Tree decomposition and parameterized algorithms for RNA structure-sequence alignment including tertiary interactions and pseudoknots

Philippe Rinaudo, Yann Ponty, Dominique Barth, Alain Denise

Tree decomposition and parameterized algorithms for RNA structure-sequence alignment including tertiary interactions and pseudoknots (extended abstract)

Philippe Rinaudo[1,2], Yann Ponty[3], Dominique Barth[2], and Alain Denise[1,4]

[1] LRI, Univ Paris-Sud, CNRS UMR8623 and INRIA AMIB, Orsay, F91405
[2] PRISM, Univ Versailles Saint-Quentin and CNRS UMR8144 Versailles, F78000
[3] LIX, Ecole Polytechnique, CNRS UMR7161 and INRIA AMIB, Palaiseau, F91128
[4] IGM, Univ Paris-Sud and CNRS UMR8621, Orsay, F91405

Abstract. We present a general setting for structure-sequence comparison in a large class of RNA structures that unifies and generalizes a number of recent works on specific families on structures. Our approach is based on *tree decomposition* of structures and gives rises to a general parameterized algorithm, where the exponential part of the complexity depends on the family of structures. For each of the previously studied families, our algorithm has the same complexity as the specific algorithm that had been given before.

1 Introduction

The RNA structure-sequence comparison problem arises in two main kinds of applications: searching for a given structured RNA in a long sequence or a set of sequences, and three-dimensional modeling by homology. In [6], Jiang and *al.* addressed the problem of pairwise comparison of RNA structures in its full generality. They defined the edit distance problem on RNA structures represented as graphs, using a set of atomic edition operations. Notably, they gave a dynamic programming algorithm in $\mathrm{O}(nm^3)$ time complexity for comparing a sequence to a *nested* structure where n is the length of the sequence with known structure and m the length of the sequence of unknown structure. They also established that computing the edit distance between a sequence and a structure is a Max-SNP-hard problem if the structure contains pseudoknots.

Meanwhile, consiodering all the known interactions in RNAs including non-canonical ones [8] and pseudoknots is crucial for precise structure-sequence alignment. In [6], a polynomial algorithm was developed for pseudoknotted structures, but it involves constraints on the costs of the edition operations. It has been observed that the so-called *H-type* and *kissing-hairpin* pseudoknots represent more than 80% of the pseudoknots in known structures [9]. If the structure contains only only H-Type pseudoknots, the alignment problem can be solved in $O(nm^3)$ [5, 11]. Moreover, in [5] these two classes of pseudoknots where embedded into a more general class, the *standard pseudoknots*. Two $O(nm^k)$ and $O(nm^{k+1})$ algorithms, respectively for a single standard pseudoknot, and for a standard pseudoknot which is embedded inside a nested structure were developed, where k is the so-called *degree* of the pseudoknot. Recently, the more general class of simple non-standard pseudoknots was defined, and an algorithm was given in $O(nm^{k+1})$ if alone, or in $O(nm^{k+2})$ for a simple recursion [11].

In the present paper, we give a general setting for sequence-structure comparison in a large class of RNA structures that unifies and generalizes all the above families of structures. Notably, we handle structures where every nucleotide can be paired to any number of other nucleotides, thus considering all kinds of non-canonical interactions [8]. Our approach is based on *tree decomposition* of structures and gives rises to a general parameterized algorithm, where the exponential part of the complexity depends on the family of structures. For each of the previously studied families, our algorithm has the same complexity as the specific algorithm that had been given before. Table 1 give a summary of the previous works that are generalized by our approaches, and the time complexity of our algorithm for each of the classes.

Class of Structures	Time comp.	Multiple interactions	Ref.
RCS – Recursive Classical Structures	$O(nm^{k+2})$	$\surd$	–
PKF – Secondary Structures (Pseudoknot-free)	$O(nm^3)$		[6]
ESP – Embedded Standard Pseudoknots	$O(nm^{k+1})$		[5]
SST – Standard Structures	$O(nm^k)$	$\surd$	[5]
SPK – Standard Pseudoknots	$O(nm^k)$		–
2RP – 2-Level Recursive Simple Non-Standard Pseudoknots	$O(nm^{k+2})$		–
SNS – Simple Non-Standard Structures	$O(nm^{k+1})$	$\surd$	[11]
SNP – Simple Non-Standard Pseudoknots	$O(nm^{k+1})$		–
ETH – Extended Triple Helices	$O(nm^3)$	$\surd$	[11]
STH – Triple Helices	$O(nm^3)$	$\surd$	[10]

Table 1. Summary of existing algorithms for structure-sequence alignment. Our approach unifies all these algorithms and, for each class of structures captured by pre-existing works, specializes into time complexities that matches previous efforts: the tree structure represents an inclusion relation. Hence the root class RCS includes all other classes. Notation: k is the degree of the pseudoknot/(simple) standard structure.

2 Sequence-Structure Alignment

At first let us state some definitions, starting with the concept of arc-annotated sequence, which will be used as an abstract representation for RNA structure.

Definition 1 (Arc-annotated sequence). *An arc-annotated sequence is a pair (S, P), where S is a sequence over an alphabet Σ and P is a set of unordered pairs of positions in S.*

For RNA structures, obviously S is the nucleotide sequence and P the set of the interactions over S, as illustrated by Figure 2 (Upper part). Here $\Sigma = \{A, U, G, C\}$, and any $(i, j) \in P$ represents an interaction between the nucleotides at position i and j. The nucleotides are numbered from 1 to n (where n is the sequence length), follow a 5' to 3' order, and $S[i]$ denotes the nucleotide number i. One should notice that, unlike most definitions, a position i can be involved in multiple interactions, allowing for the representation of tertiary structures. In the following, we sometimes refer to such an arc-annotated sequence as an **RNA graph interaction structure** or, in short, an **RNA graph**.

