R, S)$ (resp. $\phi(i, R, T)$) 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:

$$\begin{aligned} V = \;& S_1\ T_1\ R[1]\ S_2\ T_2\ R[2] \ldots S_n\ T_n\ R[n]\ S_{n+1}\ T_{n+1} \\ W = \;& \{(\psi_u(i), \psi_u(j));\ (i, j) \in P\} \\ & \cup \{(\psi_v(i), \psi_v(j));\ (i, j) \in Q\} \end{aligned}$$

where n is the length of R and S_i (resp. T_i) denotes $S[\phi(i-1, R, S)+1..\phi(i, R, S)-1]$ (resp. $T[\phi(i-1, R, T)+1..\phi(i, R, T)-1]$). By convention, we have $\phi(0, R, S) = \phi(0, R, T) = 0$ and $\phi(n+1, R, S)$ (resp. $\phi(n+1, R, T)$) 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

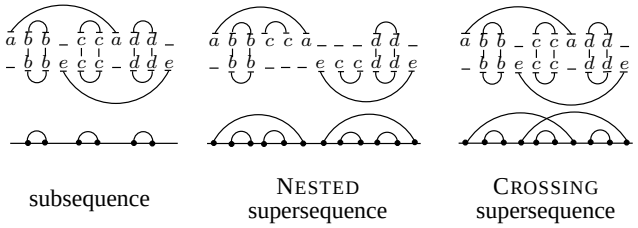


Fig. 1. Comparison of the optimal common subsequence and the optimal common supersequences for a pair of arc-annotated sequences, $u = abbccadd$ and $v = bbeccdde$. The optimal common subsequence (first picture) is derived from u and v with two arc-removing operations. The optimal common NESTED supersequence requires four arc-removing operations (second picture). In this example, it is necessary to allow crossing arcs in the supersequence to get the same cost as for the subsequence (third picture).

$\texttt{cost}(x, u, K) = \texttt{cost}(v, w, K)$. Indeed, in order to obtain u from x, or w from v, one just has to delete all bases and arcs originated from v without being in w. By a similar reasoning, we can show that $\texttt{cost}(x, v, K) = \texttt{cost}(u, w, K)$. It follows that $\texttt{cost}(x, u, K) + \texttt{cost}(x, v, K) = \alpha$.

($\Leftarrow$) The reverse direction is similar. The common subsequence is obtained as the intersection of u and v, instead of considering the union as in the previous case. Let $u = (S, P)$, $v = (T, Q)$ and $x = (V, W)$ be three arc-annotated sequences such that x is a common K-supersequence of u and v. The subsequence $w = (R, U)$ is defined as follows: R is the common subsequence composed of conserved positions between S and T in the mapping induced by x and

$$\begin{aligned} U = \quad & \{(\phi(i, R, S), \phi(j, R, S)); (i, j) \in P\} \\ & \cap \{(\phi(k, R, T), \phi(l, R, T)); (k, l) \in Q\} \end{aligned}$$

We have $\texttt{cost}(x, u, K) = \texttt{cost}(v, w, K)$ and $\texttt{cost}(x, v, K) = \texttt{cost}(u, w, K)$. Hence $\texttt{cost}(u, w, K) + \texttt{cost}(v, w, K) = \alpha$. ■

In Lemma 2, it is worth to notice that the type of the common supersequence is not guaranteed to be the same as the type of the common subsequence. Figure 1 illustrates such an example. The edit script associated with the optimal subsequence (which is of NESTED type) has a smaller cost than the edit script associated with the optimal NESTED supersequence. Indeed, when constructing the set of arcs of the common K-supersequence of u (above) and v (below), it is likely to create crossing or multiple arcs that are absent from the initial sequences. In general, when considering arc-annotated sequences of NESTED types, searching for a common NESTED supersequence is more restrictive than searching for a common subsequence. In example of Figure 1, it is necessary to authorize crossing arcs in the supersequence to get the same cost as for the EDIT problem. This observation gives rise to a family of new problems, which we call the *alignment hierarchy*.

Definition 3 (Alignment hierarchy) *Given three types of sequence A, B and C of $\{\text{NEST}, \text{CROS}, \text{UNLIM}\}$ and an edit model $K \in \{\text{I}, \text{II}, \text{III}\}$, the $\text{ALIGN}(A, B, K) \rightarrow C$ problem is defined as:*

INPUT: *two arc-annotated sequences u and v of type A and B respectively.*

OUTPUT: *a common K-supersequence w of type C of minimum cost.*

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 $K \in \{\text{I}, \text{II}, \text{III}\}$, 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 $\text{ALIGN}(A, B, \text{I}) \rightarrow \text{UNLIM}$ and $\text{ALIGN}(A, B, \text{I}) \rightarrow \text{CROS}$ 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 CROSSING and UNLIMITED types do not have in common. ■

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

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 CROSSING 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 $\text{ALIGN}(\text{NEST}, \text{NEST}, \text{I}) \rightarrow \text{NEST}$, 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.

$A \times B \to C$	EDIT	model I
NEST × NEST → NEST		$O(n^4)$ – Jiang [18]
NEST × NEST → CROS NEST × NEST → UNLIM	×	$O(n^3 \log(n))$ – Klein [21]
CROS × NEST → CROS CROS × NEST → UNLIM	×	$O(n^3 \log(n))$ – Ma [24]
CROS × CROS → CROS CROS × CROS → UNLIM	×	**NP**-complete – Ma [24]
UNLIM× NEST → UNLIM	×	$O(n^3 \log(n))$ – Lemma 4
UNLIM× CROS → UNLIM	×	**NP**-complete – Ma [24]
UNLIM×UNLIM → UNLIM	×	**NP**-complete – Ma [24]

TABLE I
ALIGNMENT HIERARCHY FOR THE EDIT MODEL I.

