Practical structure-sequence alignment of pseudoknotted RNAs

Wei Wang

NNT : 2017SACLS563

Thèse de doctorat de l'Université Paris-Saclay préparée à l'Université Paris-Sud

Ecole doctorale n°580 (STIC)
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Informatique

par

M. Wei WANG

Alignement pratique de structure-séquence d'ARN avec pseudonœuds

Thèse présentée et soutenue à Orsay, le 18 Décembre 2017.

Composition du Jury :

Mme	Hélène TOUZET	Directrice de Recherche CNRS, Université Lille 1	(Présidente)
M.	Guillaume FERTIN	Professeur Université de Nantes	(Rapporteur)
M.	Jan GORODKIN	Professeur University of Copenhagen	(Rapporteur)
Mme	Johanne COHEN	Directrice de Recherche CNRS, Université Paris-Sud	(Examinatrice)
M.	Laurent BULTEAU	Chargé de Recherche CNRS, Université Paris-Est	(Examinateur)
M.	Alain DENISE	Professeur Université Paris-Sud	(Directeur de thèse)
M.	Yann PONTY	Chargé de Recherche CNRS, École Polytechnique	(Co-encadrant)

Résumé

L'alignment de macromolécules telles que les protéines, les ADN et les ARN afin de révéler ou d'exploiter à l'inverse leur homologie fonctionnelle est un défi classique en bioinformatique, avec des applications de grande envergure dans la modélisation de la structure et l'annotations du génome. Dans le contexte spécifique des ARN complexes, présentant des pseudoknots, des interactions multiples et des paires de bases non canoniques, de multiples solutions et outils algorithmiques ont été proposés pour le problème d'alignement de séquence de structure. Cependant, de tels outils sont rarement utilisés dans la pratique, en partie à cause de leurs exigences de calcul extrêmes, et de leur incapacité à supporter des types généraux de structures.

Au chapitre 2, nous illustrons d'abord les opérations d'édition pour calculer le coût d'un alignement et la définition du problème d'alignement structure-séquence. Ensuite, nous expliquons plusieurs algorithmes d'état de l'art pour le problème. Ces algorithmes comprennent l'algorithme de Han [39] avec le programme `PAL`, l'algorithme de Matsui [60] avec le programme `PSTAG`, l'algorithme de Song [91] et l'algorithme de Rinaudo [78]. Tous ces algorithmes sont principalement conçus pour prédire l'alignement de la structure-séquence pseudo-notée. Cependant, comparé aux autres algorithmes, Rinaudo *et al.* a donné une méthode entièrement générale pour la comparaison séquence-structure, qui est capable de prendre en entrée n'importe quel type de structures pseudo-notées. Mon travail est basé sur l'algorithme de Rinaudo.

Au chapitre 3, je décris d'abord quelques détails sur la mise en œuvre de notre nouveau programme `LiCoRNA` (aLignment of Complet RNAs) incluant le schme de scoring et la programmation dynamique par bandes qui permettent d'accélérer la programmation dynamique sans perdre trop de précision. Ensuite, trois algorithmes seront introduits pour obtenir

les alignements sous-optimaux de la structure-séquence, l'un est un algorithme stochastique de backtracking basé sur la fonction de partition, l'un est l'algorithme d'alignement Δ-sous-optimal et l'un est l'algorithme d'alignement suboptimal K-best. Notre raisonnement pour explorer l'espace des alignements sous-optimaux, ou quasi-optimaux, est triple: Premièrement, un alignement optimal est ambigu, ce qui signifie que plusieurs alignements ayant le même score peuvent coexister. Deuxièmement, un alignement optimal n'est qu'une approximation de celui qui est biologiquement pertinent. Troisièmement, un alignement optimal peut être sensible aux perturbations des paramètres d'évaluation, en particulier les pénalités d'écart. Sur la base de la fonction de partition et de l'algorithme intérieur-extérieur, on peut également calculer la probabilité de concordance de Boltzmann. En outre, nous introduisons la notation Maximum Expected structure-sequence Alignment (MEA) pour calculer un alignement avec une précision maximale prévue sur un ensemble d'alignements. L'alignement MEA peut être intuitivement comparé au centre de masse de l'espace d'alignement. Notre raisonnement est que, si l'alignement est bien défini, alors l'alignement MEA devrait être proche de l'alignement optimal. Inversement, un alignement optimal mal défini, admettant de nombreux alignements sous-optimaux, devrait être soit éloigné' du MEA, soit avoir une faible précision associée.

Le chapitre 4 illustre les résultats des tests de `LiCoRNA` qui sont principalement divisés en deux parties. La première partie consiste à évaluer la performance de `LiCoRNA` sur la base des séquences de graines dans les familles pseudoknotted dans RFAM par la comparaison avec d'autres programmes à la fine pointe `PAL`, `PSTAG` et `Profile-csHMMs` [120]. Les paramètres d'évaluation sont la sensibilité, valeur prédictive positive et l'AFI. Le résultat principal est que LiCoRNA peut prédire les alignements par paires pour toutes les familles et génère généralement des résultats équivalents ou meilleurs que ses concurrents pour presque toutes les familles. La deuxième partie est que nous traitons les alignements pseudoknotted complets RFAM en utilisant `LiCoRNA`. Le résultat montre que `LiCoRNA` prend en charge pseudoknots sans perte d'identité de séquence en calculant le pourcentage de paire de bases pour chaque position de paire dans la structure de référence.

Le dernier chapitre présente les perspectives. Tout d'abord, le séchma de score pour `LiCoRNA` est indépendant de la position qui signifie que les scores pour les substitutions et les lacunes sont les mêmes dans différentes positions de l'ARN. Nous pouvons étendre au schéma de score basé sur le profil et au modèle évolutionnaire probabiliste pour améliorer encore la précision. Deuxièmement, la programmation dynamique basée sur la décomposition arborescente peut également être utilisée pour identifier des motifs 3D d'ARN qui sont définis par leur propre modèle d'appariement de base non-canonique.

Acknowledgements

First of all, I owe my deepest gratitude to my advisors, Alain Denise, Yann Ponty for their patience, encouragement and immense knowledge. Without their guidance, I would not have finished my PhD study smoothly.

Beside my advisors, I would like to thank the rest of my thesis committee for their time and extreme patience. Thanks to Guillaume Fertin and Jan Gorodkin for their insightful comments and hard questions which incented me to widen my research. I would also like to the other members Hélène Touzet, Johanne Cohen, Laurent Bulteau for their time and insightful questions.

My thanks also go to all the colleagues in the bioinformatics group in LRI and LIX for their discussions and for solving everyday trouble during my PhD.

Thanks to the China Scholarship Council, who provided financial support during my graduate studies.

I am forever indebted to my parents for their unconditional supports. I also thank JiaYin XUE for her accompany. Last but not least, I'd like to thank all my friends in France for their help and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Our knowledge of the role of RNA has changed since nucleic acids were discovered in 1868 by Friedrich Miescher [20]. In 1952, James Watson sketched the "central dogma" [50]. As DNA was located in the cell nucleus, proteins were synthesized in the cytoplasm and RNAs were detected in both cytoplasm and nucleus, he speculated that there must be a coding RNA that carries genetic information from DNA to protein. Then the RNA which is called mRNA was proved to carry the genetic information from DNA to the ribosomes [13]. Later, ribosomal ribonucleic acid (rRNA) as a component of ribosome was found [37]. rRNA is important in evolution, and genes that encode rRNA are used to identify an organism's taxonomic group [90]. Based on Francis Crick's "adaptor" hypothesis [19], transfer RNA (tRNA) was observed [42] and proved [76] to be the adaptor that serves as the physical link between the mRNA and the amino acid sequence of proteins.

Several abundant, small non-mRNAs, other than rRNA and tRNA named small nuclear RNAs (snRNAs) like U1, U2, U4, U5 and U6 were discovered since 1993. They participate in the splicing of mRNAs [121] by forming ribonucleo-protein (RNP) complexes. The discovery of Ribonuclease P [38] was another landmark in RNA biology. Since then RNA is known to play an important role in cellular processes, not just carrying genetic information. The discovery of catalytic RNA leads to the RNA World Hypothesis which means that RNA may play a key role in prebiotic evolution before the evolution of DNA and protein which have more specialized functions. The concept of the RNA world was first proposed in 1962 [69] and coined in 1986 [33]. Since then, the number of new functional RNAs has greatly expanded [104, 71].

After briefly illustrating the history of RNA, we now focus on the details of RNA.

1.1 Types of RNA

RNA molecules are single stranded nucleic acids consisting of nucleotides adenine (A), guanine (G), cytosine (C) and uracil (U). Each nucleotide contains three components: a nitrogenous base, a five-carbon sugar (ribose) and a phosphate group.