There exists several equivalent ways to define a structure-sequence alignment, that is an alignment between an arc-annotated sequence and a (plain) sequence. We choose to abstract an alignment as a mapping, i.e. we represent an alignment as a partial mapping between positions in the arc-annotated sequence and positions in the plain sequence, as shown in Figure 2.

Definition 2 (Structure-Sequence Alignment). *A structure-sequence alignment between an arc-annotated sequence $A = (S_A, P_A)$ of size n and a plain sequence $B = (S_B, \varnothing)$ of size m is a partial mapping μ from $[1, n]$ to $[1, m + 1]$ such that:*
- *μ is injective.*
- *μ preserves the order: $\mu(i) < \mu(j) \Rightarrow i < j$.*

We write $\mathcal{F}(A, B)$ (or $\mathcal{F}$ when there is no ambiguity) for the set of all possible alignments between A and B.

Remark that some positions in the arc-annotated sequence may be without a corresponding position in the plain sequence. We note $\mu(i) = \perp$ if position i does not have an image by μ, and qualify it as **unmatched**. Consecutive sequences of unmatched positions in the structure ($\mu(i) = \perp$) or in the sequence ($\mu^{-1}(i) = \perp$) are usually grouped and scored together. A **(composite) gap** is then the maximum set of consecutive positions $(i, \ldots, j)$ that are either unmatched or do no have an antecedent by μ. The **length** $|g|$ of a gap g is simply the number of positions it contains. By grouping unmatched positions within gaps, one can handle affine penalty functions in the cost function.

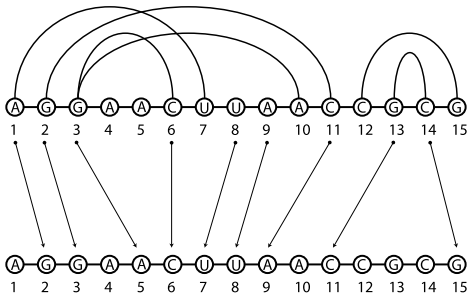

Fig. 1. A representation of a partial mapping between an arc-annotated sequence (upper part) and a (plain) sequence (lower part).

Let us now define the cost of an alignment, which captures the level of similarity between two RNAs, and needs account both for structure and sequence elements.

Definition 3 (Cost of a Sequence-Structure Alignment). *The cost of a structure-sequence alignment μ between an arc-annotated sequence $A = (S_A, P_A)$ of size n and a plain sequence $B = (S_B, \varnothing)$ is defined by:*

$$\mathsf{Cost}(\mu) = \sum_{i\in[1,n],\mu(i)\neq\perp} \gamma(i,\mu(i)) + \sum_{gap\ g\subset A} \lambda_A(|g|) + \sum_{gap\ g\subset B} \lambda_B(|g|) + \sum_{(i,j)\in P_A} \varphi(i,j,\mu(i),\mu(j)) \quad (1)$$

where

- $\gamma(i,\mu(i))$ *is the cost of a* ***base substitution*** *between position i in A and position $\mu(i)$ in B.*
- $\lambda_Y(x) = \alpha_Y \cdot x + \beta_Y$, *is the* ***affine cost penalty*** *for a gap of length x in a sequence Y.*
- $\varphi(i,j,\mu(i),\mu(j))$ *is the cost of an* ***arc removing*** *($\mu(i) = \mu(j) = \perp$),* ***arc altering*** *($\mu(i)$ or $\mu(j) = \perp$) or* ***arc substitution*** *involving paired positions i and j in A.*

Note that unlike the score functions defined in [5, 11], our formulation captures arc-alterations and arc-breakings as atomic operations on their interactions (as in [6]), allowing for general cost schemes.

Definition 4 (Structure-Sequence Alignment Problem). *Given an arc-annotated sequences A and a plain sequence $B = (S_B, \varnothing)$, the structure-sequence alignment problem is to find an alignment between A and B having the minimum cost.*

As stated in [12, 2] the problem is already NP-Hard if we consider Crossing interactions (*i.e.* pseudoknots, without multiple pairings). In the following, we will use a parameterized approach to handle general structures including unrestricted crossing interactions and multiple interactions per position, assuming that the total number of interactions per position is bounded by a constant.

3 Tree Decomposition and Alignment Algorithm

As previously sketched, our alignment algorithm relies on tree decomposition of an arc-annotated sequence. Tree decompositions are usually defined on graphs rather than on arc-annotated sequences. Here we give a straightforward adaptation that preserves all the properties of the standard tree decompositions.

3.1 Definitions

Definition 5 (Tree Decomposition of an arc-annotated sequence). *Given an arc-annotated sequence $A = (S, P)$, a tree decomposition of A is a pair (X, T) where $X = \{X_1, \ldots, X_N\}$ is a family of subsets of positions $\{i, i \in [1, n]\}$, $n = length(S)$, and T is a tree whose nodes are the subsets X_l (called* bags*), satisfying the following properties:*

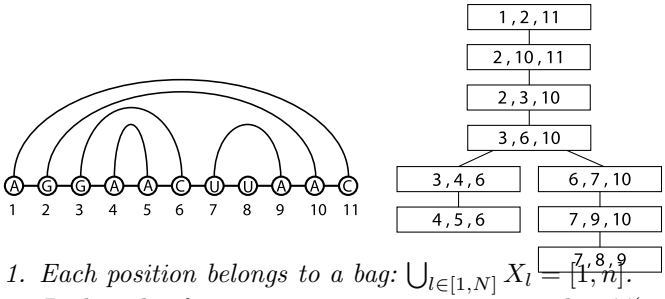

Fig. 2. An arc-annotated sequence and an associated tree decomposition. Each bag contains three positions, so the tree width of this tree decomposition is 2. As there is only one bag which does not respect the smooth property (the one with two children), the tree decomposition is 1-weakly-smooth.