According to Lemma 3, the ten problems of the hierarchy reduce to seven distinct instances. We indicate entries that can also be formulated as edit problems with × in the second column (see Theorem 1). 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-annotated 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 $w \trianglelefteq_{\text{II}} v$ and $w \trianglelefteq_{\text{II}} u$.*

Proof: The proof relies on the following property: Let $u' = (S, P)$ and $v' = (T, Q)$ be two arc-annotated sequences. We have $u' \trianglelefteq_{\text{II}} v'$ iff S is a common arc-preserving subsequence of T considering the implicit mapping – denoted M – from u' to v' induced by $u' \trianglelefteq_{\text{II}} v'$.
($\Rightarrow$) 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.
($\Leftarrow$) 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 $u \trianglelefteq_{\text{II}} w$ 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 $w \trianglelefteq_{\text{II}} v$, which implies $u \trianglelefteq_{\text{II}} v$ by transitivity of $\trianglelefteq_{\text{II}}$.

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 $U = T[1..i-1] \circ T[i+1..|T|]$ and $R = Q$. We have $w \trianglelefteq_{\text{II}} v$ 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 $u \trianglelefteq_{\text{II}} w$.

In the other cases, $T[i]$ is a paired base. Let k be the position of its partner (i.e. $(i, k) \in Q$).
– 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 $U = T[1..i-1] \circ T[i+1..|T|]$ and $R = Q - \{(i, k)\}$. We have $w \trianglelefteq_{\text{II}} v$ 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 $u \trianglelefteq_{\text{II}} w$.
– 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 $u \trianglelefteq_{\text{II}} w$. ■

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(NESTED, NESTED), that 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.

$A \times B \rightarrow C$	EDIT	model II
NEST × NEST → NEST		$O(n^4)$ – Theorem 3
NEST × NEST → CROS		**NP**-complete – Theorem 4
NEST × NEST → UNLIM	×	**NP**-complete – Lin [22]
CROS × NEST → CROS		**NP**-complete – Theorem 4
CROS × NEST → UNLIM	×	**NP**-complete – Evans [11]
UNLIM× NEST → UNLIM	×	
CROS × CROS → CROS		**NP**-complete– Theorem 4
CROS × CROS → UNLIM	×	**NP**-complete – Evans [11]
CROS ×UNLIM → UNLIM	×	
UNLIM×UNLIM → UNLIM	×	

TABLE II
ALIGNMENT HIERARCHY FOR EDIT MODEL II.

We indicate problems that can be formulated as edit distance problems in the second column. In these cases, known results stem from the LAPCS 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. $\max(|u|, |v|) = n$.

$A \times B \rightarrow C$	EDIT	model III
NEST × NEST → NEST		$O(n^4)$ – Theorem 6
NEST × NEST → CROS		
NEST × NEST → UNLIM	×	**NP**-complete – Blin [5]
CROS × NEST → CROS		
CROS × NEST → UNLIM	×	Max SNP-hard – Jiang [16]
UNLIM× NEST → UNLIM	×	
CROS × CROS → CROS		
CROS × CROS → UNLIM	×	Max SNP-hard – Jiang [16]
CROS ×UNLIM → UNLIM	×	
UNLIM×UNLIM → UNLIM	×	

TABLE III
ALIGNMENT HIERARCHY FOR EDIT MODEL III.

We indicate problems that can be formulated as edit distance problems in the second column. In these cases, known results stem from the general edit distance for the model III (Theorem 1). Other problems are specific to the ALIGN hierarchy and are introduced in this paper. Blank cells are for problems that are still open. Complexity results are indicated for two arc-annotated sequences u and v s.t. $\max(|u|, |v|) = n$.

C. ALIGN(NESTED, NESTED, II) → 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) → 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 ℓ.
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 ℓ ?

We initially notice that this problem is in **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 ℓ. In order to prove that it is **NP**-complete, we propose a polynomial reduction from the **NP**-complete problem MIS-3P [4]. The MIS-3P problem is also used in the polynomial reduction of the **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 a 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:

$$\begin{aligned} S &= \#^n\, S_1\, \#^n\, S_2\, \ldots\, \#^n\, S_n \\ T &= \#^n\, T_1\, \#^n\, T_2\, \ldots\, \#^n\, T_n \end{aligned}$$

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 $baaa$ 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

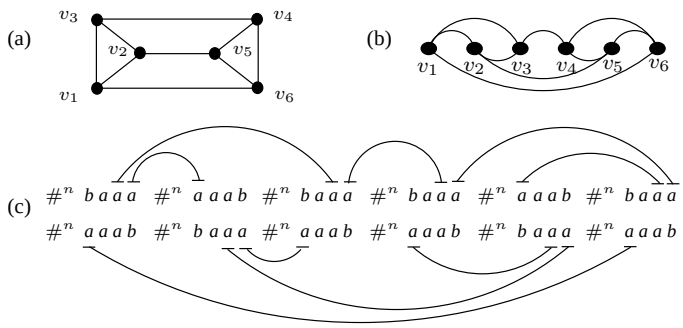


Fig. 2. Example of an align-construction for the proof of Theorem 4. The graph (a) is a cubic planar bridgeless connected graph of 6 vertices. The graph (b) is a two-page book embedding of the graph (a) such that, on each page, any vertex has a non-null degree. (c) The two arc-annotated sequences of 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 $3\frac{n}{2}$. 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: $w_d(b) = 2$, $w_d(\#) = 6$, $w_d(a) = 1$, $w_a(a,a,a) = 1.5$, $w_r(a,a) = 2$. As a matter of fact, the proofs of Lemmas 5, 6, 7, 8 are still valid with any combination of parameters that fullfils these two inequalities: $3w_a(a,a,a) + 2w_d(b) < 3w_r(a,a) + 3w_d(a)$ and $w_r(a,a) + 3w_d(a) < w_a(a,a,a) + 2w_d(b)$.