RNAs are basically classified into two classes: coding RNA and non-coding RNA. Coding RNA refers to the RNA that encodes protein. mRNA is the only known example. Non-coding RNAs (ncRNAs) are functional RNA molecules that do not encode proteins. Similar to proteins, ncRNAs usually form a specific tertiary structure to perform functions. Proteins with similar structures usually have similar sequences. However, ncRNAs often have a conserved secondary structure which may be better preserved than the RNA sequence.

Non-coding RNAs include two classes of notable interest: small ncRNAs and long ncRNAs (lncRNAs) [84]. Small ncRNAs which are the subjects of intensive translational research contain microRNAs (miRNAs), small interfering RNAs (siRNAs), piwi nucleolar RNAs (piRNAs), transcription initiation RNAs (tiRNAs) and so on. lncRNAs (> 200 nucleotides) are mainly involved in cellular processes including alternative splicing and nuclear import [74]. lncRNAs contain long intergenic non-coding RNAs (lincRNAs), Telomere-associated ncRNAs (TERRAs), Transcribed- ultraconserved regions (T-UCRs) and so on.

1.2 RNA structure

Understanding the functions of ncRNAs is clearly not possible without knowing the structures. This section describes the basic ideas about RNA structure. RNA structure is usually divided into three levels: primary, secondary and tertiary structure which is analogous to the classification of protein structure. RNA primary structure refers to the sequence bases in the nucleic acid chain formed by the phosphodiester bonds. To completely determine the primary structure, one has to purify RNA from the source and characterize it by using sequencing methods [54] at the same time to get rid of the effect of post-transcriptional modifications.

RNA secondary structure is a two-dimensional representation of Watson-Crick base pairs (C-G and A-U), the slightly weaker wobble base pairs (G-U) and the

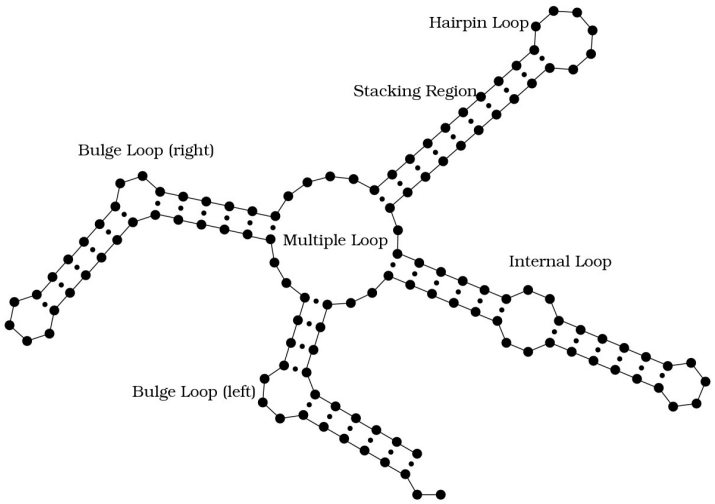


Figure 1.1: Typical elements found in RNA secondary structures (from [92])

"unpaired" regions. Figure 1.1 gives an example of standard elements found in RNA secondary structure, called bulge loop, multiple loop, internal loop, stacking region and hairpin loop [92]. There are two main computational ways to get the secondary structure of ncRNA from primary structure: (1) using secondary structure prediction programs to fold primary sequence directly; (2) comparative sequence analysis which is a relatively more reliable mean [116]. To verify the correctness of the computation result, one needs to use the biochemical techniques to probe the solution structure of a particular ncRNA [4].

Based on secondary structure, **pseudoknotted structure** is formed by pairing between a loop and a region located outside (upstream or downstream) of the stem flanking the loop [96]. Since the first RNA pseudoknot was discovered [73], more and more pseudoknots were found. Now more researches have shown that pseudoknots are essential elements of topology of many structural RNAs, like rRNAs [15], tmRNAs [126]. Some pseudoknots are catalytically active, for example, group I self-splicing intron in which the pseudoknot established the catalytic core [1]. Besides the catalytic activities, pseudoknots also play a key role in altering gene expression by inducing ribosomal frameshifting [29]. Nowadays, there is no accepted nomenclature all kinds of pseudoknots. Therefore, I only introduce several common pseudoknots:

- The H-type pseudoknot. An H-type pseudoknot is formed by the interaction between nucleotides in a hairpin loop in the secondary structure and the single-

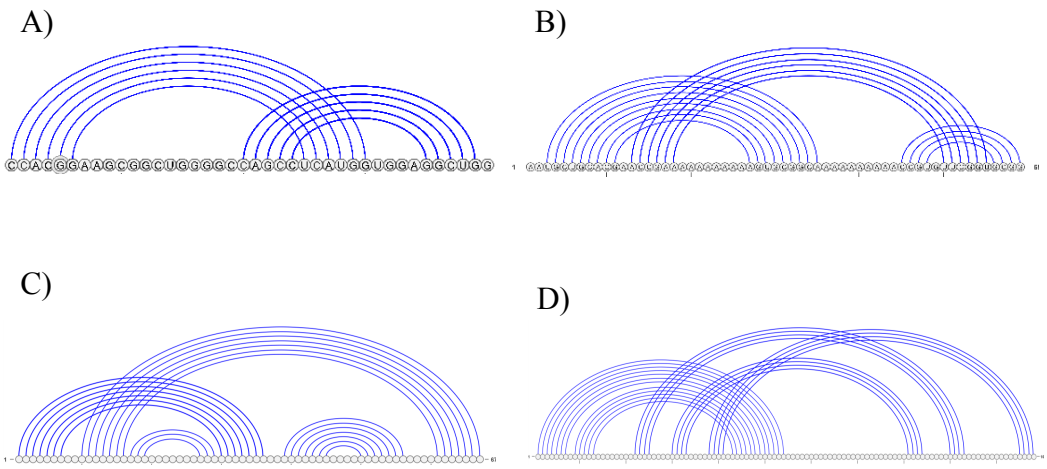


Figure 1.2: A) H-type pseudoknot. B) Kissing hairpin.

stranded region as shown in Figure 1.2 A)

- The kissing hairpin pseudoknot. A Kissing hairpin forms when unpaired nucleotides in a hairpin region interact with unpaired nucleotides in another hairpin region as shown in Figure 1.2 B).
- The recursive pseudoknot. A pseudoknot can further have secondary structure elements (including pseudoknots) in the loops as shown in Figure 1.2 C).
- The complex pseudoknot. The complex pseudoknot contains more complex than the former three types. One example is shown in Figure 1.2 D).

The **tertiary structure** is the three-dimensional arrangement of the secondary structure elements which are associated through numerous van der Waals contacts, specific hydrogen bonds via the formation of a small number of additional Watson-Crick pairs and/or unusual pairs involving hairpin loops or internal bulges [111]. Usually, the tertiary structure is determined by the experiment using X-ray crystallography and NMR spectroscopy. In fact, RNA nucleotides can interact with other nucleotides through many different ways: (1) base with base, (2) base with ribose sugar, (3) base with phosphate, (4) ribose with ribose, (5) ribose with phosphate, and (6) phosphate with phosphate. Among them, Base-base interaction is the most sequence specific and is the most useful when predicting RNA structure [86]. Actually, as shown in Figure 1.3, each RNA base (purines and pyrimidines) interacts edge-to-edge using

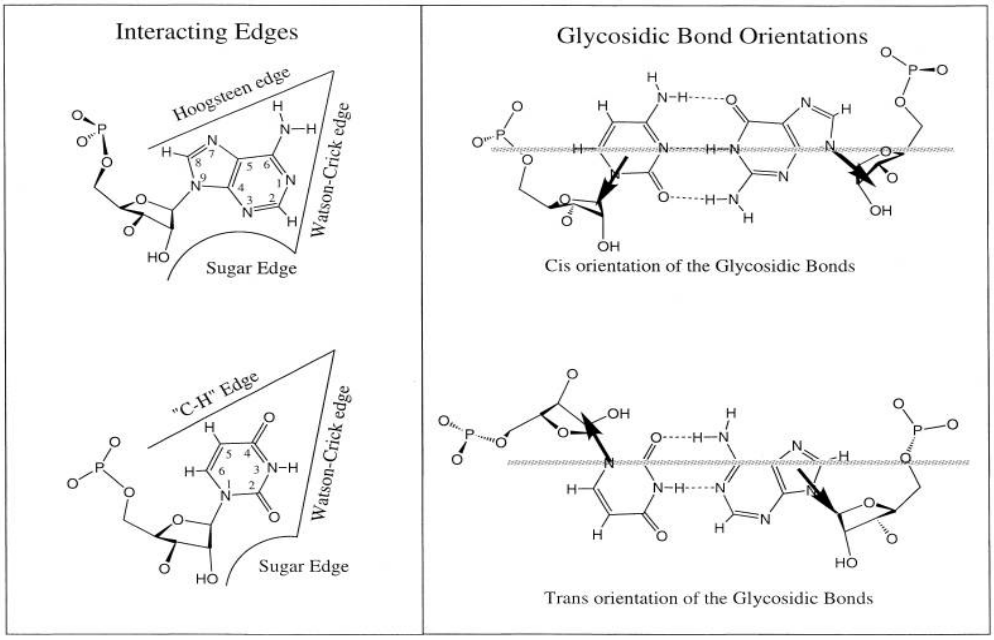


Figure 1.3: (Left) Identification of edges in the RNA bases. The upper part shows the edges of Purine and the down part shows the edges of Pyrimidine (Right) cis versus trans orientation of glycosidic bonds. (from [55]).