1. *Each position belongs to a bag:* $\bigcup_{l\in[1,N]} X_l = [1, n]$.
2. *Both ends of an interaction are present in a bag:* $\forall (i,j) \in P,\ \exists l \in [1,N], \{i,j\} \subset X_l$.
3. *Consecutive positions are both present in a bag:* $\forall i \in [1, n-1],\ \exists l \in [1,N], \{i, i+1\} \subset X_l$.
4. *For every* X_l *and* X_s, $l, s \in [1,N]$, $X_l \cap X_s \subset X_r$ *for all* X_r *on the path between* X_l *and* X_s.

Figure 2 illustrates the tree decomposition of an arc-annotated sequence.

Definition 6 (Treewidth). *The width of a tree decomposition* (X,T) *is the size of its largest set* X_l *minus one. The* ***treewidth*** $t_w(A)$ *of an arc-annotated sequence* A *is the minimum width among all possible tree decompositions of* A.

In general, tree decomposition are not rooted. Nevertheless, for the sake of clarity, we will arbitrarily choose a root X_0 in our decompositions. Additionally, we use the following notation: for any two bags X_l and X_r in X, we write $X_{l,r}$ as shorthand for $X_l \cap X_r$.

The process of assigning consecutive positions can be simplified by the affine gap penalties, using the general idea of Gotoh's algorithm [4]. Let us then define the notion of smooth bag, that will help us take advantage of this optimization.

Definition 7 (Smooth Bag of a Tree Decomposition). *Let* $X_l \in X$ *be a bag in a tree decomposition* (X,T) *for an arc-annotated sequence* $A = (S,P)$. *If* $X_l \neq X_0$, *then let* X_r *be its father.* X_l *is then* ***smooth*** *iff there exist two consecutive positions* i *and* j *such that* $i \in X_l - X_r$, $j \in X_{l,r}$, j *is not in one of the child of* X_l, *and there is no* $i' \in X_l - X_r$ *such that* $(i', j) \in P$ *or* i' *(except* i*) consecutive to* j. *The root* X_0 *is* ***smooth*** *iff (1) there exist two consecutive positions* $i, j \in X_0$ *such that* j *is not in any child of* X_0 *and* $(i,j) \notin P$, *or (2) iff the size of the root is strictly smaller than the size of one of its children.*

Definition 8 ((Weakly-)Smooth Tree Decomposition). *A tree decomposition* (X,T) *for an arc-annotated sequence* $A = (S,P)$ *is* ***smooth*** *iff every bag* $X_l \in X$ *are smooth.*
A tree decomposition (X,T) *is* k***-Weakly-Smooth*** *iff at most* k *of its bags are not smooth.*

As stated in [3], a tree decomposition can always be converted into a binary tree in linear time. Moreover, this transformation can be done without breaking the smoothness of the tree decomposition. Therefore, we will limit, without loss of generality, the scope of our algorithm and analysis to binary tree decompositions. At last, if the tree decomposition is smooth and root X_0 is smooth by condition (1), then the tree can made smooth by condition (2) by creating a new root composed of the positions of the old one except the position i (with i as in the definition above) and add it as the father of X_0. In the following we will only consider case (2) for the alignment algorithms.

3.2 An Alignment Algorithm based on Tree Decomposition

Let us describe an algorithm that computes the minimum cost alignment of an arc-annotated sequence A and a (flat) sequence B ($P_B = \varnothing$). This algorithm implements a dynamic programming strategy, based on a k-Weakly-Smooth tree decomposition of A.

An alternative internal representation for alignments. The recursive step in our scheme consists in extending a partial alignment, assigning positions that are **proper** to the bag (i.e. not

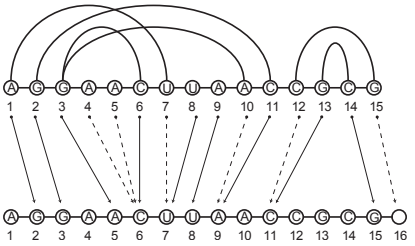

Fig. 3. Internal representation of alignments as a pair $f = (\mu, \delta)$, used within our dynamic programming algorithm. A function μ defines a complete mapping, while δ discriminates matched positions (solid stroke) from unmatched ones (dashed stroke). Additionally, unmatched positions are forced to aggregate to their nearest matched neighbor to their right, and we create an additional virtual position (16 here) to provide an image to trailing unmatched positions in the structure.

present in the bag's father). One of the main challenge is to preserve the sequential order. Indeed, only consecutive positions $(i, i+1)$ are guaranteed to be simultaneously present in a bag, allowing for a direct control over the sequential ordering ($\mu(i) < \mu(i+1)$) at the time of an assignment. Intuitively, this property may extend transitively over μ, since $\mu(i) < \mu(i+1)$ and $\mu(i+1) < \mu(i+2)$ implies $\mu(i) < \mu(i+2)$. However, this property no longer holds when a position is unmatched, as $\mu(i+1)$ is then undefined and cannot serve as a reference point for the relative positioning of $\mu(i)$ and $\mu(i+2)$.

To work around this issue, we identify, in the description of our algorithm, an alignment with a pair $f = (\mu, \delta)$, where $\mu : [1, n] \to [1, m+1]$ is a full ordered mapping ($\mu(\cdot) \neq \perp$) and $\delta : [1, n] \to [0, 1]$ distinguishes between matched ($\delta(i) = 1$) and unmatched positions ($\delta(i) = 0$). As illustrated by Figure 3, this new representation aggregates consecutive unmatched positions in A to their nearest rightward position that is matched in the alignment. A *virtual position* $m+1$ is then added to B to serve as an image for the unmatched positions appearing at the end of A.