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 $\#^n U_1 \ldots \#^n U_n$ where for each $i \in 1..n$, $U_i = aaabaaa$ or $U_i = baaab$.*

Proof: It is easy to verify that $(\#^n aaabaaa)^n$ is a common II-supersequence whose cost is lower than or equal to $n(\frac{3}{2}w_r(a,a) + 3w_d(a)) = 6n$. This observation ensures that any optimal supersequence is of the form $U = \#^n U_1 \ldots \#^n U_n$, where $U_i \in \{a,b\}^*$. Indeed, assume that an optimal supersequence contains more than n^2 occurrences of the $\#$ 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 $nw_d(\#) = 6n$.

By construction, each U_i is a supersequence of $aaab$ and $baaa$. There are five candidate strings: $aaabaaa$, $baaab$, $baaaab$, $baaaaab$ and $baaaaaab$ (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 $U_i = baaaab$. We suppose w.l.o.g. that $S_i = aaab$ and $T_i = baaa$. 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. of 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 $aaabaaa$ does not increase the cost of the alignment.

1. $U[j]$ does not appear 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, T_i is derived from U_i by an arc-removing and a base deletion of b. The associated cost is $w_a(a,a,a) + 2w_r(a,a) + 2w_d(b) = 9.5$. On the other hand, S_i is derived from $aaabaaa$ by two arc-removings and one base deletion of a, whereas T_i is derived from $aaabaaa$ by one arc-removing and two base-deletions of a. The total cost is $3w_r(a,a) + 3w_d(a) = 9$.

2. $U[j]$ does not appear 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-alterings and a base-deletion of b, T_i is derived from U_i by an arc-removing and a base-deletion of b. The associated cost is $2w_a(a,a,a) + w_r(a,a) + 2w_d(b) = 9$. On the other hand, S_i is derived from $aaabaaa$ by an arc-altering, an arc-removing

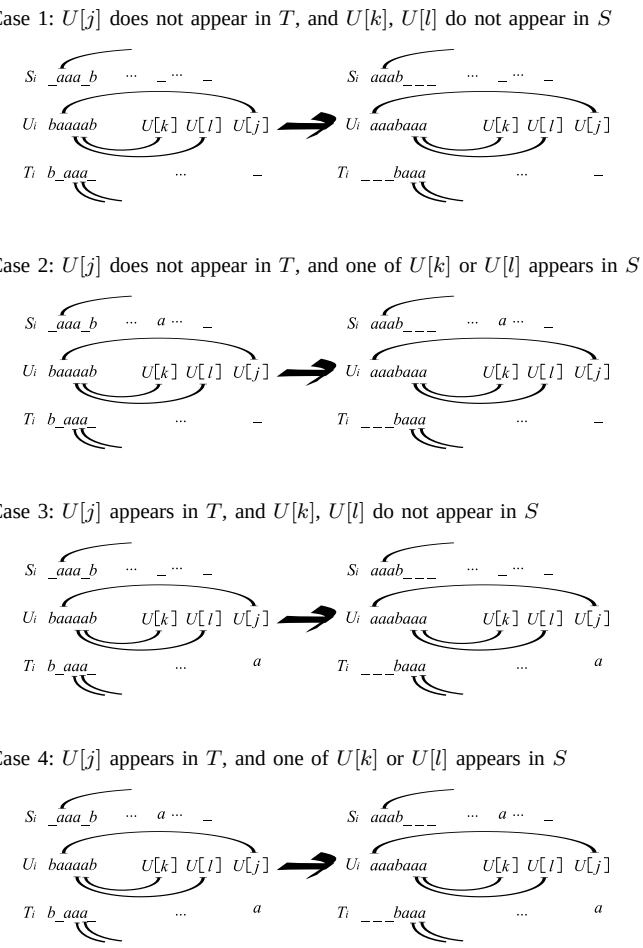


Fig. 3. Four first cases for the replacement of U_i when $U_i = baaaab$ in Proof of Lemma 5

and a base-deletion of a, whereas T_i is derived from $aaabaaa$ by an arc-removing and two base-deletions of a. The total cost is $2w_r(a,a) + w_a(a,a,a) + 3w_d(a) = 8.5$.

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 $w_r(a,a) + 2w_a(a,a,a) + 2w_d(b) = 9$. On the other hand, S_i is derived from $aaabaaa$ by two arc-removings and a base-deletion of a, whereas T_i is derived from $aaabaaa$ by an arc-altering and two base-deletions of a. The total cost is $2w_r(a,a) + w_a(a,a,a) + 3w_d(a) = 8.5$.

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-alterings, 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 $3w_a(a,a,a) + 2w_d(b) = 8.5$. On the other hand, S_i is derived from $aaabaaa$ 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 $w_r(a,a) + 2w_a(a,a,a) + 3w_d(a) = 8$.

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

The reasoning is similar for $baaaaab$ and $baaaaaab$. ■

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 be both 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 $3pw_a(a,a,a) + 3(\frac{n}{2} - p)w_r(a,a) + 3(n-p)w_d(a) + 2pw_d(b)$, where p is the number of segments of w of type $baaab$.*

Proof: By construction, the supersequence w contains $3\frac{n}{2}$ 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 $aaabaaa$, and $3\frac{n}{2} - 3p$ arcs connecting two segments $aaabaaa$. 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 $aaabaaa$ gives rise to an arc-removing, whereas an arc between a segment $baaab$ and a segment $aaabaaa$ gives rise to an arc-altering. It follows that the total cost of deletion operations on arcs is $3pw_a(a,a,a) + 3(\frac{n}{2} - p)w_r(a,a)$.