one of three edges with other bases. Considering the orientation of glycosidic bonds relative to the hydrogen bonds (*cis* or *trans*), there are 12 families of edge-to-edge base pairs [55] as shown in Table 1.1. The often mentioned Watson-Crick base pair in the secondary structure is a *cis* Watson-Crick/Watson-Crick base pair.

At the tertiary level, many studies have illustrated that the RNA tertiary structure is modular and is composed of recurrent conserved motifs that are embedded in the hairpin loops (HL), internal loops (IL) and multi-helix junction loops (MHJ) in RNA secondary structures [4, 111, 94, 97]. There are several properties for RNA 3D motifs as summarized by Petrov [72]: (1) Modular. They appear as discrete units (2) Autonomous. It means that they can fold into characteristic geometrical structure without being affected by the molecular contexts (3) Recurrent. It means that the 3D motifs can occur over and over in one molecule or in different RNA tertiary molecules (4) Mutipurpose. One motif can bind with a protein in one structural context and can be involved in the interactions with other RNAs in another context.

One last thing to mention is that RNA folds in a hierarchical and sequential manner, which means that the secondary structure forms first, followed by the tertiary

No.	Glycosidic bonds orientation	Interaction Edge
1	Cis	Watson-Crick/Watson-Crick
2	Trans	Watson-Crick/Watson-Crick
3	Cis	Watson-Crick/Hoogsteen
4	Trans	Watson-Crick/Hoogsteen
5	Cis	Watson-Crick/Sugar Edge
6	Trans	Watson-Crick/Sugar Edge
7	Cis	Hoogsteen/Hoogsteen
8	Trans	Hoogsteen/Hoogsteen
9	Cis	Hoogsteen/Sugar Edge
10	Trans	Hoogsteen/Sugar Edge
11	Cis	Sugar Edge/Sugar Edge
12	Trans	Sugar Edge/Sugar Edge

Table 1.1: The 12 families of edge-to-edge base pairs formed by nucleic acid bases

structure [99]. In other words, RNA secondary structure often determines tertiary structure. Therefore, an RNA sequence may form multiple secondary structures and consequently multiple tertiary structures. This implies that correct prediction of RNA secondary structure is a key step for predicting RNA tertiary structure from sequence.

1.3 RNA secondary structure prediction

1.3.1 Secondary Structure

An RNA structure is represented as an arc-annotated sequence (S, P) where S is a string of characters over $\Sigma = \{A, C, G, U\}$ and P is a set of arcs which is a set of pairs of positions in S connecting two distinct characters. There are four levels of arc-annotated sequences as mentioned by Blin *et al.* [8] and originally proposed by Evans [30]:

- UNLIMITED - no restriction.
- CROSSING - any position has at most one incident arc.
- NESTED - any position has at most one incident arc and no arcs are crossing.

- PLAIN - no arc.

The common secondary structures involving A-U, C-G and G-U belong to the NESTED structure. Pseudoknotted structure which belongs to the CROSSING structure follows the same base pairing rules, but allows crossing arcs [24]. If we ignore the spatial coordinates of the tertiary structure and just draw it planarly, it will normally belong to UNLIMITED structures involving non-Watson-Crick base pairs. In this thesis, we focus on RNA secondary structures including pseudoknots.

Finding the structure of a ncRNA is often the first step to explain its function. There are two general computational methods to predict the secondary structure of an RNA: *De novo* folding approach when we only have one single sequence and *comparative* approach when we have additional homologous sequences at hand.

I will first describe *De novo* folding approach. Historically, the Nussinov algorithm [70] is the first attempt to solve the RNA folding problem (predicting RNA secondary structure from single-stranded RNA). The idea is to maximize the number of base pairs in the structure. It is a simple and efficient dynamic programming algorithm which starts by calculating the maximum number of base pairs for the smallest subsequences and extends to larger and larger subsequences. However, this algorithm does not give accurate results. There are several reasons. For example, affected by the stacking interaction, the stability of a base pair varies according to the neighboring base pairs. Besides, the algorithm does not distinguish loop size. The energy minimization algorithm proposed by Zuker and Stiegler [125] gave an improvement of the Nussinov algorithm. The free energy of an RNA structure is approximated as the sum of the free energies of all the loops and base pair stacks. Zuker dynamic programming algorithm tries to find the minimum free energy (MFE) among all the foldings of a single-stranded sequence. However, there are still limitations for the MFE method. First is the topological limitations which means that this algorithm does not allow pseudoknots. Second, a single-stranded RNA sequence might have multiple conformations. To solve the problem, Wuchty *et al.* designed an algorithm to generate all suboptimal secondary structures within a range Δ from the optimal structure [118] and Ding *et al.* [26] proposed a statistically sampling algorithm to generate the RNA secondary structures according to the Boltzmann equilibrium probability distribution.

Recently, bioinformatician scientists tried to incorporate structure probing data into folding algorithms to increase the accuracy of RNA folding. This method is the combination of structural probing experiment and computational method. One of the most

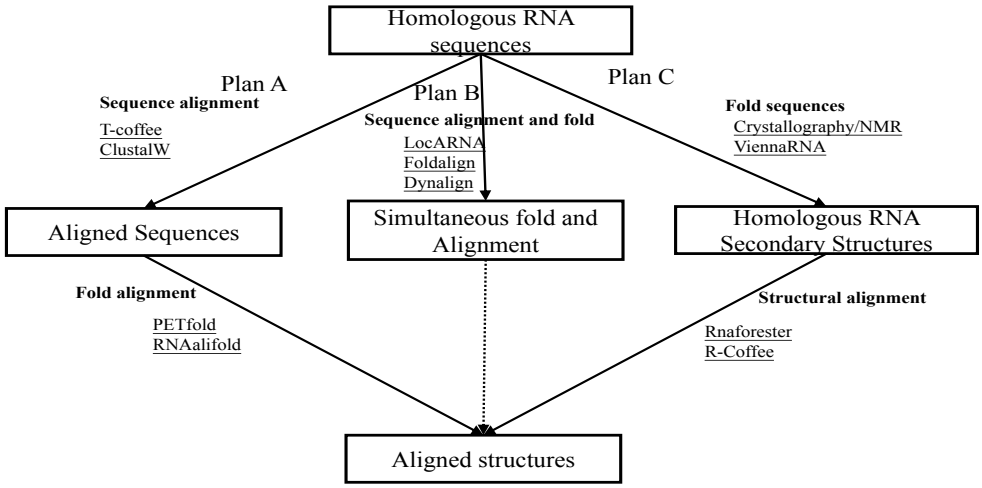


Figure 1.4: Current three approaches to analyze homologous RNA sequences and structures. (from [31])

widely used probing experiments is to detect the paired and unpaired bases. There are two kinds of reagents, one kind is chemical reagents and the other kind is enzyme reagents. Chemical reagents can form stable adducts with nucleotides in the loop regions while in the stacked regions it can not. This kind of reagent includes kethoxal (KT) [14], dimethyl sulfate (DMS) [98], selective 2-hydroxyl acylation analyzed by primer extension (SHAPE) [112]. The enzyme reagents are RNases which catalyze the degradation of the single- or double-stranded regions into smaller segments [124]. Deigan *et al.* [23] have used this method to predict the secondary structure of Escherichia coli 16S rRNA ($>$ 1,300 nt) by combining the SHAPE pseudo-free energies and nearest neighbor parameters with high accuracy.