Dynamic programming recursion. Our dynamic programming algorithm assigns positions in B to the elements of a bag, proceeding to a recursive call that assigns the positions found further down in the tree decomposition. This requires a few additional notions and definitions.

Let (S, P) be an arc-annotated sequence, and S' be a subset of S. The **arc-annotated subsequence induced by** S' is the pair (S', P') such that P' is the subset of arcs in P whose both extremities are in S'. Let X_l be a bag in the tree-decomposition of A. The **descending subsequence** of X_l is the subset of positions appearing in the descendants of X_l in the tree. The **descending arc-annotated subsequence** of X_l is the arc-annotated subsequence induced by its descending subsequence. The notion of alignment between A and B naturally extends to alignments involving a sub-arc-annotated structure of A, and we denote by $\mathcal{F}|_{S'}$ the set of all possible alignments between the arc-annotated subsequence induced by S', and B.

Let X_l be a bag having father X_r ($X_r := X_l$ when $l = 0$), let $f \in \mathcal{F}|_{X_{l,r}}$ be an alignment for the common positions of X_l and X_r to B. Let us denote by C_f^l the cost of the best alignment between the descending arc-annotated subsequence of X_l and B, which matches f on $X_{l,r}$. It can be shown that C_f^l obeys

$$C_f^l = \min_{\substack{f'=(\mu',\delta')\in\mathcal{F}|_{X_l} \\ f'(i)=f(i),\forall i\in X_{l,r}}} \left\{ \mathsf{LCost}(X_l, f') + \sum_{s\in sons(l)} C^s_{f'|_{X_{s,l}}} \right\}. \tag{2}$$

Moreover, the local contribution $\mathsf{LCost}(X_l, f)$ of a bag X_l to an alignment $f = (\mu, \delta)$ is

$$\begin{aligned}
\mathsf{LCost}(X_l, f) = & \sum_{\substack{i \in X_l - X_r \\ \delta(i)=1}} \gamma(i, \mu(i)) + \sum_{\substack{i,j \in X_l \text{ s.t.} \\ i \text{ or } j \in X_l - X_r \\ \text{and } (i,j) \in P_A}} \varphi(i, j, f(i), f(j)) && \text{// Bases and interactions} \\
& + \sum_{\substack{i, i+1 \in X_l \text{ s.t.} \\ i \text{ or } i+1 \in X_l - X_r \\ \text{and } \mu(i+1) > \mu(i)}} \alpha_B \cdot (\mu(i+1) - \mu(i)) + \beta_B, && \text{// Gaps in sequence } B \\
& + \sum_{\substack{i, i+1 \in X_l \text{ s.t.} \\ i \text{ or } i+1 \in X_l - X_r \\ \delta(i)=1 \text{ and } \delta(i+1)=0}} \beta_A + \sum_{\substack{i \in X_l - X_r \\ \text{s.t. } \delta(i)=0}} \alpha_A, && \text{// Gaps in sequence } A
\end{aligned}$$

assuming a gentle abuse of notation, in which $f(i) = \{\mu(i) \text{ if } \delta(i) = 1, \text{ or } \perp \text{ otherwise}\}$.

A dynamic programming algorithm follows from this general recurrence equation, in a standard way. The cost of the best alignment is given by $min\{C_f^0 \mid f \in \mathcal{F}|_{X_0}\}$. A simple backtrack procedure gives the best alignment between A and B. The following theorem gives the worst-case complexity of the algorithm.

Theorem 1. *Let A and B be two arc-annotated sequences ($P_B = \varnothing$), and let (X, T) be a tree decomposition of A. The structure-sequence alignment of A and B can be computed in $\Theta(N \cdot m^{t+1})$ time and $\Theta(N \cdot m^t)$ in space, where $N = |X|$, t is the tree-width of (X, T), and $m = |B|$.*

This complexity can be further improved when the tree decomposition is smooth (or even only k-weakly smooth), by taking advantage of the affine nature of gap penalty functions. To that purpose, one uses the general principle underlying Gotoh's algorithm [4], and introduce a secondary matrix to distinguish gap openings from gap-extensions. We obtain the dynamic programming equation summarized in Figure 4

Theorem 2. *If the tree decomposition of A is k-weakly smooth, then the sequence-structure alignment of A and B, can be computed in $\Theta(k \cdot m^{t+1} + (N - k) \cdot m^t)$ time.*

Corollary 1. *Let A and B be as before. If the tree decomposition (X, T) of A is smooth and has width t, then time complexity of the structure-sequence alignment algorithm is in $\Theta(N \cdot m^t)$.*

4 Tree Decomposition and Sequence Structure Alignment of RNA Structures

In its full generality, the problem of computing a tree decomposition of minimum width for an arc-annotated sequence is NP-Hard [1]. However, by restricting the problem to some specific RNA structure families, one can obtain a tree decomposition with a small width in reasonable time. The key idea relies on a total ordering of the positions in the arc-annotated sequence, as shown in the following. For the sake of clarity, we suppose at first that there is no unpaired position in the structure.

Definition 9. *A **wave embedding** W of an arc-annotated sequence $A = (S, P)$ is defined by an increasing sequence of **pivot** positions $\mathbf{y} = \{y_i\}_{i=0}^k$, such that $y_0 := 1$ and $y_k := n$ (Figure 5). The **degree** of a Wave Embedding is its number of pivots minus one.*

Now, a total ordering on the interactions can be inferred from a wave embedding, in which case the wave embedding is said to be **ordering**. Let us now give a sufficient condition for a given embedding to be ordering. To that purpose, we introduce the **upward graph**, and show that its acyclicity is a sufficient condition for the embedding to be ordering.