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

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 $\ell = 3kw_a(a,a,a) + 3(\frac{n}{2} - k)w_r(a,a) + 3(n-k)w_d(a) + 2kw_d(b)$, where u and v are arc-annotated sequences of* NESTED *type resulting from an align-construction of G and $n = |V|$.*

Proof: ($\Rightarrow$) Let $V' \subseteq V$ such that $|V'| \geq k$ and V' is an independent set. Let $w = (U,R)$ be the arc-annotated sequence of CROSSING type defined by $U = \#^n U_1 \ldots \#^n U_n$, where $\forall v_i \in V'$, $U_i = baaab$ and $\forall v_i \in V - V'$, $U_i = aaabaaa$. By Lemma 7, the cost of the alignment induced by w is $3|V'|w_a(a,a,a) + 3(\frac{n}{2} - |V'|)w_r(a,a) + 3(n-|V'|)w_d(a) + 2|V'|w_d(b)$. Since by hypothesis $|V'| \geq k$, this cost is majored by $3kw_a(a,a,a) + 3(\frac{n}{2} - k)w_r(a,a) + 3(n-k)w_d(a) + 2kw_d(b)$, which equals ℓ.

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

One can remark that the arc-annoted 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 $\#$ 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 III

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 hierachy from known results on the *general edit distance* [5], [16] with Theorem 1. As illustrated in Table III, the complexity of ALIGN(NEST, NEST, III) $\rightarrow$ {NEST, CROS}

and of ALIGN(CROS, {NEST, CROS}, III) → CROS only is still to elucidate. We solve ALIGN(NEST, NEST, III) → NEST.

Theorem 6 ALIGN(NESTED, NESTED, III) → NESTED *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 NESTED arc-annotated sequences.

A. Notations

We write $\alpha(f)$ a NESTED arc-annotated sequence, or equivalently a tree, that is composed of a root α and a subforest f. A NESTED arc-annotated sequence is defined recursively by concatenating a tree and an arc-annotated sequence. Let $\circ$ be a binary operator that concatenates two arc-annotated sequences. $\alpha(u) \circ v$ 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 in Σ. $b \circ u$ 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 $\mathtt{Al}$. 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 $\mathtt{Al}$ 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, $\mathtt{Al}(f, g)$ 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, $\mathcal{S}(f, g)$ denotes an optimal common supersequence of f and g.

Case 2. Let $S = \mathcal{S}(\alpha(u) \circ v, \beta(w) \circ x)$, 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 $\beta(\mathcal{S}(u, w)) \circ \mathcal{S}(v, x)$. The resulting cost is $w_{am}(\alpha, \beta) + \mathtt{Al}(u, w) + \mathtt{Al}(v, x)$.
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 $i < k$ and $j < k$: This configuration corresponds to an arc-removing of α. Indeed, k is the first position in S corresponding to a base present in $\beta(w) \circ x$. So α and u have no counterpart in $\beta(w) \circ x$, and S is obtained as $\alpha(u) \circ \mathcal{S}(v, \beta(w) \circ x)$. The resulting cost is $w_r(\alpha) + \mathtt{Al}(u, \varepsilon) + \mathtt{Al}(v, \beta(w) + x)$.

- If $i < k$ and $l < j$, 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 $\beta(w) \circ x = y \circ b \circ z$. S is obtained as $\alpha(\mathcal{S}(u, y)) \circ \mathcal{S}(v, z)$. The resulting cost is $w_a(\alpha, b) + \mathtt{Al}(u, y) + \mathtt{Al}(v, z)$. In the latter case, let y be the largest subforest of $\beta(w) \circ x$ ending at position j in S, and let z be the largest subforest of $\beta(w) \circ x$ starting at position j in S. S is obtained as $\alpha(\mathcal{S}(u, y)) \circ \mathcal{S}(v, z)$. The resulting cost is $w_r(\alpha) + \mathtt{Al}(u, y) + \mathtt{Al}(v, z)$.
If $k < i$, then this configuration is exactly equivalent to $i < k$ when we exchange $\alpha(u) \circ v$ and $\beta(w) \circ x$.

Case 3. Let $S = \mathcal{S}(b \circ v, \beta(w) \circ x)$, 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 $b \circ v$: This configuration corresponds to an arc-breaking of β. Let b_2 be the base of v occurring at position l in the supersequence S. b_2 is necessarily a single base, since multiple incident arcs are prohibited in S. Furthermore, there exists no arc in v spanning b_2, otherwise we would have crossing arcs in S. So there are two (possibly empty) subforests y and z such that $v = y \circ b_2 \circ z$. The optimal supersequence S is obtained as $\beta(\mathcal{S}(y, w)) \circ \mathcal{S}(z, x)$. The resulting cost is $w_b(\beta, b, b_2) + \mathtt{Al}(y, w) + \mathtt{Al}(z, x)$.

- If $i = k$ and l does not correspond to any position in $b \circ v$: This configuration corresponds to an arc-altering of β, whose 5' base is aligned with b. Let y be subforest of v starting at position $i + 1$ and ending at position k in S, and let z be the largest subforest 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 $\beta(\mathcal{S}(y, w)) \circ \mathcal{S}(z, x)$. The resulting cost is $w_a(\beta, b) + \mathtt{Al}(y, w) + \mathtt{Al}(z, x)$.

- 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 $b \circ v$ and that has no counterpart in $\beta(w) \circ x$.

The remaining part of S is obtained as $\mathcal{S}(v, \beta(w) \circ x)$. The total resulting cost is $w_d(b) + \mathtt{Al}(v, \beta(w) \circ x)$.