For the *comparative* approach, the input is a list of sequences with assumed structural similarities and the output is the common structural elements and an alignment between sequences. For this approach, Gardner and Giegerich [31] listed three ways as shown in Figure 1.4:

- Plan A: Align the sequences and look for common structural elements in the alignment.
- Plan B: Simultaneous folding and aligning
- Plan C: Fold each sequence and align the structures

Plan A goes in the way: first align and then fold. To align the homologous sequences, the researcher can use the standard multiple sequence alignment tools, like ClustalW, t-coffee. Then the consensus structure can be derived by analyzing the structure neutral mutations meaning that changing the sequence in RNA will not change the structure. Current tools includes PETfold [88], RNAalifold [6].

Plan B follows the way: simultaneously align and fold. The Sankoff Algorithm [85] is the classical algorithm to simultaneously align and infer a consensus structure. However, this algorithm needs extreme amounts of time and memory. Current implementation of this algorithm are LocARNA [113], Foldalign [34], Dynalign [40].

Plan C is to first fold and then align. Plan C will be used in the case where reliable structures for the sequences are known. First all the structure for the homologous sequences will be folded using ViennaRNA(MFE method) or be found through experiment method, like crystallography or NMR. Then, we need to consider how to compare two RNA secondary structures. If we do not consider pseudoknots, RNA secondary structures can be represented as trees and tree edition and tree alignment algorithms have been proposed [122, 49]. The current implementations include Rnaforester [43] and R-Coffee [114].

1.3.2 Pseudoknotted Structures

First, I will introduce *De novo* approaches to predict the pseudoknotted structure. The prediction of RNA secondary structures containing pseudoknots of arbitrary types from a single-strand sequence is NP-hard when an energy model is used [59]. Therefore, quite a lot of algorithms consider only a certain type of pseudoknots into their dynamic programming equations. Based on dynamic programming algorithms proposed by Uemura *et al.* [100], Akutsu [2] (A&U), Rivas and Eddy [79](R&E), Lyngsø and Pedersen [58] (L&P) and Dirks and Pierce [27](D&P), Condon *et al.* [18] devised a hierarchy of the pseudoknot complexity. Each polynomial time algorithm corresponds to a class of pseudoknot that the algorithm can handle. Then the hierarchy could be shown as

$$PKF \subset L\&P \subset D\&P \subset A\&U \subset R\&E$$

Here, PKF represents pseudoknot free secondary structure. Saule *et al.* [87] extended the hierarchy by adding two other algorithms with new classes: Reeder and

Giegerich [77] (R&G) and Cao and Chen [16] (C&C) algorithms. Then in the hierarchy, R&G $\subset$ D&P, L&P $\cap$ R&G $= \emptyset$ and R&G $\not\subset$ L&P and C&C $\subset$ D&P, L&P $\cap$ C&C $\neq 0$, C&C $\not\subset$ L&P and C&C $\subset$ R&G.

Besides *De novo* approaches, *comparative* approaches could also be used for predicting pseudoknotted structures. Next chapter will go into more details about this approaches.

In detail, the dissertation is organized as follows. In Chapter 2, we first illustrate edit operations to calculate the cost of an alignment and the definition of structure-sequence alignment problem. Then we explain several state-of-the-arts algorithms for the problem. These algorithms include Han's algorithm [39] with program `PAL`, Matsui's algorithm [60] with program `PSTAG`, Song's algorithm [91] and Rinaudo's algorithm [78]. Compared with other algorithms, Rinaudo's algorithm is a fully general method for sequence-structure comparison, which is able to take as input any type of pseudoknotted structures. My work is based on Rinaudo's algorithm.

In Chapter 3, I first describe some implementation details about our new program `LiCoRNA` (aLignment of Complex RNAs). Then three algorithms will be introduced to get the suboptimal structure-sequence alignments, one is stochastic backtracking algorithm based on partition function, one is Δ-suboptimal alignment algorithm and one is K-best suboptimal alignment algorithm. Based on the partition function and inside-outside algorithm, one can also compute the Boltzmann match probability. Furthermore, we introduce the notation Maximum Expected structure-sequence Alignment (MEA) to compute an alignment with maximum expected accuracy over a set of alignments.

Chapter 4 illustrates the test results of `LiCoRNA` which are mainly divided into two parts. The first part is to evaluate the performance of `LiCoRNA` based on the seed alignment in the pseudoknotted RFAM families by the comparison with three other state-of-the-art programs `PAL`, `PSTAG` and `profile-csHMMs` [120]. The evaluation parameters are sensitivity/Positive Predictive Value (PPV) and AFI. This experiment mainly answers the following question: How does the predictive capacity of the accuracy of `LiCoRNA` compare with that of other state-of-the-art programs? The second experiment mainly answers the following questions: Covariance models do not consider pseudoknots when aligning, which may lead to misalign when building the full alignment. Does `LiCoRNA`'s support of pseudoknots of arbitrary complexity translate

into better performances? Conversely, is the lack of support for complex pseudoknots detrimental to the quality of the alignments?

The last chapter summarizes the dissertation by the concluding remarks of the previous chapters, and proposes the perspectives of the future work.

Chapter 2

Structure-sequence alignment

This chapter introduces the basic ideas of structure-sequence alignment. We first illustrate edit operations to calculate the cost of an alignment and the definition of structure sequence alignment problem. Then we explain several state-of-the-arts algorithms for the problem for when the arc-annotated structure is PLAIN, NESTED or CROSSING structure, especially Rinaudo's algorithm [78] which is a fixed-parameterized tractable algorithm to handle arbitrary pseudoknots. Finally, the dynamic programming based on tree decomposition can also be formulated as two basic operations **factor maximization** and **factor summation** in variable elimination algorithm which is widely used to solve the maximum a posteriori (MAP) problem in Bayesian networks or Markov networks.

Let us first present some notations. For an RNA structure, an **arc-annotated sequence** is defined as a pair (S, P), where $S = s_1, \ldots, s_n$ represents a sequence over a finite set $\Sigma = \{A, U, G, C\}$ and $P = \{(i, j) \mid 1 \leq i < j \leq n\}$ is a set of paired base positions. A (canonical) base pair is either a Watson-Crick pair (A-U, G-C) or a Wobble pair (G-U). Then the definition of **structure-sequence alignment** between an arc-annotated sequence $Q = (S_Q, P_Q) = (S_Q, P)$ (for short) of length m and a plain sequence $W = (S_W, \emptyset)$ of length n is given as follows.

Definition 1 (**structure-sequence alignment**)**.** A structure-sequence alignment between a arc-annotated sequence Q of length m and a sequence W of length n is a set $A \in \mathscr{A}_{m,n}$, where $\mathscr{A}_{m,n} = \{1 \ldots m\} \times \{1 \ldots n\}$ represents the potential **matches**, such that A respects the following properties. Let θ and γ be two operations (projection) such that, for every $a \in \mathscr{A}_{m,n}$, if $a = (i, j)$, $\theta(a) = i$ and $\gamma(a) = j$. Then

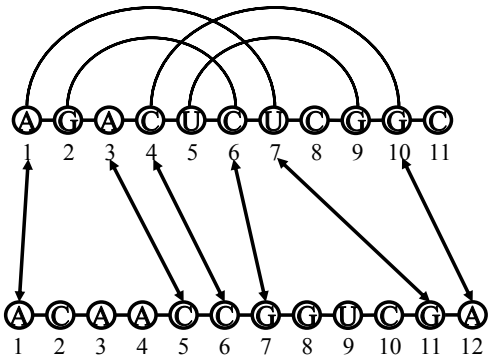


Figure 2.1: A representation of structure-sequence alignment between an arc-annotated sequence and a plain sequence.

- for every $i \in \{1 \dots m\}$, there exists at most one $a \in A$ such that $\theta(a) = i$;
- for every $j \in \{1 \dots n\}$, there exists at most one $a \in A$ such that $\gamma(a) = j$;
- for every $a, b \in A$, $\theta(a) < \theta(b) \Rightarrow \gamma(a) < \gamma(b)$

Figure 2.1 illustrates an example of a structure-sequence alignment. Given an alignment A, if $i \in S_Q$ is not in a match, we denote it by $(i, \perp)$. Likewise, $(\perp, j)$ represents that $j \in \{1 \dots n\}$ is not in a match in A. A **gap** is the position with $(i, \perp)$ or $(\perp, j)$ in A. Then we extend $\mathscr{A}_{m,n}$ to $\mathscr{A}'_{m,n} = \mathscr{A}_{m,n} \cup \{1 \dots m\} \times \{\perp\} \cup \{1 \dots n\} \times \{\perp\}$.

To calculate the cost for a structure-sequence alignment, we use a subset of **edit operations** defined by Jiang *et al.* [48] and the subset of edition operations is shown in Figure 2.2 and is defined as follows. Suppose that i, i' are indices of S_Q and j, j' are indices of S_W.