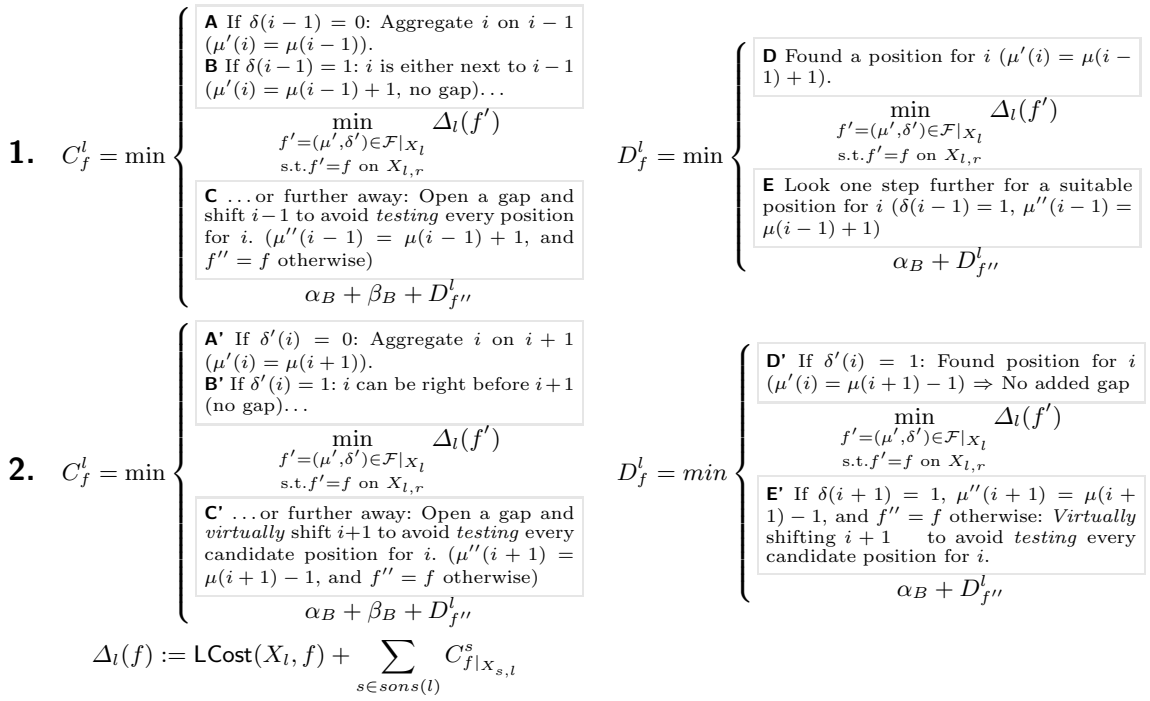

Fig. 4. Dynamic programming equation for aligning a smooth bag X_l, whose father X_r has previously been assigned. Case **1.** and **2.** above apply respectively to smooth bags such that $i-1 \in X_r$ or $i+1 \in X_r$ (note that these two cases are mutually exclusive from the definition of smoothness).

Given a wave embedding W of an arc-annotated sequence $A = (S, P)$, we call **intervals of** W the half open intervals: $I_t = [y_t, y_{t+1}[$ for $t \in [1, k-2]$ and the interval $I_k = [y_{k-1}, y_k]$. Now, let us start by defining a partial order $\cdot \prec \cdot$ on the positions of a single interval, as follows: for any $i, j \in I_t$, one has $i \prec j$ if either $i < j$ and t is odd, or $i > j$ and t is even. This relation can be used to define the position **directly below** i, i.e. the closest position i^- to i such that $i^- \prec j$. In other words, one has $i^- = i-1$ if $i, i-1$ belong to an odd interval, and $i-1 \in I_t$ or $i^- = i+1$ if $i, i+1$ belongs to an even interval. In the absence of such a position, we set $i^- = 0$. The highest position of an interval I_t is the position $i \in I_t$ such that there is no position j in I_t with $i \prec j$. Now we can define the upward graph:

Definition 10. *Given a wave embedding W of an arc-annotated sequence $A = (S, P)$, the **upward graph** of A associated to W is the directed graph $G = (V_G, A_G)$ such that $V_G = P$, and A_G is the set of arcs $((i, j) \mapsto (i', j'))$ such that i or j is directly below i' or j' in W.*
*A wave embedding of an arc-annotated sequence $A = (S, P)$ is **ordering** if its upward graph is acyclic.*

Algorithm 1 takes as input an upward graph, supposed to be acyclic, and assigns a level for each vertex, thus a level for each interaction of the associated arc-annotated sequence . This algorithm is a straightforward modification of Kahn's topological ordering algorithm [7], illustrated in Figure 6.

Now we define the level of any position as the minimum level of all interactions in which the position is implicated, as illustrated in Figure 6.

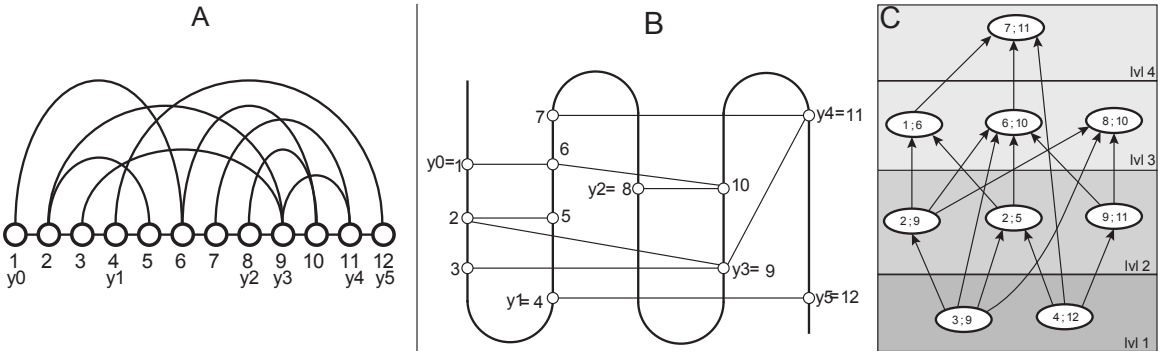

Fig. 5. A: An arc-annotated sequence with its pivots. B: An embedding wave representation of the same arc-annotated sequence. C: The associated (acyclic) upward graph (interactions are rank by their level).