- If $k < i$ and l is aligned with some position of $b \circ v$: This configuration corresponds to an arc-altering of β, 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 $\beta(\ \mathcal{S}(y, w)) \circ \mathcal{S}(z, x)$. The resulting cost is $w_a(\beta, b_2) + \mathtt{Al}(y, w) + \mathtt{Al}(z, x)$.

- If $k < i$ and l does not correspond to any position in $b \circ v$: This configuration corresponds to an arc-removing of β. 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 pistion $l + 1$. 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 $\beta(\mathcal{S}(y, w)) \circ \mathcal{S}(z, x)$. The resulting cost is $w_r(\beta) + \mathtt{Al}(y, w) + \mathtt{Al}(z, x)$. ■

C. Worst-case complexity

Let us state some definitions and notations. Let f be a forest. We denote n_f its number of nodes, ℓ_f its number of leaves, and w_f its width, that 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 subforest that contains the rightmost tree of a complete subforest. We write $\mathcal{S}_f$ for the set of suffix subforests and subtrees of f, and $\mathcal{C}_f$ for the set of closed 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 $\mathcal{S}_f \times \mathcal{C}_g \cup \mathcal{C}_f \times \mathcal{S}_g$.*

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 $(\alpha(u) \circ v, \beta(w) \circ x)$ belongs to $\mathcal{S}_f \times \mathcal{C}_g$, then (u, w), (u, y), (v, x), (v, z), (z, x) are in $\mathcal{S}_f \times \mathcal{C}_g$, and (y, w) is in $\mathcal{C}_f \times \mathcal{S}_g$. Similarly, if $(\alpha(u) \circ v, \beta(w) \circ x)$ belongs to $\mathcal{C}_f \times \mathcal{S}_g$, then (u, w), (y, w), (v, x), (z, x), (v, z) are in $\mathcal{C}_f \times \mathcal{S}_g$, and (u, y) is in $\mathcal{S}_f \times \mathcal{C}_g$.
Case 3: If $(b \circ v, \beta(w) \circ x)$ belongs to $\mathcal{S}_f \times \mathcal{C}_g$, then $(v, \beta(w) \circ x)$, (z, x) are in $\mathcal{S}_f \times \mathcal{C}_g$, and (y, w) is in $\mathcal{C}_f \times \mathcal{S}_g$. If $(b \circ v, \beta(w) \circ x)$ belongs to $\mathcal{C}_f \times \mathcal{S}_g$, then $(v, \beta(w) \circ x)$, (z, x), (y, w) are in $\mathcal{C}_f \times \mathcal{S}_g$ 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 $\mathcal{S}_A$ is $n_A + \ell_A - 1$, and*
$$\sum_{f \in \mathcal{S}_A} w_f \ = \ \ell_A + \sum_{v \in A} \binom{d_v + 1}{2}$$
2) *the cardinality of $\mathcal{C}_A$ is $\sum_{v \in A} \binom{d_v+1}{2}$, and*
$$\sum_{f \in \mathcal{C}_A} w_f \ = \ \sum_{v \in A} \binom{d_v + 2}{3}$$

Lemma 11 *Let A and B be two trees. The number of operations necessary to compute $\mathtt{Al}(A, B)$ is proportional to*

$$\begin{array}{l} \left(\sum_{v \in B} \binom{d_v+1}{2}\right) \left(\ell_A + \sum_{v \in A} \binom{d_v+1}{2}\right) \\ +(n_A + \ell_A - 1) \sum_{v \in B} \binom{d_v+2}{3} + (n_B + \ell_B - 1) \sum_{v \in A} \binom{d_v+2}{3} \\ + \left(\sum_{v \in A} \binom{d_v+1}{2}\right) \left(\ell_B + \sum_{v \in B} \binom{d_v+1}{2}\right) . \end{array}$$

Proof: For each pair of subforests $(f, g) \in A \times B$, the number of operations needed to compute $\mathtt{Al}(f, g)$ is majorized by a $5(w_f + w_g)$. From Lemma 9, the total number of operations needed to compute $\mathtt{Al}(A, B)$ is

$$5 \sum_{(f,g) \in \mathcal{S}_A \times \mathcal{C}_B \cup \mathcal{C}_A \times \mathcal{S}_B} w_f + w_g$$

which is

$$5(|\mathcal{C}_B| \sum_{f \in \mathcal{S}_A} w_f + |\mathcal{S}_A| \sum_{g \in \mathcal{C}_B} w_g + |\mathcal{S}_B| \sum_{f \in \mathcal{C}_A} w_f + |\mathcal{C}_A| \sum_{g \in \mathcal{S}_B} w_g)$$

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

$$\begin{array}{cc} |\mathcal{S}_T| \leq 2n_T \, , & |\mathcal{C}_T| \leq \frac{n_T(d_T+1)}{2} \, , \\ \sum_{f \in \mathcal{S}_T} w_f \leq 2n_T d_T \, , & \sum_{f \in \mathcal{C}_T} w_f \leq \frac{n_T d_T (d_T+1)}{2} \, . \end{array}$$

Putting this in Lemma 9 and Lemma 11 gives the result. ■

Hence the worst-case complexity of the algorithm is in $O(n^4)$, 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 GenRGenS software [25], 1000 trees of each size $n = 50$, 150, 200, 250, ..., 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.

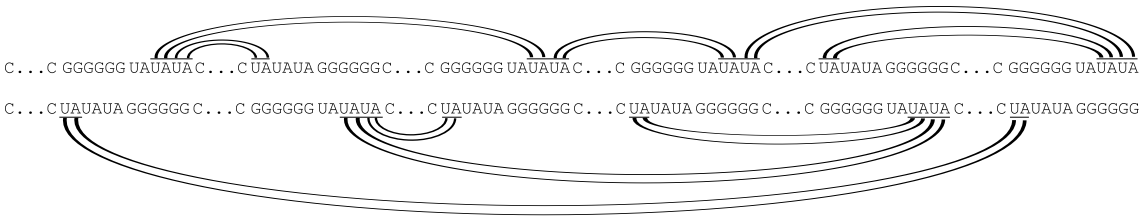


Fig. 4. RNA-like arc-annotated sequences for the example of Figure 2.

