How to compare arc-annotated sequences: The alignment hierarchy

Guillaume Blin, Helene Touzet

How to compare arc-annotated sequences: The alignment hierarchy

Guillaume Blin[1] and Hélène Touzet[2]

[1] IGM-LabInfo - UMR CNRS 8049 - Université de Marne-la-Vallée
77 454 Marne-la-Vallée Cedex 2 - FRANCE
gblin@univ-mlv.fr
[2] LIFL - UMR CNRS 8022 - Université Lille 1
59 655 Villeneuve d'Ascq Cedex - FRANCE
Helene.Touzet@lifl.fr

Abstract. We describe a new unifying framework to express comparison of arc-annotated sequences, which we call *alignment of arc-annotated sequences*. We first prove that this framework encompasses main existing models, which allows us to deduce complexity results for several cases from the literature. We also show that this framework gives rise to new relevant problems that have not been studied yet. We provide a thorough analysis of these novel cases by proposing two polynomial time algorithms and an **NP**-completeness proof. This leads to an almost exhaustive study of alignment of arc-annotated sequences.

Keywords: computational biology, RNA structures, arc-annotated sequences, NP-hardness, edit distance, algorithm

1 Introduction

In computational biology, comparison of RNA molecules has attracted a lot of interest recently. From a combinatorial perspective, one can distinguish two types of modeling that allow for various flexibility and preciseness in the encoding of RNA structures: macroscopic representations, with two-interval graphs [16, 4], and microscopic representations with arc-annotated sequences, originally introduced in [6]. We focus here on arc-annotated sequences, which are raw sequences provided with related additional information in the form of arcs connecting pairs of positions. The set of arcs determines the way the sequence folds into a three-dimensional space.

Arc-annotated sequences may be refined into four main paradigms: tree edit distance [15, 17, 11, 5], tree alignment [10], longest common arc-preserving subsequence [6, 9, 12], and general edit distance [8, 3]. We propose a unifying framework to express comparison of arc-annotated sequences that is based on the introduction 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 called the Align *hierarchy* in reference to the tree alignment. We show that this hierarchy brings together all previously mentioned comparison models for arc-annotated sequences,

and leads to the introduction of new comparison models that are biologically relevant. In particular, we propose two polynomial time algorithms for the problem of comparing two Nested arc-annotated sequences, whereas corresponding algorithms considering the same set of edit operations in other formalisms are not polynomial (Sections 4.2 and 5). We also give an **NP**-completeness result that gives some new insight on the hardness of the comparison of Crossing arc-annotated sequences (Section 4.3). This leads to an almost exhaustive study of the Align hierarchy. Due to space considerations, complete proofs are deferred to the full version of the paper.

2 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. 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 [6]):

- 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.

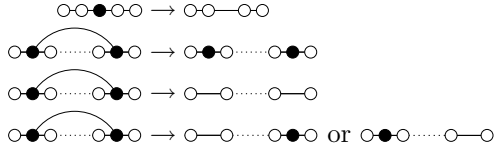
There is an obvious inclusion relation between those arc types with the $\subset$ operator (Plain $\subset$ Nested $\subset$ Crossing $\subset$ Unlimited). Since we focus here on structure comparison, we do not consider Plain sequences, which do not carry any structural information. In the remaining 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 [13] and classify it into two groups:

Substitution operations, inducing renaming of bases in the arc-annotated sequence: *base-match* ($w_m : \Sigma^2 \to \mathbb{R}$), *base-mismatch* ($w_m : \Sigma^2 \to \mathbb{R}$), *arc-match* ($w_{am} : \Sigma^4 \to \mathbb{R}$), *arc-mismatch* ($w_{am} : \Sigma^4 \to \mathbb{R}$).

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

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



Given the above set of operations, we 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.