Algorithm 1: Level Algorithm

Input : a directed acyclic graph $G = (V_G, A_G)$
Output: Assign a level for each vertex
- $L = \{v \in V_G,\ d^-(v) = 0\}$
- **For each** $v \in L$, $level(v) = 1$
- **While** $L \neq \varnothing$:
  - pop front v from L.
  - **For each** v' such that $(v, v') \in A_G$:
    - remove (v, v') from A_G
    - **If** $d^-(v') = 0$:
      - push back v' into L
      - $level(v') = level(v) + 1$

Definition 11 (Level of a position). *Given an ordering wave embedding W of an arc-annotated sequence $A = (S, P)$, the* ***level of a position*** *i defined by: $level(i) = \min_{(i,j)\in P}(level((i, j)))$. We define a* ***total order*** *$\preccurlyeq$ on S through: $i \preccurlyeq j$ iff either $level(i) < level(j)$, or $level(i) = level(j)$ and $i < j$.*

Then, we introduce Algorithm 2 which, starting from an ordering wave embedding, decomposes any arc-annotated sequence. The key idea is to create a root which contains the highest position in each interval. The successor of a bag is then obtained by changing the highest level position into the position directly below it (Figure 6).

Theorem 3. *Given an ordering wave embedding of degree k for an arc-annotated sequence A, then a tree decomposition of A having width k can always be computed in time $O(k \cdot n)$.*

Corollary 2. *Let A and B be two arc-annotated sequences with $P_B = \varnothing$. Given an ordering wave embedding of degree k of A, the structure-sequence alignment of A and B can be computed in $O(n \cdot m^k)$.*

5 Application to three general classes of structures

In this section, we define three new structure classes, which respectively generalize the standard pseudoknots [5], the simple non-standard pseudoknots [11] and the standard triple helices [10]. For each of them, a *natural* ordering wave embedding can be found, such that our general alignment algorithm has the same complexity as its, previously introduced, *ad hoc* alternatives.

Algorithm 2: Chaining Algorithm

Input : an arc-annotated sequence A and an ordering wave embedding of degree k
Output: A tree decomposition of A

- Assign a level for each interaction using Algorithm 1, and map level to each position
- $X = \varnothing$ and T is an empty tree
- Create a node X_0 composed of the highest position of each interval and set X_0 as the root of T
- $l = 0$
- **While** there is a position $i \in X_l$ such that $i^- \neq 0$
  - Search the position $p \in X_l$ with the highest level and such that $p^- \neq 0$
  - Add p^- to X_l
  - Add X_l to X
  - If $l > 0$, set X_l as the son of X_{l-1}
  - Set $l = l + 1$ and $X_l = X_{l-1} - \{p\}$
- **Return** (X, T)

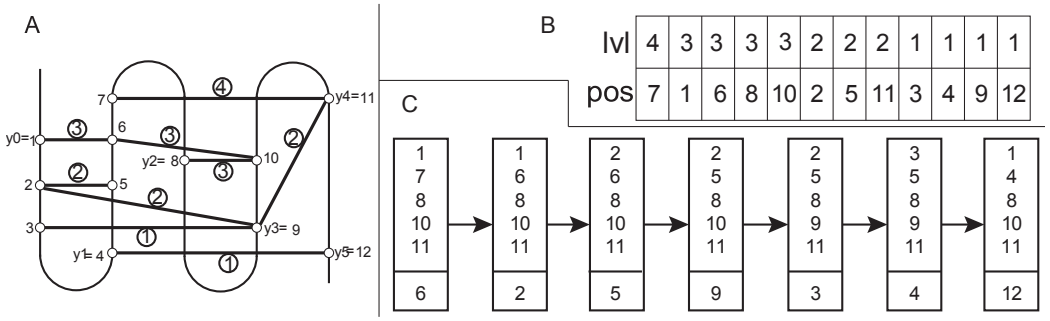

Fig. 6. A: Arc-annotated sequence and its pivots. B: Ranking of the positions by decreasing for the level. C: Tree decomposition obtained with Algorithm 2. The highlighted position in each bag is the position denoted as position p within Algorithm 2. The last position in each bag is the position p^-.

5.1 Standard Structures

Here we define and describe the alignment of **standard structures**, a natural generalization of the standard pseudoknots defined by Han *et al* [5]. The main specificity of this class is that bases can interact with several other bases. This allows the consideration oft multiple non-canonical interactions (e.g. base triples) in RNA structures (see Figure 7.A).

Definition 12 (Standard Structure). *An arc-annotated sequence $A = (S, P)$ is a **standard structure** if there exists an ordering wave embedding, based on a pivot list $\mathbf{y} = \{y_i\}_{i=0}^{k}$, $k > 1$, such that the extremities of any interaction $(i, j) \in P$ are separated by exactly one pivot.*

The ordering wave embedding can then be used by Algorithm 2 to yield a smooth tree decomposition of width k, therefore the complexity of the structure-sequence alignment is $O(n \cdot m^k)$.