1. base operations
   (a) Base match. $(i, j) \in A$ and $S_Q[i] = S_W[j]$.
   (b) Base mismatch. $(i, j) \in A$ and $S_Q[i] \neq S_W[j]$.
   (c) Base deletion. $(i, \perp) \in A$.
   (d) Base insertion. $(\perp, j) \in A$.

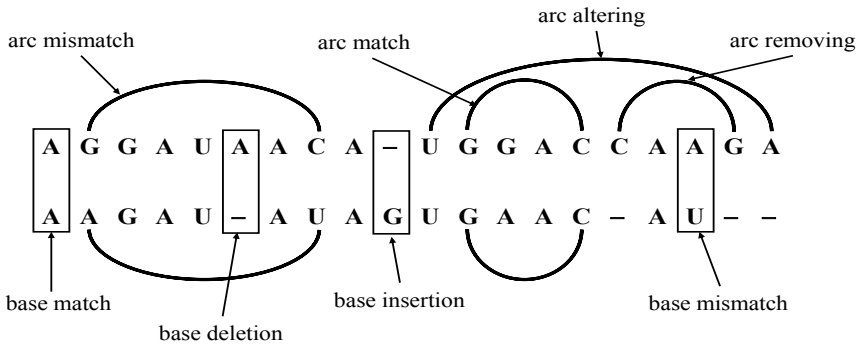


Figure 2.2: A sequence-structure alignment and the allowed edit operation.

2. arc operations
   (a) Arc match. Given $(i,j) \in A$, $(i',j') \in A$ and $(i,i') \in P_Q$, $S_Q[i] = S_W[j]$ and $S_Q[i'] = S_W[j']$.
   (b) Arc mismatch. Compared with arc match, at least one the equality $S_Q[i] = S_W[j]$ and $S_Q[i'] = S_W[j']$ is not satisfied.
   (c) Arc altering. Given $(i,i') \in P_Q$, one of $(i,\perp) \in A$ and $(i',\perp) \in A$ satisfies.
   (d) Arc removing. Given $(i,i') \in P_Q$, $(i,\perp) \in A$ and $(i',\perp) \in A$.

Here, We denote the gaps in Q with $G(Q) := \{(i,\ell) \mid \text{Q has a gap of length } \ell \text{ at position } i\}$. Likewise, we have the set $G(W)$. Therefore, the cost of an alignment $A \in \mathscr{A}'_{m,n}$ is defined as follows.

$$\begin{aligned} cost(A) = & \sum_{\substack{a \in A \\ \text{s.t.}\theta(a),\gamma(a) \neq \perp}} \sigma(a) + \sum_{a,b \in A \text{ s.t.}(\theta(a),\theta(b)) \in P_Q} \tau(a,b) \\ & + \sum_{(i,\ell) \in G(Q)} \lambda_Q(i,\ell) + \sum_{(i,\ell) \in G(W)} \lambda_W(i,\ell) \end{aligned} \tag{2.1}$$

where $\sigma : \Sigma^2 \to \mathbb{R}$ is the cost function of the base match and the base mismatch operations (base substitution operations), $\tau : \Sigma^4 \to \mathbb{R}$ is the cost function of the arc match and the arc mismatch operations (arc substitution operations), the arc removing operation and the arc altering operation. In section 3.1.2.2, we add certain restrictions on the scoring of arc altering and arc removing for the further computation. The most widely used gap penalty function is the **affine gap penalty**. By using affine gap penalty, $\lambda_Q(i,\ell)$ can be represented as $\lambda_Q(i,\ell) = \alpha_Q + (\ell - 1)\beta_Q$ where α_Q and β_Q are the **gap open** and **gap extension** penalties for the arc-annotated sequence

$A \times B \to C$	Complexity
PLAIN $\times$ PLAIN $\to$ PLAIN	mn
PLAIN $\times$ NESTED $\to$ NESTED	mn^3
PLAIN $\times$ CROSSING $\to$ CROSSING	Max SNP-hard
PLAIN $\times$ UNLIMITED $\to$ UNLIMITED	Max SNP-hard

Table 2.1: Complexities of the structure-sequence alignment. A, B, C are the types of the arc-annotated sequence

and $\lambda_W(i, \ell)$ can be represented as $\lambda_W(i, \ell) = \alpha_W + (\ell - 1)\beta_W$ where α_W and β_W are the **gap open** and **gap extension** penalties in the plain sequence.

Definition 2 (**structure-sequence alignment problem**)**.** Given an arc-annotated sequence $Q = (S_Q, P)$ and a plain sequence $W = (S_W, \emptyset)$. The structure-sequence alignment problem is to find the the alignment between Q and W with minimum cost.

The type of P in an arc-annotated sequence Q can be classified as PLAIN, NESTED, CROSSING, UNLIMITED as mentioned before. According to the alignment hierarchy defined in [8, 9], they have different complexities as shown in table 2.1. A, B and C are the types of the arc-annotated sequence and the type C determines the search space in the alignment. In our program `LiCoRNA`, we do not support UNLIMITED structures which involve non-Watson-Crick base pairs. In the following section, we will introduce the state-of-the-art algorithms to solve the structure-sequence alignment problem for PLAIN, NESTED, CROSSING structures.

2.1 Sequence-sequence alignment

If the arc-annotated sequence Q is PLAIN, Q can be represented as $(S_Q, \emptyset)$. The alignment between $Q = (S_Q, \emptyset)$ and $W = (S_W, \emptyset)$ can be solved by the Needleman-Wunsch algorithm [68] and the alignment with the affine gap penalty function can be solved by the Gotoh's algorithm [35].

For the Needleman-Wunsch algorithm, we create a matrix with size $m \times n$. Each cell in the matrix stores a value $F(i, j)$ where i and j are indices of the matrix. Here $F(i, j)$ is the cost of the optimal alignment between $S_Q(1 \dots i)$ and $S_W(1 \dots j)$. The

dynamic programming (DP) equation to calculate $F(i,j)$ is shown below:

$$F(i,j) = \min \begin{cases} F(i-1,j-1) + \sigma((i,j)) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases} \tag{2.2}$$

Here, $a = (i,j)$ is a match in the alignment and $\sigma((i,j))$ is the cost function of base substitution operation. The constant d stands for the gap penalty. The algorithm takes $\mathcal{O}(mn)$ time and $\mathcal{O}(mn)$ memory. Actually, you can rewrite the DP equation into equation 2.3. This kind of formula $F = \max\{MATCH,DELETE,INSERT\}$ gives a general idea for structure-sequence alignment.

$$\begin{aligned} F(i,j) &= \min\{MATCH, DELETE, INSERT\} \\ MATCH &= F(i-1,j-1) + \sigma((i,j)) \\ DELETE &= F(i-1,j) + d \\ INSERT &= F(i,j-1) + d \end{aligned} \tag{2.3}$$

Affine gap penalty function is more reasonable to model the phenomenon that if a particular position is gapped, the probability of the next position being gapped is higher. **Gotoh's algorithm** [35] allows the affine gap penalty function without increasing the time complexity. The affine gap penalty function takes the form $d + (L-1)e$ where d represents the gap opening penalty, e is the gap extension penalty and L is the length of the gap. Now instead of using one matrix F, we use three matrices F_M, F_x, F_y for each combination (i,j). Each combination (i,j) can be in three states in the alignment: F_M state that $S_Q(i)$ is aligned to $S_W(j)$. F_x state that $S_Q(i)$ is aligned to a gap and F_y state that $S_W(j)$ is aligned to a gap. If $-d-e$ is less than the lowest mismatch score, we do not need to consider the case where a deletion is followed directly by an insertion. This gives:

$$F_M(i,j) = \min \begin{cases} F_M(i-1,j-1) + \sigma((i,j)) \\ F_x(i-1,j-1) + \sigma((i,j)) \\ F_y(i-1,j-1) + \sigma((i,j)) \end{cases} \tag{2.4}$$

$$F_x(i,j) = \min \begin{cases} F_M(i-1,j) + \alpha_W \\ F_x(i-1,j) + \beta_W \end{cases}$$

$$F_y(i,j) = \min \begin{cases} F_M(i,j-1) + \alpha_Q \\ F_y(i,j-1) + \beta_Q \end{cases}$$

Here, α_Q and β_Q are the **gap open** and **gap extension** penalties for the arc-annotated sequence. The **gap open** and **gap extension** penalties in the plain sequence are denoted as α_W and β_W. The cost function of **base substitution operation** is represented as $\sigma((i,j))$ where $a = (i,j)$ is a match in the alignment. The time complexity of Gotoh's algorithm is still $\mathcal{O}(mn)$.