In the following, given two arc-annotated sequences u and v, a *K-edit script* from u to v will refer 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 an *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 strings 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.*

We now turn to a novel paradigm, 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 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 $\texttt{cost}(w, u, K) + \texttt{cost}(w, v, K)$. First, we prove that each $\textsc{Edit}$ problem can 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.*

A point worth to notice with Lemma 2 is 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 arcs or multiple arcs incident to a single character that are absent in 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 supersequences to get the same cost as for the EDIT problem. This observation gives rise to a family of new problems, which we call the ALIGN *hierarchy*.

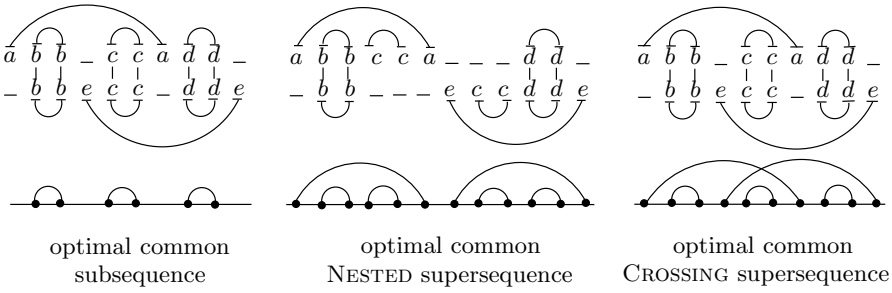


Fig. 1. Comparison of the optimal common subsequence and the optimal common supersequences. The optimal common subsequence is derived from u and v with two arc-removings. The optimal common NESTED supersequence requires four arc-removings. In this example, it is necessary to allow crossing arcs in the supersequence to get the same cost as for the subsequence (third scheme).

Definition 3 (Arc-annotated sequence alignment). *Given three types of sequences A, B and C of* $\{\text{NESTED}, \text{CROSSING}, \text{UNLIMITED}\}$ *and an edit model* $K \in \{\text{I}, \text{II}, \text{III}\}$*, the* $\text{ALIGN}(A, B, K) \to C$ *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 ALIGN hierarchy and confront it to known results for existing comparison models for arc-annotated sequences. Since $\text{ALIGN}(A, B, K) \to C$ is equivalent to $\text{ALIGN}(B, A, K) \to C$, 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 when considering all relevant possibilities for A, B, C and K.

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

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

3 Ordered trees and the 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. Moreover, in this model, considering arc-annotated I-supersequences 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* ALIGN$(A, B, \text{I}) \rightarrow$ UNLIM *and* ALIGN$(A, B, \text{I}) \rightarrow$ CROS *problems are equivalent.*

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

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 ALIGN(NEST, NEST, I) $\rightarrow$ NEST, which fully corresponds to the ordered tree alignment, introduced by Jiang *et al.* in [10]. Therefore, the ALIGN hierarchy is completely solved for the edit model I, as summed up in Table 1.

$A \times B \rightarrow C$	EDIT	model I
NEST × NEST → NEST		$O(n^4)$ – Jiang [10]
NEST × NEST → CROS NEST × NEST → UNLIM	×	$O(n^3 \log(n))$ – Klein [11]
CROS × NEST → CROS CROS × NEST → UNLIM	×	$O(n^3 \log(n))$ – Ma [14]
CROS × CROS → CROS CROS × CROS → UNLIM	×	**NP**-complete – Ma [14]
UNLIM × NEST → UNLIM	×	$O(n^3 \log(n))$ – Lemma 4
UNLIM × CROS → UNLIM	×	**NP**-complete – Ma [14]
UNLIM × UNLIM → UNLIM	×	**NP**-complete – Ma [14]

Table 1. ALIGN 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. $\max(|u|, |v|) = n$.