1. $\mathtt{Al}(\alpha(u), \beta(w)) = \min \begin{cases} w_{am}(\alpha,\beta) + \mathtt{Al}(u,w) - \text{arc-(mis)match between } \alpha \text{ and } \beta \\ w_r(\beta) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,\varepsilon) | y \circ z = \alpha(u)\} - \text{arc-removing of } \beta \\ w_r(\alpha) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(\varepsilon,z) |\ y \circ z = \beta(w)\} - \text{arc-removing of } \alpha \end{cases}$

2. $\mathtt{Al}(\alpha(u) \circ v, \beta(w) \circ x) = \min \begin{cases} w_{am}(\alpha,\beta) + \mathtt{Al}(u,w) + \mathtt{Al}(v,x) - \text{arc-(mis)match between } \alpha \text{ and } \beta \\ w_r(\beta) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) | y \circ z = \alpha(u) \circ v\} - \text{arc-removing of } \beta \\ w_r(\alpha) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) |\ y \circ z = \beta(w) \circ x\} - \text{arc-removing of } \alpha \\ w_a(\alpha,b) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) |\ y \circ b \circ z = \beta(w) \circ x\} - \text{arc-altering of } \alpha \\ w_a(\beta,b) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) |\ y \circ b \circ z = \alpha(u) \circ v\} - \text{arc-altering of } \beta \end{cases}$

3. $\mathtt{Al}(b \circ v, \beta(w) \circ x) = \min \begin{cases} w_d(b) + \mathtt{Al}(v, \beta(w) \circ x) - \text{base-deletion of } b \\ w_r(\beta) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) | y \circ z = b \circ v\} - \text{arc-removing of } \beta \\ w_a(\beta,b) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) |\ y \circ z = v\} - \text{arc-altering of } \beta \\ w_a(\beta,b_2) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) |\ y \circ b_2 \circ z = b \circ v\} - \text{arc-altering of } \beta \\ w_b(\beta,b,b_2) + \min\{\mathtt{Al}(y,w) + \mathtt{Al}(z,x) | y \circ b_2 \circ z = v\} - \text{arc-breaking of } \beta \end{cases}$

4. $\mathtt{Al}(\alpha(u) \circ v, b \circ x) = \min \begin{cases} w_d(b) + \mathtt{Al}(\alpha(u) \circ v, x) - \text{base-deletion of b} \\ w_r(\alpha) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) | y \circ z = b \circ x\} - \text{arc-removing of } \alpha \\ w_a(\alpha,b) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) |\ y \circ z = x\} - \text{arc-altering of } \alpha \\ w_a(\alpha,b_2) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) |\ y \circ b_2 \circ z = b \circ x\} - \text{arc-altering of } \alpha \\ w_b(\alpha,b,b_2) + \min\{\mathtt{Al}(u,y) + \mathtt{Al}(v,z) | y \circ b_2 \circ z = x\} - \text{arc-breaking of } \alpha \end{cases}$

5. $\mathtt{Al}(b \circ v, b_2 \circ x) = \min \begin{cases} w_d(b) + \mathtt{Al}(v, b_2 \circ x) - \text{base-deletion of } b \\ w_d(b_2) + \mathtt{Al}(b \circ v, x) - \text{base-deletion of } b_2 \\ w_m(b,b_2) + \mathtt{Al}(v,x) - \text{base-(mis)match between } b \text{ and } b_2 \end{cases}$

TABLE IV

ALGORITHM FOR THE ALIGN(NESTED, NESTED, II) → NESTED AND ALIGN(NESTED, NESTED, III) → NESTED PROBLEMS.

This table shows the recurrence relations for the ALIGN(NESTED, NESTED, II) → NESTED problem (Theorem 6). Al 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) → NESTED problem. For that, it is enough to remove all rules concerning arc-breaking operations.

tRNA ALA /tRNA LEU

```
(((((((..((((........) ))).(((((.......)))))..  .   . .  ((((((.......)))))))))))....
GGGGCUAUAGCUCAGCUGGGAG-AGCGCUUGCAUGGCAUGCAAGAG--G---U-C--AGCGGUUCGAUCCCGCUUAGCUCCACCA
GCCGAAGUGGCGAAAUCGGUAGACGCAGUUGAUUCAAAAUCAACCGUAGAAAUACGUGCCGGUUCGAGUCCGGCCUUCGGCACCA
(((((((..(((...........))).(((((.......)))))).(((....))))..((((((.......))))))))))....
*  *   * **  *   ** **  **  ***  *   *  ***  *  *   * *    ********  ***     *  *****
```