2.2 NESTED structure-sequence alignment

If the arc-annotated sequence Q is NESTED, the alignment between $Q = (S_Q, P)$ and $W = (S_W, \emptyset)$ becomes more complicated. Jiang *et al.* [48] solved the problem using dynamic programming algorithm with time complexity $\mathcal{O}(mn^3)$.

They use the **edit distance** which is the cost of series of edit operations to transform one arc-annotated sequence into the other to measure the similarity between the two arc-annotated sequences. The **Edit problem** is to compute the **optimal** series of edit operations with minimum cost. The Edit problem between a NESTED structure and a PLAIN sequence is equivalent to the alignment problem [9, 8]. Therefore, the edit distance can be defined via the alignment. In the dynamic programming algorithm, instead of using $F(i,j)$ to represent the optimal subsequence-subsequence alignment, we use $F(i,i';j,j')$ to consider the alignment for two bases in S_Q because of the existing of the base pairs. Here $F(i,i';j,j')$ represents the edit distance between partial sequences $(S_Q[i,i'], P[i,i'])$ and $(S_W[j,j'], \emptyset)$ with $1 \leq j \leq j' \leq n$, and $P[i,i']$ denotes a subset of base pairs $\{(k,k')|i \leq k < k' \leq i', (k,k') \in P\}$.

However, not all combinations (i,i') of S_Q in $F(i,i';j,j')$ will be considered. The binarizing algorithm by Bafna *et al.* [3] can directly generate the set of valid combinations (i,i') and build a binary tree at the same time. One example is illustrated in Figure 2.3. The left part of the figure is a NESTED structure, the solid edges correspond to a set of combinations (i,i') where $(i,i') \in P$ and the dashed edges correspond to a set of added combinations (i,i') generated by the algorithm and we denote this set as P'. The total combinations $P \cup P'$ are shown in the right part of Figure 2.3. The dynamic programming is designed to compute the optimal alignment between Q and W. As described by Jiang [48], the DP calculates $F(i,i';j,j')$ in the ascending order of $i' - i$ which exactly is the same as the order that you traverse the binary tree from bottom to top.

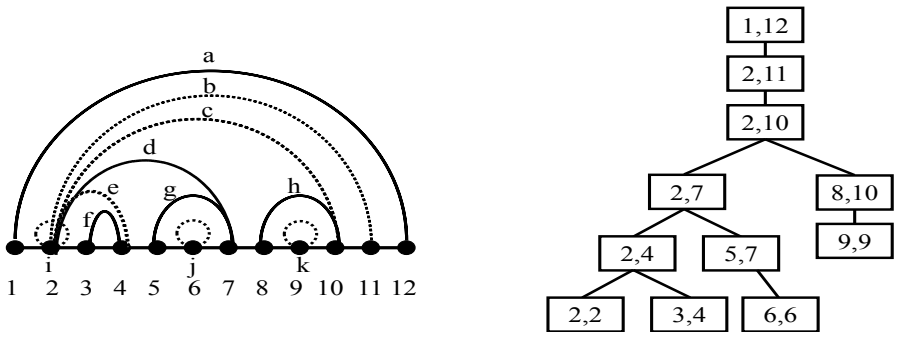


Figure 2.3: Example of binarizing an RNA NESTED structure and a binary tree is built at the same time.

To compute $F(i, i'; j, j')$, if $(i, i') \in P$, DP equation 2.5 is used and if $(i, i') \in P'$, DP equations 2.6 and 2.7 are used. Now let us consider the combinations $(i, i') \in P$ and there are six cases to be considered. We follow the general formula scheme as shown in the sequence-sequence alignment.

$$
\begin{aligned}
F(i, i'; j, j') &= \min\{MATCH, DELETE, INSERT\} \\
MATCH &= F(i+1, i'-1; j+1, j'-1) + \tau((i, j), (i', j')) \\
DELETE &= \min \begin{cases} F(i+1, i'-1; j, j'-1) + \tau((i, \perp), (i', j')) \\ F(i+1, i'-1; j+1, j') + \tau((i, j), (i', \perp)) \\ F(i+1, i'-1; j, j') + \tau((i, \perp), (i', \perp)) \end{cases} \\
INSERT &= \min \begin{cases} F(i, i'; j, j'-1) + d \\ F(i, i'; j+1, j') + d \end{cases}
\end{aligned} \tag{2.5}
$$

Here, τ is the cost function for **arc substitution operations** (arc match and arc mismatch operations), **arc removing operation** and **arc altering operation**. The constant d stands for the gap penalty.

Now let us consider the combinations $(i, i') \in P'$ where $S_Q[i]$ and $S_Q[i']$ do not form base pairs in the secondary structure. There are two different subcases depending on whether $S_Q[i']$ forms a base pair with another base except $S_Q[i]$ or not. The two different subcases are shown in the binary tree in Figure 2.3. For subcase 1, like the node $(2, 11)$ and base 11 does not form base pair with other base in the secondary structure. For subcase 2, like node $(2, 10)$, base 10 forms base pair with base 8 in the secondary structure. For subcase one, we only need to consider the alignment of base

i' and the DP equation is

$$\begin{aligned} F(i,i';j,j') &= \min\{MATCH, DELETE, INSERT\} \\ MATCH &= F(i,i'-1;j,j'-1) + \sigma((i',j')) \\ DELETE &= F(i,i'-1;j,j') + d \\ INSERT &= F(i,i';j,j'-1) + d \end{aligned} \tag{2.6}$$

where $\sigma((i,j))$ is the cost function of **base substitution operations**. The constant d stands for the gap penalty.

In subcase two, we use $u(i)$ to denote the base pair in P_Q incident on position i and $u(i)_l$, $u(i)_r$ are the left and right endpoint of the base pair. Therefore, we have

$$F(i,i',;j,j') = \min_{j \le j'' \le j'} F(i,u(i')_l - 1;j,j''-1) + F(u(i')_l,i';j'',j') \tag{2.7}$$

where $i' = u(i')_r$. Finally, the entry $F(1,m;1,n)$ corresponds to the edit distance between Q and W and the standard backtracking technique is used to get the optimal alignment. The time complexity for the algorithm is $\mathcal{O}(mn^3)$.

2.3 CROSSING structure-sequence alignment

Now we consider the structure-sequence alignment when the arc-annotated sequence Q is CROSSING. Thus the structure of Q contains pseudoknots. It has been stated that H-type and kissing-hairpin pseudoknots account for 80% of the pseudoknots in the known structures [78]. Han's algorithm [39] can deal with standard pseudoknots(contains H-type pseudoknot and kissing-hairpin). Later, Wong *et al.* [117] design an algorithm to deal with non-standard pseudoknots and recursive pseudoknots. Unlike previous algorithms, Matsui *et al.* [60] use tree adjoining grammars (TAGs) to model pseudoknots and based on that, they develop a dynamic programming algorithm to obtain the optimal structural alignment. Recently, Rinaudo *et al.* give a general setting for structure-sequence comparison for a large class of RNA structures that unifies all the pseudoknotted structures [78]. In this section, I briefly introduce Han's algorithm, Matsui's algorithm and Rinaudo's algorithm. The reason that I only introduce those algorithm in this section is that the corresponding softwares are available on the Internet.

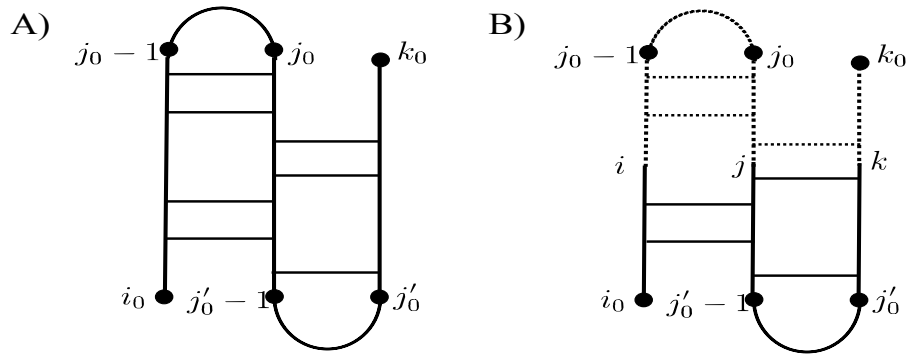


Figure 2.4: A) A simple pseudoknot. B) Subpseudoknot structure ends with frontier (i, j, k).