4 The edit model II

4.1 Some correspondences with the LAPCS problem

As introduced by Evans in [6], 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 $w \trianglelefteq_{\text{II}} u$.*

This theorem combined with Theorem 1 allows us to derive several cases of the ALIGN hierarchy for the edit model II from recent results published in the LAPCS literature. All known results are summed up in Table 2. It remains four specific problems: ALIGN(NEST, NEST, II) $\rightarrow$ {NEST, CROS} and ALIGN(CROS, {NEST, CROS}, II) $\rightarrow$ CROS. The first two problems can be seen as a refinement of the EDIT(NESTED, NESTED, II) problem, which is not tractable. 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) $\rightarrow$ CROS and ALIGN(CROS, CROS, II) $\rightarrow$ CROS are also **NP**-complete.

$A \times B \rightarrow C$	EDIT	model II	model III
NEST × NEST → NEST		$O(n^4)$	$O(n^4)$
NEST × NEST → CROS		**NP**-complete	
NEST × NEST → UNLIM	×	**NP**-complete – Lin [12]	**NP**-complete – Blin [3]
CROS × NEST → CROS		**NP**-complete	
CROS × NEST → UNLIM	×	**NP**-complete – Evans [6]	Max SNP-hard – Jiang [8]
UNLIM × NEST → UNLIM	×		
CROS × CROS → CROS		**NP**-complete	
CROS × CROS → UNLIM	×	**NP**-complete – Evans [6]	Max SNP-hard – Jiang [8]
CROS × UNLIM → UNLIM	×		
UNLIM × UNLIM → UNLIM	×		

Table 2. ALIGN hierarchy for edit models II and III. We indicate problems that can be formulated as edit distance problem in the second column. In these cases, known results stem from the LAPCS problem for the model II (Theorems 1 and 2), and from the general edit distance for the model III (Theorem 1). Other problems are specific to the ALIGN hierarchy and are introduced and studied 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. $\max(|u|, |v|) = n$.

4.2 ALIGN(NESTED, NESTED, II) → NESTED problem is polynomial

We exhibit a polynomial algorithm for the ALIGN(NEST, NEST, II) → NEST problem. This result is somehow unexpected since the associate edit problem EDIT(NESTED, NESTED, II) is **NP**-complete. It shows that imposing structural

constraints on the type of the common supersequence is an adequate way for lower complexity of untractable problems.

We saw in Section 3 that in the model I the ALIGN(NEST, NEST, I) → NEST problem is polynomial, since it is equivalent to ordered tree alignment. The algorithm proposed in [10] proceeds by dynamic programming. 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.

We show here that the formulas for the edit model I can be extended to the edit model II by adding supplementary rules for the arc-altering operation. All rules concerning substitutions, base-deletions and arc-removings are identical.

We introduce some notations for the representation of arc-annotated sequences. 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. $b \circ u$ denotes the arc-annotated sequence composed by the single base b concatenated to the arc-annotated sequence u. The common supersequence is built from right to left. 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 should not be considered for all forms of pairs of arc-annotated sequences: At least one of the two sequences should begin with a base incident to an arc. We write A for the cost of the alignment between two arc annotated-sequences.

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

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

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

and symetrically

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

$$5.A(b \circ v, b_2 \circ x) \;=$$
$$\min \begin{cases} w_d(b) + A(v, b_2 \circ x) & \text{– base-deletion} \\ w_d(b_2) + A(b \circ v, x) & \text{– base-deletion} \\ w_m(b, b_2) + A(v, x) & \text{– base-(mis)match} \end{cases}$$

The hypothesis that the common supersequence is of NESTED type guarantees the correctness of the recurrence relations. The whole complexity remains unchanged: it is $O(n^4)$. A full analysis of this algorithm and its application to RNA structure comparison (global alignment, local alignment etc.) is presented in further detail in [7].

Theorem 3. ALIGN(NEST, NEST, II) → NEST *is polynomial.*

4.3 Hardness result for ALIGN(NESTED, NESTED, II) → CROSSING

We show in this section that relaxing the constraint on crossing arcs in the common supersequence makes the problem difficult.