D.desulfuricans RNase P RNA/*C.jejuni* RNase P RNA

```
((((((((((((((((((.(((((((((....)))))))))..............(((( ........((((((((((.....)))))))(((((....)))) ))((...(((((............
GGAGUCGGACGGAUCGUCGCCGCGGGGGCAACUCCGGGGAGGAAAGUCCGGGCUCC-AAAGGGCAGAACGCUGGAUAACAUCCAGGGAGGGCAACCUC-CGGACAGCGCCACAGAAAGCAAA
AAGCAUAGUAA-AUGCUCGCUUCUUU----U-U-AGGAGAGGAAAGUCCGAGCUGCUAAAGACAAACAUUCCAUCUAACAGAUGGCUAGGGUAACCUAAGGGAUAGUGCAACAGAAAGAAAA
(((((((((.( ((((((.(((((..    . . )))))).............((((.........((((((((((.....)))))))(((((....)))))).)((...((.............
        *    **  ****  *         *  ** ************ *** * ****   *  *  *    *****    *  **** *****   *** ** ** ******** ***

(((((((((....)))))))..)).......((((((.......))))))((((((((((....)))))))).)).)))..)))))))))))))...((((.....((((((((((....
CCGCCCGGCCUCGGCCGGGUAAGGGUGAAACGGUGGUGUAAGAGACCACCAGAUGCCGUGGUGACACGGCAUGCUCGGCAUACCCCGUUCGGAGCAAGACCAAAUAGGGAAGGCGGCCGGCC
CUACCACGC--AA--GUGGAAAAGGUGAAACGGCGGGGUAAAAGCCCACCAGCGAUUUUGGUAACAAUUUCGGCUAUGUAAACCCAAUGUGCAGCAAGAAGGGAU-GGUUAG-CGUCU---U
((.((((..  ..  ))))...)).......((.(((.......))).)).(((..(((....))))..))).........))))))))))))...((((((.. ..(((( .....   .
*  **  **        ** ** ********** ** **** ** *******      **** ***      ***  * * ****  *  * *******    ** **  ** ** * *

.))))))...)))))).....))))......(((((((((....)))))))).........)))))))(((((((((....)))))))).............).)))))))))....
CGGCCGAAGCCUUCCGGGUAGGUUGCUUGAGGGUGUGGGCAACCGCACUCCUAGAGGAAUGACGGUCACACGCGGGCAACCGUGUGGACAGAACCCGGCUUACAGUCCGACUCCCGCA
UG-----UU-UUAA-CC---CUUCGCUUGAUUUUGUUUGCAAAAACAAAACUAGAUAAAUGAGCAUU-CA--------A--G-----ACAGAACUCGGCUUAUC-GCUAUGCUU-UUU
..     .. )))) ))   ))))......(((((((((....)))))))).........)))))). ..         .  .     ..............). )))))))) ...
 *         *          * ******   ***  ****   **   *****  *****   *  **        *  *     ******* *******    *
```

Fig. 5. Gardenia alignments: two tRNA genes from *E. coli* (above), and *D. desulfuricans* and *C. jejuni* RNase P RNAs (below). * indicate conserved positions. Alignments obtained with RNAforester are identical.

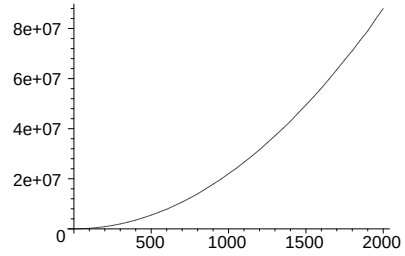


Fig. 6. Experimental average complexity on pairs of random trees for the ALIGN(NEST, NEST, III) $\rightarrow$ NEST algorithm. The horizontal axis is the size of the trees and the vertical axis is the number of operations.

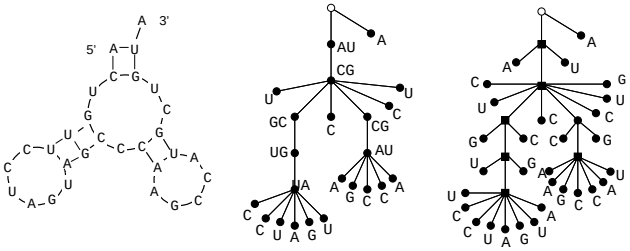


Fig. 7. A secondary structure (left), its associated tree according to the classical representation (middle), and according to [14] (right).

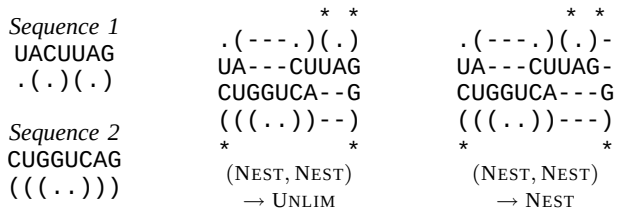


Fig. 8. Given the two input sequences *sequence 1* and *sequence 2*, the edit distance algorithm, corresponding to the (NEST, NEST $\rightarrow$ UNLIM) scheme, leads to a questionnable alignment (center). Arcs are represented by a pair of brackets, and single bases by dots here. The base pair `U-G` marked with $*$ in *Sequence 1* is modified by an arc-altering operation, followed by an arc-breaking operation creating the base pair `C-G` in *sequence 2*. As a result, 3' bases are matched in the superstructuree, whereas the incident arcs are unrelated. The alignment induced by the (NEST, NEST) $\rightarrow$ NEST scheme is more convincing (right).

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 $N(n,k) = \frac{1}{n}\binom{n}{k}\binom{n}{k-1}$. The average number of leaves in a random tree is $\sum_k N(n,k)/C_n = \frac{n+1}{2}$. It remains $\frac{n+1}{2}$ inner nodes in average for n edges, hence the result.

E. Application to RNA comparison

From a historical perspective, RNA secondary structures, corresponding to NESTED sequences, were first encoded by labeled ordered rooted 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 evoluationary 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 EDIT(NESTED, NESTED) problem is 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 EDIT(CROSSING, NESTED) problem (corresponding to ALIGN(CROSSING, NESTED, III) $\rightarrow$ 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

```
(((....)))     (((....)))
GACCAAUGUC     GACCAAUGUC
GA-CAAU-UC     GAC-AAU-UC
((-....-))     ((.-...-))
```

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.