2.3.1 Han's algorithm

Han's algorithm is based on the algorithm of NESTED structure-sequence alignment between $Q = (S_Q, P)$ and $W = (S_W, \emptyset)$ and they develop a program named `PAL` which handles simple pseudoknots. Here I first show the definition of simple pseudoknot and later I will explain the chaining algorithm which is used to decompose the simple pseudoknot. $P_{i_0,k_0} = \{(i,k) \in P | i_0 \leq i \leq k \leq k_0\}$ contains the positions of base pairs where the two ends of the base pair are within the range $[i_0, k_0]$. An RNA secondary structure P_{i_0,k_0} is **regular** if and only if $P_{i_0,k_0} = \emptyset$ or P_{i_0,k_0} is a NESTED structure.

Definition 3 (**simple-pseudoknot** [39])**.** P_{i_0,k_0} is a simple pseudoknot if and only if P_{i_0,k_0} is regular or $\exists j_0, j'_0$ with $i_0 \leq j_0 \leq j'_0 \leq k_0$ such that the resulting partition, $D_1 = [i_0, j_0 - 1]$, $D_2 = [j_0, j'_0 - 1]$, $D_3 = [j'_0, k_0]$, satisfies the following:

- $P_{i_0,k_0} = (S_L \cup S_R)$, where $S_L = \{(i,j) \in P_{i_0,j_0} | i \in D_1, j \in D_2\}$ and $S_R = \{(i,j) \in P_{i_0,j_0} | i \in D_2, j \in D_3\}$.
- S_L and S_R are regular.

Figure 2.4 A) illustrates an example of a simple pseudoknot. **The subpseudoknot** $Q(i,j,k)$ is defined as the union of two partitions $[i_0, i] \cup [j, k]$. The **frontier** of the subpseudoknot $Q(i,j,k)$ is a triple (i,j,k) as shown in Figure 2.4 B). Unlike the previous algorithms, Han's algorithm aims to align the frontier (i,j,k) to (i',j',k') of W which means that it considers the alignment of three bases at the same time.

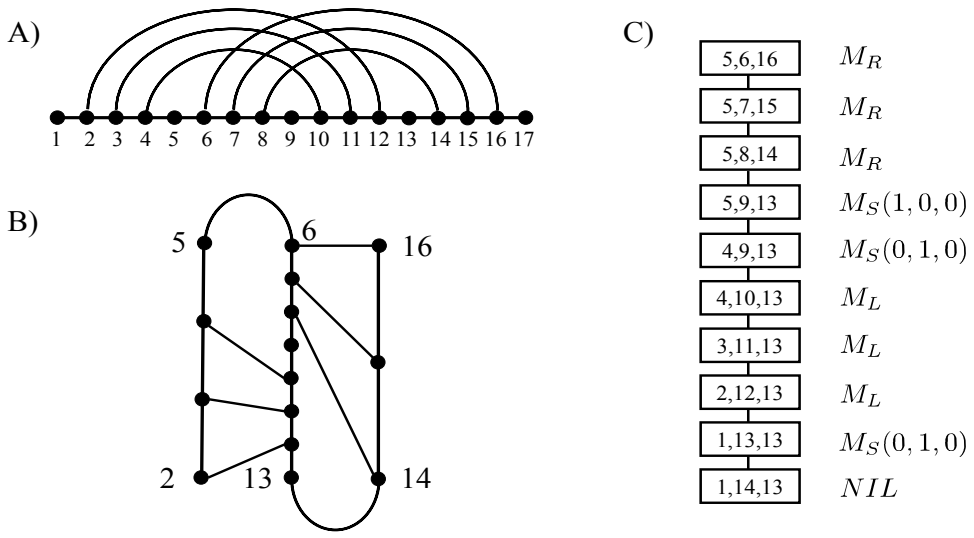


Figure 2.5: An example for chaining algorithm to decompose simple pseudoknot. A) A simple pseudoknot. B) Another representation of the simple pseudoknot. C) The output of the chaining algorithm.

The **chaining algorithm** is used to decompose the simple pseudoknot and the process is shown in Figure 2.5. The input for the chaining algorithm is $P_{i_0=2,k_0=16}$. According to the definition of simple pseudoknot, the structure $P_{i_0=2,k_0=16}$ can be divided into three partitions: $D_1 = [2, 5]$, $D_2 = [6, 13]$ and $D_3 = [14, 16]$ where $j_0 = 6$ and $j_0' = 14$. Figure 2.5 B) is another representation of the simple pseudoknot. The chaining algorithm begins with the root frontier $(j_0 - 1, j_0, k_0)$ which is $(5, 6, 16)$ in Figure 2.4. Actually, the subpseudoknot $Q(j_0 - 1, j_0, k_0) = [i_0, j_0 - 1] \cup [j_0, k_0]$ is the entire simple pseudoknot. The output of the chaining algorithm is a directed path of **nodes** as shown in Figure 2.5 C). The node represents a subpseudoknot with frontier (i, j, k).

The nodes are classified into 3 kinds: M_R, M_L, M_S. $M_L(M_R)$ is a kind of node representing the subpseudoknot $Q(i, j, k)$ where $(i, j) \in S_L((j, k) \in S_R)$ while M_S is another kind of node representing the subpseudoknot $Q(i, j, k)$ where $(i, j) \notin S_L$ and $(j, k) \notin S_R$. $(1, 0, 0)$ means that the chaining algorithm moves from the current node with frontier (i, j, k) to the child node with frontier (i', j', k') by subtracting 1 from i to get i' which means that $i' = i - 1$, $j' = j$ and $k' = k$. For different kinds of nodes, there are different DP recursions. The recursions for M_R and M_L are almost the same and the recursions for $M_S(1, 0, 0)$, $M_S(0, 1, 0)$ and $M_S(0, 0, 1)$ are almost the same, so

I only show the recursions for M_L and $M_S(1,0,0)$. We also follow the general formula and the difference is that we need to consider the alignment of three bases at the same time. For the node which belongs to M_L which means that $(i,j) \in P$ where P is a set of positions of base pairs. The cost of the alignment for $(i,j) \in P$ should be considered at the same time and the edit operations are the same with NESTED structure-sequence alignment.

$$
\begin{aligned}
F(i,j,k;i',j',k') &= \min\{MATCH, DELETE, INSERT\} \\
MATCH &= F(i-1,j+1,k;i'-1,j'+1,k') + \tau((i,i'),(j,j')) \\
DELETE &= \min \begin{cases} F(i-1,j+1,k;i',j'+1,k') + \tau((i,\perp),(j,j')) \\ F(i-1,j+1,k;i'-1,j',k') + \tau((i,i'),(j,\perp)) \\ F(i-1,j+1,k;i',j',k') + \tau((i,\perp),(j,\perp)) \end{cases} \\
INSERT &= \min \begin{cases} F(i,j,k;i'-1,j',k') + d \\ F(i,j,k;i',j'+1,k') + d \\ F(i,j,k;i',j',k'-1) + d \end{cases}
\end{aligned} \tag{2.8}
$$

Here, τ is the cost function for **base pair substitution operations** (arc match and arc-mismatch operations), **arc-removing operation** and **arc-altering operation**. The constant d stands for the gap penalty.

For the node $M_S(1,0,0)$ we consider the alignment of i in the frontier (i,j,k). The DP equation in this case is shown below.

$$
\begin{aligned}
F(i,j,k;i',j',k') &= \min\{MATCH, DELETE, INSERT\} \\
MATCH &= F(i-1,j,k;i'-1,j',k') + \sigma((i,i')) \\
DELETE &= F(i-1,j,k;i',j',k') + d \\
INSERT &= \min \begin{cases} F(i,j,k;i'-1,j',k') + d \\ F(i,j,k;i',j'+1,k') + d \\ F(i,j,k;i',j',k'-1) + d \end{cases}
\end{aligned} \tag{2.9}
$$

where $\sigma((i,j))$ is the cost function of **base substitution operation**. The constant d stands for the gap penalty.

Recursions 2.8 and 2.9 take totally $\mathcal{O}(n^3)$ time, separately. Each time, we generate a new node using chaining algorithm, the frontier (i,j,k) decreases by at least 1. Therefore, the number of nodes in the chain is $\mathcal{O}(m)$. Thus, the time complexity for the algorithm is $\mathcal{O}(mn^3)$.

2.3.2 Matsui's algorithm

Matsui *et al.* [60] used **tree adjoining grammars** (TAGs) to represent pseudoknots and developed a DP algorithm to calculate the optimal structural alignment. This work is based on the pair hidden Markov models on tree structures (PHMMTSs) proposed by Sakakibara [83]. However, PHMMTSs can only handle structural alignment for NESTED structures. By incorporating the TAGs into PHMMTSs, Matsui *et al.* proposed pair stochastic TAGs (PSTAGs) for pseudoknot structural alignment. Although TAGs can still not model all kinds of pseudoknots, it gives another view for structure-sequence alignment by using grammars. Another widely used grammar for structure-sequence alignment is context-free grammar which is used to represent secondary structures and the main corresponding available software is `INFERNAL` [67].