Theorem 4. ALIGN(NEST, NEST, II) → CROS *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 [2].

MIS-3P
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 2-page book embedding.

Theorem 5 (Bernhart and al. [1]). *One can always find, in polynomial time, a 2-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 *2-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 2-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 $baaa$ 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 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*.

For the sake of simplicity, but w.l.o.g.[1], 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 proof is 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 $U_i = baaab$.*

[1] Since a subcase of ALIGN(NEST, NEST, II) → CROS is hard, so does the general problem.

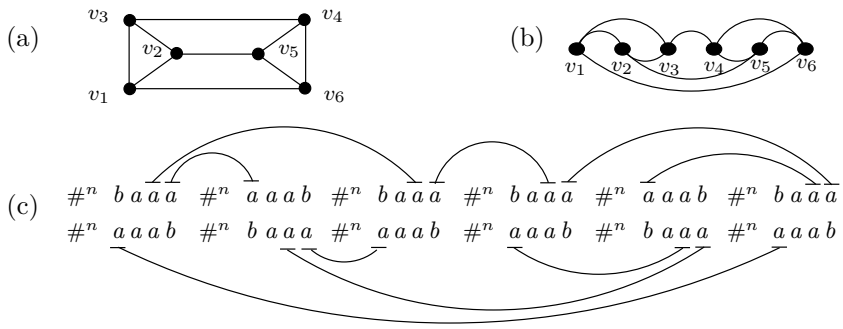


Fig. 2. Example of an align-construction. The graph (a) is a cubic planar bridgeless connected graph of 6 vertices. The graph (b) is a 2-page book embedding of the graph (a) such that, on each page, any vertex has a non-null degree. (c) The two arc-annotated sequences of NESTED type obtained from the graph (a) by an align-construction.

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.*

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.*

We now turn to prove that ALIGN(NEST,NEST,II) $\rightarrow$ CROS is **NP**-complete with this following Lemma. This 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 $n = |V|$.*

Remark 1. The arc-annoted sequences of the **NP**-completeness proof are not conform to the representation of an RNA molecule. It is likely to impose supplementary constraints on the encoding of the 2-page book embedding in order to get sequences that are more RNA-like: the alphabet is $\{A, U, C, G\}$, all arcs correspond to Watson-Crick pairings ($A \leftrightarrow U$ and $C \leftrightarrow G$) and base-deletion costs are more realistic. 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 2-page book embedding now corresponds to two arcs between AU and AU. Figure 3 shows this new representation for the example of Figure 2.

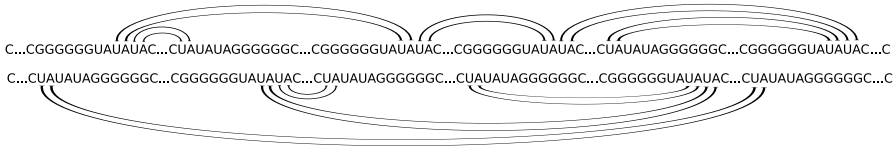


Fig. 3. RNA-like arc-annotated sequences for the example of Figure 2.

5 The general edit distance and the edit model III

The edit model III corresponds to the set of operations introduced by Jiang *et al.* in the *general edit distance* problem [8]. Therefore it allows us to derive several complexity results from known results on the *general edit distance* [8, 3] with Theorem 1. As illustrated in Table 2, the complexity of Align(Nest, Nest, III) → {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(Nest, Nest, III) → Nest *is polynomial.*

To prove the correctness of the above Theorem, we show that we can enrich the polynomial time algorithm defined in Section 4.2 by incorporating rules for arc-breaking operations. At each step of the construction of the common supersequence, it is necessary that one of the sequence begins with an arc, and the other one with a single base for the arc-breaking operation to be valid. So only cases 3 and 4 in the recurrence relations are concerned by the application of an arc-breaking rule.