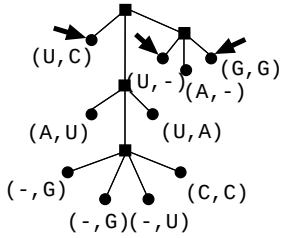


Fig. 10. P-node supertree for sequences 1 and 2 of Figure 8. The supertree induces the same optimal alignment as the ALIGN(NEST, NEST) → UNLIM scheme in figure 8. Positions marked with * in the alignment are pointed with arrows in the supertree.

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 of Figure 8.

We conclude this section with some comment on the time requirement on biological data. The algorithm has been implemented in C language in a prototype software, called *gardenia*. We ran gardenia on four pairs of RNAs – tRNA genes from *E. coli*, prokaryotic RNase P RNAs, IRES elements and 16S rRNAs – and compare it with results obtained with RNAforester. We have used the default score scheme of RNAforester for both programs. In all four examples, we get the same score and the same alignment with both programs (up to minor local changes due to the existence of co-optimal solutions). Figure 5 displays alignments for tRNAs and RNase P RNAs. We also report in Figure 11 the execution times. It appears that gardenia outperforms RNAforester. Both algorithms have the same algorithmic asymptotic complexity. The difference comes from the size of the input. The P-node encoding implemented in RNAforester increases the size of the tree by adding supplementary nodes. Each base-pair is encoded by three nodes, instead of a single node, which penalizes the efficiency of the algorithm.

	gardenia	RNAforester
tRNAs (80nt) – [14] *E. coli*	0	0
RNase P RNAs (350 nt) – [7] *D. desulfuricans* and *C. jejuni*	0.23	6
IRES RFAM 00549 (600nt) – [12] *H. sapiens* and *M. musculus*	0.4	5
16S rRNAs (1500 nt) – [20] *B. subtilis* and *A. pernix*	2.3	25

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

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

[1] Julien Allali and Marie-France Sagot. Novel tree edit operations for RNA secondary structure comparison. In *WABI*, pages 412–425. Lecture Notes in Computer Science, 2004.

[2] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, volume 937, pages 1–16. Lecture Notes in Computer Science, 1996.

[3] F. Bernhart and B. Kainen. The book thickness of a graph. *J. Comb. Theory Series B*, 27:320–331, 1979.

[4] T.C. Biedl, G. Kant, and M. Kaufmann. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19(4):427–446, 1997.

[5] G. Blin, G. Fertin, I. Rusu, and C. Sinoquet. RNA sequences and the EDIT(NESTED, NESTED) problem. *technical report - LINA*, 2003.

[6] G. Blin and H. Touzet. How to compare arc-annotated sequences: The alignment hierarchy. In *13th International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 4209 of *Lecture Notes in Computer Science*, pages 291–303. Springer Verlag, 2006.
[7] J.W. Brown. The Ribonuclease P database. *Nucleic Acids Research*, 27(314), 1999. http://www.mbio.ncsu.edu/RNaseP/.
[8] M. Crochemore, D. Hermelin, G.M. Landau, and S. Vialette. Approximating the 2-interval pattern problem. In *ESA'05*, pages 426–437, 2005.
[9] Erik Demaine, Shay Mozesand Benjamin Rossman, and Oren Weimann. An optimal decomposition algorithm for tree edit distance.
[10] S. Dulucq and H. Touzet. Decomposition algorithms for the tree edit distance problem. *Journal of Discrete Algorithms*, 3(2-4):448–471, 2005.
[11] P. Evans. *Algorithms and Complexity for Annotated Sequences Analysis*. PhD thesis, University of Victoria, 1999.
[12] Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean R. Eddy, and Alex Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33(Database issue):D121–D124, 2005.
[13] Valentin Guignon, Cedric Chauve, and Sylvie Hamel. An edit distance between RNA stem-loops. In *String Processing and Information Retrieval (SPIRE 2005)*, volume 3772, pages 335–347. Lecture Notes in Computer Science, 2005.
[14] Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structures. In *Proc. 2nd IEEE Computer Society Bioinformatics Conference (CSB 2003), 11-14 August 2003, Stanford, CA, USA*, pages 159–168, 2003.
[15] I.L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13), 2003. disponible à http://www.tbi.univie.ac.at/ ivo/RNA/.
[16] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–388, 2002.
[17] T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. *Journal of Dicrete Algorithms*, pages 257–270, 2004.
[18] T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.
[19] Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143:137–148, 1995.
[20] Cannone JJ, Subramanian S, Schnare MN, Collett JR, D'Souza LM, Du Y, Feng B, Lin N, Madabusi LV, Müller KM, Pande N, Shang Z, Yu N, and Gutell RR. The comparative RNA web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3:2, 2002.
[21] P. Klein. Computing the edit-distance between unrooted ordered trees. In *6th European Symposium on Algorithms*, pages 91–102, 1998.
[22] 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*, 65:465–480, 2002.
[23] G. Lin, B. Ma, and K. Zhang. Edit distance between two RNA structures. In *RECOMB*, pages 211–220, 2001.
[24] B. Ma, L. Wang, and K. Zhang. Computing similarity between RNA structures. *Theoretical Computer Sciences*, 276:111–132, 2002.
[25] Yann Ponty, Michel Termier, and Alain Denise. Genrgens: software for generating random genomic sequences and structures. *Bioinformatics*, 22(12):1534–1535, 2006.
[26] and Zhang KZ Shapiro BA. Comparing multiple RNA secondary structures using tree comparisons. *Comput Appl Biosci.*, 6(4):309–18, 1990.
[27] K.C. Tai. The tree-to-tree correction problem. *Journal of the Association for Comput. Machi.*, 26:422–433, 1979.
[28] S. Vialette. On the computational complexity of 2-interval pattern matching. *Theoretical Computer Science*, 312(2-3):223–249, 2004.
[29] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6):1245–1262, 1989.
[30] M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bull. Math. Biol.*, 46:591–621, 1984.