5.2 Simple Non-Standard Structures

In [11], the algorithm of [5] is extended to capture the so-called simple non-standard pseudoknots. Briefly, a simple non-standard pseudoknot contains a standard pseudoknot, and defines a special region from which interactions may initiate, possibly crossing interactions in the standard pseudoknot. We extend this class in order to capture multiple interactions, as illustrated by Figure 7.B.

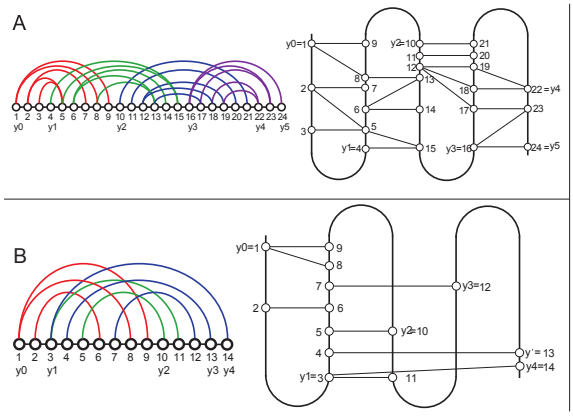

Fig. 7. A: A standard structure and one of its ordering wave embedding representation. B: A simple Non-Standard structure and one of its ordering wave embedding representation ($k' = 2$). C: An extended triple helix and one of its ordering wave embedding representation.

Definition 13 (Simple Non-Standard Structure). *An arc-annotated sequence $A = (S, P)$ is a* ***simple non-standard structure*** *(Type I) if there exist an ordering wave embedding, based on a pivot list $\mathbf{y} = \{y_i\}_{i=0}^{k}$, $k > 1$ and $\tau \in [1, k - k' - 1]$, $k' \in \{1, 2\}$, such that the extremities of any interaction $(i, j) \in P$ with $j < y_{k-k'}$ are separated by exactly one pivot and the others interactions $(i, j) \in P$ with $y_{k-k'} \leq j$ are such that $y_{\tau-1} \leqslant i < y_\tau$.*

As in [11], Type II simple non-standard structures are symmetric to Type I: the special region lies on the beginning of the sequence. To be coherent with the definition of simple non-standard pseudoknots given in [11], we define the **degree** of a standard structure as it number of pivots in its ordering wave embedding. Therefore, the treewidth of a simple non-standard structure of degree $k + 1$ is at most $k + 1$ and, given its pivots sequence, we can build a smooth tree decomposition of width $k + 1$. Hence the complexity of the structure-sequence alignment is $O(n \cdot m^{k+1})$.

5.3 Extended Standard Triple Helices

To our knowledge, the standard triple helices [10] constitute the first attempt to handle base triples in sequence/structure alignments. A standard triple helix is a kind of standard pseudoknot of degree 3 where some positions are allowed to be involved in multiple base pairs.

We define the **extended standard triple helix** as the structures admitting an ordering wave embedding of degree 3 (Figure 7.C). This new class strictly includes standard triple helices. Furthermore, each such structure can be represented by a tree-decomposition which is smooth and has width at most 3. This gives an algorithm in $O(n \cdot m^3)$ for the structure-sequence alignment.

6 Recursive Structures

Now we consider much more general RNA structures, where different kinds of pseudoknots can occur anywhere. As will be seen below, such structures can be decomposed into **primitives**, and from the tree-decomposition of each primitive a global tree-decomposition of the structure can be built. The set of **primitive sub-arc-annotated subsequences** (**primitives** for short) of an arc-annotated sequence is the set of all sub-arc-annotated sequences induced by the connected components of its **conflict graph**, which is defined as follows. The conflict graph $G = (V, E)$ of an arc-annotated sequence $A = (S, P)$ is the graph such that:

- $V = P$ (the nodes of G are the interactions of A).
- $(v_1, v_2) \in E$ with $v_1 = (i_1, j_1)$ and $v_2 = (i_2, j_2)$ $(i_1 < i_2)$ iff $i_1 < i_2 < j_1 < j_2$ (interactions cross).

Algorithm 3: Recursive Algorithm

- Compute a tree decomposition for all primitive extensions using Algorithm 2.
- For each primitive A_0 of depth 0:
  - Create a bag χ containing the right boundary of A_0 and the left boundary of its next primitive A'_0.
  - Let (X^0, T^0) be the tree decomposition of the extension of A_0 and (X'^0, T'^0) the one of the extension of A'_0.
  - Add to χ the position i of the root of (X'^0, T'^0) such that $i-1$ or $i+1$ belongs to the root too.
  - Connect the leaf of (X^0, T^0) to χ and connect χ to the root of (X'^0, T'^0).
- For each possible depth $i > 0$ in increasing order:
  - For each primitive A of depth i:
    - Let (X, T) be the tree decomposition of the extension of A and (X', T') the one of the extension of the arc-annotated sequence A' in which A is encapsulated.
    - Find a bag χ in (X', T') such that χ contains the boundaries of A.
    - Connect (X, T) to (X', T') by connecting the root of (X, T) to χ.
    - Add the right-most boundary of A in (X, T) to all bags from the root to the first bag containing it.

The **boundaries** of a primitive are its left-most and right-most positions.

Let A and A' be two primitives of an arc-annotated sequence, and let i' and j' be the boundaries of A' ($i' < j'$). We say that A is **encapsulated** in A' iff for any position $i \in A$, one has $i' \leq i \leq j'$, and there exists at least one position $j \in A$ such that $i' < j < j'$. The **depth** of a primitive of an arc-annotated sequence is the number of primitives which encapsulate it. We say that A is **directly** encapsulated in A' if A is encapsulated in A' and $depth(A) = depth(A') + 1$.