3. $A(b \circ v, \beta(w) \circ x) \ =$
$$\min \begin{cases} \cdots \\ w_b(\beta, b, b_2) + \min\{A(y, w) + A(z, x) | x \circ b_2 \circ z = v\} \end{cases}$$

4. $A(\alpha(u) \circ v, b \circ x) \ =$
$$\min \begin{cases} \cdots \\ w_b(\alpha, b, b_2) + \min\{A(u, y) + A(v, z) | y \circ b_2 \circ z = x\} \end{cases}$$

6 Conclusion

In this article, we have proposed and studied a new framework for comparing arc-annotated sequences, namely the Align hierarchy. We think that this study is relevant both from a practical perspective and theoretical perspective. We have provided two polynomial time algorithms 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 also 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.

Those results, combined with the ones derived from EDIT and LAPCS comparison models, have almost filled the complexity table of the ALIGN hierarchy. As illustrated in Table 2, 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 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.

References

1. F. Bernhart and B. Kainen. The book thickness of a graph. *J. Comb. Theory Series B*, 27:320–331, 1979.
2. T.C. Biedl, G. Kant, and M. Kaufmann. On triangulating planar graphs under the four-connectivity constraint. *Algorithmica*, 19(4):427–446, 1997.
3. G. Blin, G. Fertin, I. Rusu, and C. Sinoquet. RNA sequences and the EDIT(NESTED, NESTED) problem. *technical report - LINA*, 2003.
4. M. Crochemore, D. Hermelin, G.M. Landau, and S. Vialette. Approximating the 2-interval pattern problem. In *ESA'05*, pages 426–437, 2005.
5. S. Dulucq and H. Touzet. Decomposition algorithms for the tree edit distance problem. *Journal of Discrete Algorithms*, 3(2-4):448–471, 2005.
6. P. Evans. *Algorithms and Complexity for Annotated Sequences Analysis*. PhD thesis, University of Victoria, 1999.
7. C. Herrbach, A. Denise, S. Dulucq, and H. Touzet. A polynomial algorithm for comparing RNA secondary structures using a full set of operations.
8. 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.
9. 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.
10. T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.
11. P. Klein. Computing the edit-distance between unrooted ordered trees. In *6th European Symposium on Algorithms*, pages 91–102, 1998.
12. 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.
13. G. Lin, B. Ma, and K. Zhang. Edit distance between two rna structures. In *RECOMB*, pages 211–220, 2001.
14. B. Ma, L. Wang, and K. Zhang. Computing similarity between RNA structures. *Theoretical Computer Sciences*, 276:111–132, 2002.
15. K.C. Tai. The tree-to-tree correction problem. *Journal of the Association for Comput. Machi.*, 26:422–433, 1979.
16. S. Vialette. On the computational complexity of 2-interval pattern matching. *Theoretical Computer Science*, 312(2-3):223–249, 2004.
17. 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.

Annexes

Proof of Lemma 1

Lemma 1. *Given two arc-annotated sequences u and v, and an edit model $K \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$, finding the K-edit distance between u and v is equivalent to to finding a common K-subsequence w of u and v of minimum 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. Once it has been done, the form of the deletion rules ensures that there are no conflicting critical pairs. It follows that the script may be modified so as 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. □

Proof of Lemma 2

Lemma 2.*Given two arc-annotated sequences u and v, and an edit model $K \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$, 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:

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

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, notice that $\texttt{cost}(x, u, K) = \texttt{cost}(v, w, K)$. 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

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

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

Proof of Lemma 3

Lemma 3. *Given two types A, B in* {Nested, Crossing}*, the* Align(A, B, I) → Unlim *and* Align(A, B, I) → 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. □

Proof of Lemma 4

Lemma 4. *Given a type B in* {Nested, Crossing}*, the* Align(Unlim, B, I) → Unlim *problem has the same complexity as* Align(Cros, B, I) → 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. □

Proof of Theorem 2

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 – noted M – from u' to v' induced by $u' \trianglelefteq_{\text{II}} v'$.
($\Rightarrow$) The proof is by recurrence 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 recurrence 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 to $P = Q$. 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 $u \trianglelefteq_{\mathrm{II}} w$ 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 recurrence hypothesis will allow us to conclude that $w \trianglelefteq_{\mathrm{II}} v$, which implies $u \trianglelefteq_{\mathrm{II}} v$ by transitivity of $\trianglelefteq_{\mathrm{II}}$.