The **extension** of a primitive A of depth i is the arc-annotated subsequence consisting of: the primitive A; the boundaries of any primitive that is directly encapsulated in A; and the unpaired positions that are directly encapsulated in A.

Given a primitive A_0 of depth 0 of an arc-annotated sequence A, the **next primitive** of A_0 is the following primitive of level 0 in the sequential order (note that they can share a boundary).

Theorem 4. *Let A be an arc-annotated sequence of size n. If there exist an ordering wave embeddings of degree at most k for each extension of its primitives, then the treewidth of A is at most $k+1$, and a κ-weakly smooth tree decomposition of A can be built in $O(k \cdot n)$ time, where κ is the number of primitives of odd degree, whose level is greater or equal to 1.*

Corollary 3. *Let A and B be two arc-annotated sequences with $P_B = \varnothing$. Given an ordering wave embedding of degree k (at most) for each extension of the primitives of A, the structure-sequence alignment of A and B can be computed in $O(\kappa \cdot m^{k+1} + n \cdot m^k)$ (with κ defined in Theorem 4).*

7 Conclusion

We have given a general parameterized dynamic programming scheme for sequence-structure comparison in a large class of RNA structures, which unifies and generalizes several families of structures that have been independently considered by previous works. Notably, we can handle structures where each nucleotide can be paired to any number of other nucleotides, thus cqpturing any type of non-canonical interactions. Our approach relies on a tree decomposition approach of arc-annotated sequences represented as wave embeddings, and the treewidth of the decomposition is then equal to the degree of the wave embedding. Computing a wave embedding of small degree is easy for all classes of pseudoknotted structures considered in this paper. However, the problem of finding a minimum degree wave embedding for any kind of pseudoknotted structure remains open.

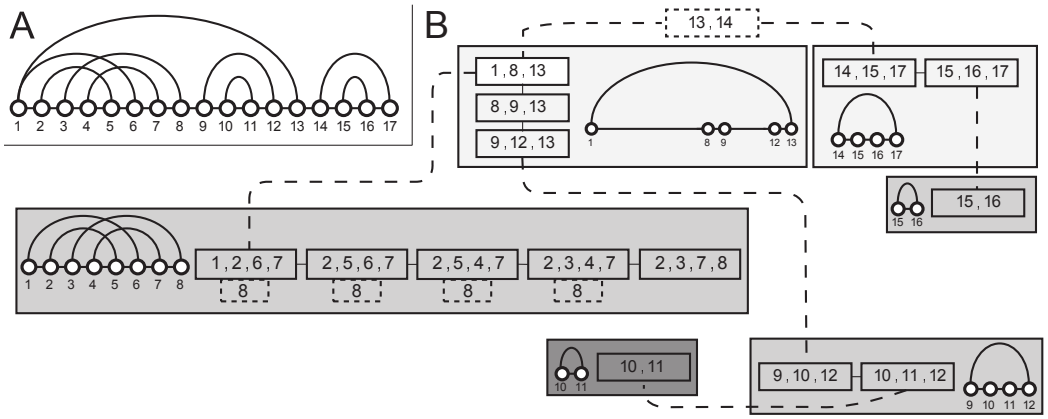

Fig. 8. (A) An arc-annotated sequence and (B) a representation of the tree decomposition given by Algorithm 3. Each box correspond to an extension of a primitive and its associated tree decomposition. Dashed links correspond to the connections made by Algorithm 3. The Dashed bag (top of B) correspond to the bag χ added to connect two consecutive primitive of level 0. The position 8 in a dashed box illustrates the case where the right boundary of a primitive need to be added in its tree decomposition.

8 Acknowledgments

This work was supported by the DIGITEO RNAOmics project (AD, PR and YP), the ANR AMIS ARN project (ANR-09-BLAN-0160, AD and PR) and the ANR MAGNUM project (ANR-2010-BLAN-0204, YP).

References

1. Arnborg, S., Corneil, D., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Alg. Disc. Meth. 8(2), 277–284 (1987)
2. Blin, G., Denise, A., Dulucq, S., Herrbach, C., Touzet, H.: Alignment of RNA structures. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7(2), 309 – 322 (2010)
3. Bodlaender, H.L.: Treewidth: Algorithmic techniques and results. In: MFCS-97. Lecture Notes in Computer Science, vol. 1295, pp. 19–36 (1997)
4. Gotoh, O.: An improved algorithm for matching biological sequences. J Mol Biol 162(3), 705–708 (1982)
5. Han, B., Dost, B., Bafna, V., Zhang, S.: Structural alignment of pseudoknotted RNA. Journal of Computational Biology 15(5), 489–504 (2008)
6. Jiang, T., Lin, G.H., Ma, B., Zhang, K.: A general edit distance between RNA structures. Journal of Computational Biology 9(2), 371–388 (2002)
7. Kahn, A.B.: Topological sorting of large networks. Communications of the ACM 5(11), 558–562 (1962)
8. Leontis, N.B., Westhof, E.: Geometric nomenclature and classification of RNA base pairs. RNA 7, 499–512 (2001)
9. Rødland, E.A.A.: Pseudoknots in RNA secondary structures: representation, enumeration, and prevalence. Journal of Computational Biology 13(6), 1197–1213 (2006)
10. Wong, T.K., Yiu, S.M.: Structural alignment of RNA with triple helix structure. Journal of Computational Biology 19(4), 365–78 (2012)
11. Wong, T., Lam, T., Sung, W., Cheung, B., Yiu, S.: Structural alignment of RNA with complex pseudoknot structure. Journal of Computational Biology 18(1) (2011)
12. Zhang, K., Wang, L., Ma, B.: Computing similarity between rna structures. In: CPM-99. Lecture Notes in Computer Science, vol. 1645, pp. 281–293 (1999)