We have to consider several cases according to the quality of $T[i]$. 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 $S'S" = S$).

– $T[i]$ is a single base: w is defined by $U = T[1..i-1] \circ T[i+1..|T|]$ and $R = Q$. We have $w \trianglelefteq_{\mathrm{II}} v$ 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 recurrence hypothesis implies $u \trianglelefteq_{\mathrm{II}} w$.

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 $w \trianglelefteq_{\mathrm{II}} v$ 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 $u \trianglelefteq_{\mathrm{II}} w$.

– 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 $u \trianglelefteq_{\mathrm{II}} w$. □

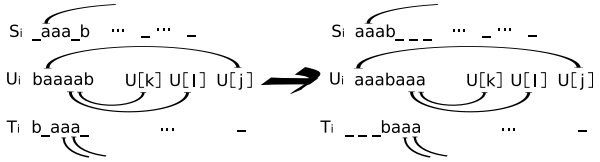
Proof of Lemma 5

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 $U_i = baaab$.*

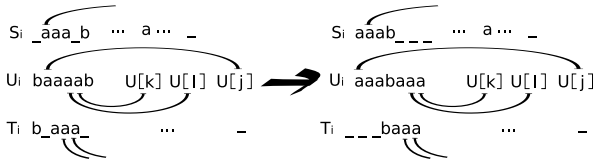
Proof. It is easy to verify that $(\#^n aaabaaa)^n$ is a common II-supersequence whose cost is lower than or equal to $n(\frac{3}{2}w_r(a, a)+3w_d(a)) = 6n$. This observation ensures that any optimal supersequence is of the form $U = \#^n U_1 \ldots \#^n U_n$, where $U_i \in \{a, b\}^*$. Indeed, assume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

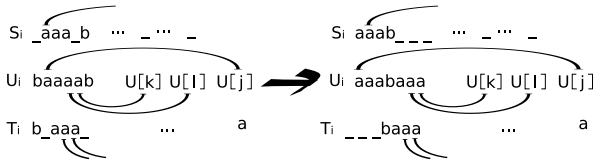
Assume there exists $i \in 1..n$ such that $U_i = baaaab$. We suppose w.l.o.g. that $S_i = aaab$ and $T_i = baaa$. The construction of u and v ensures that there is no j such that there exists an arc connecting both S_i and S_j in u, and T_i and T_j in v. Therefore three arcs are incident from U_i. Let 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 4). The main argument that is common to all cases is that replacing U_i with $aaabaaa$ does not increase the cost of the alignment.



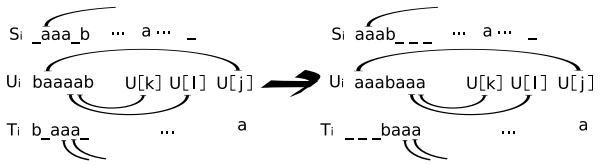
Case 1: $U[j]$ does not appear in T, and $U[k]$, $U[l]$ do not appear in S



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. 4. Four first cases for the replacement of *baaaab* in proof of Lemma 5

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 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 very similar for *baaaaab* and *baaaaaab*. □

Proof of Lemma 6

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. □

Proof of Lemma 7

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 $3pw_a(a,a,a) + 3(\frac{n}{2} - p)w_r(a,a) + 3(n-p)w_d(a) + 2pw_d(b)$. □

Proof of Lemma 8

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 $k \leq |V'|$. □