In the following, we begin with the definition of TAGs, along with two subclasses, simple linear TAGs (SL-TAGs) and extended simple linear TAGs (ESL-TAGs) and then we introduce the particular ESL-TAGs for pseudoknots. The basic operation for TAGs is the **adjoining operation** which is based on the **active** node. Finally, based on the PSTAG, we introduce the DP algorithm.

A TAG is a 5-tuple grammar $G = (V_N, V_T, S, \mathcal{I}, \mathcal{A})$. V_N is a set of non-terminal symbols. V_T is a set of terminal symbols and $S \in V_T$ is the initial symbol. $\mathcal{I}$ is a finite set of initial trees and $\mathcal{A}$ is a finite set of adjunct trees. For $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{I}$, the following must be satisfied:

- $\alpha(1) = S$ and $\mathcal{Y}(\alpha) \in V_T^*$.
- $\beta(1) = X \in V_N$ and $\mathcal{Y}(\beta) \in V_T^* X V_T^*$.

Here V_T^* is the set of finite sequences over V_T. For a tree γ, $\gamma(p) = A$ means that the label of the node p is $A \in V_N \cup V_T$. Furthermore, the root node in γ is numbered 1 and the other nodes are numbered in preorder. The yield of γ, denoted as $\mathcal{Y}(\gamma)$, is the string of the labels of the leaf nodes of γ. For $\beta \in \mathcal{A}$, the foot node of β is the node that is labeled with X in $\mathcal{Y}(\beta)$ and backbone of β is the path from root node to foot node.

The nodes in the tree γ can be **active** or **inactive**. The active nodes are indicated by $*$ and to understand the idea of active nodes, one must understand the **adjoining operation** first. Figure 2.6 illustrates the adjoining operation: γ is a tree with

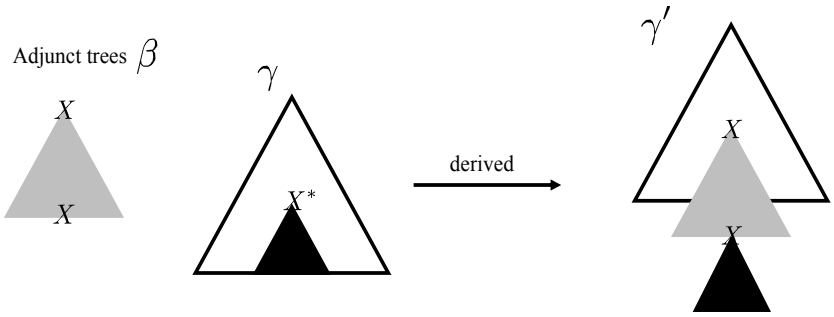


Figure 2.6: An adjoining operation in TAGs. γ is tree with $\gamma(p) = X$ and β an adjunct tree with $\beta(1) = X$. γ' is a derived tree from γ by adjoining β into γ at position p.

$\gamma(p) = X$ and β is an adjunct tree with $\beta(1) = X$. γ' is a derived tree from γ by adjoining β into γ at position p. Therefore, a node at p in γ with label $X \in V_N$ is active if there is an adjoining tree $\beta \in \mathcal{A}$ that can adjoin γ at position p.

A TAG G is an **SL-TAG** if $\forall\alpha \in \mathcal{I}$, α is simple linear which means that there is only one active node in α and $\forall\beta \in \mathcal{A}$, β is simple linear which means that there is only one active node on the backbone of β. A TAG G is an **ESL-TAG** if $\forall\alpha \in \mathcal{I}$, α is simple linear and $\forall\beta \in \mathcal{A}$, β is semi-simple linear meaning that there are two active nodes in β, one is on the backbone and the other is elsewhere.

Uemura *et al.* [100] used special ESL-TAGs to represent RNAs with pseudoknots. $G_{RNA} = (V_N, V_T, S, \mathcal{I}, \mathcal{A})$. $V_T = \{A, C, G, U\}$ and $V_N = S$. The bar notation is used to represent the Watson-Crick base pairings, such as $\overline{A} = U$ and $\overline{C} = G$. $\mathcal{I}$ and $\mathcal{A}$ are shown in Figure 2.7. TYPE 2 and TYPE 3 are to emit base pairs. TYPE 4 is to emit single base. TYPE 5 is used to generate branching structures. Furthermore, TYPE 2, TYPE 3 and TYPE 4 are simple linear and TYPE 5 is semi-simple linear.

In a previous paper [83], Sakakibara *et al.* proposed PHMMTSs to emit a pairwise alignment of trees and here they extended the idea to PSTAGs to emit the alignment of TAGs trees. The TAG tree represents the derivation process of an RNA pseudoknotted structure as shown in Figure 2.8 A). The PSTAG has three states, matched state M, insertion state I and deletion state D which is suitable for affine penalty alignment as shown in Figure 2.8 B). τ_X is the initial probability. δ_{XY} denotes the transition probability from state X to state Y. $P_O^X(\alpha, \beta)$ denotes the emission probability of adjunct trees α, β at state $X \in \{M, I, D\}$.

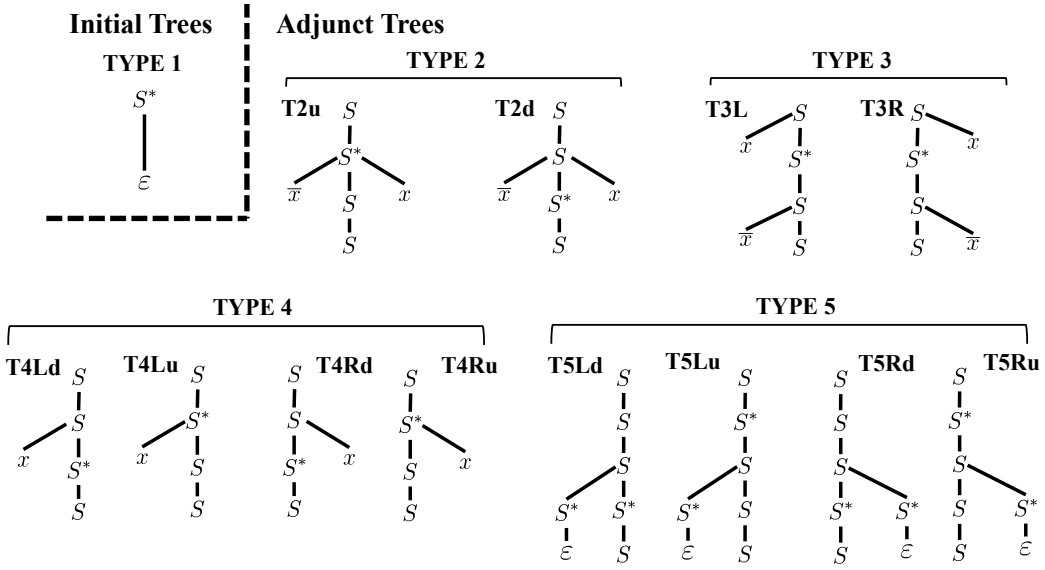


Figure 2.7: Inital trees and adjunct trees for TAG G_{RNA}. S^* is the active nodes and the other nodes are inactive. $\overline{x}$ and x forms base pairs.

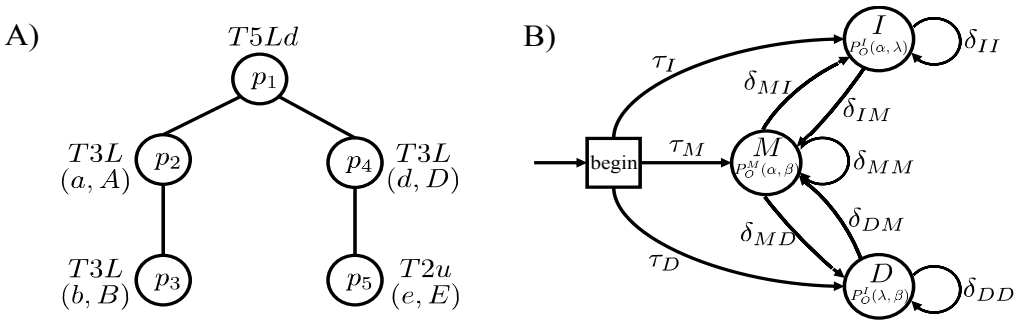


Figure 2.8: (A) A TAG tree for '(a(bB)A)(d[eD)E]'. (B) A state transition diagram of PSTAG for affine-gap alignment on tree structures. τ_X is the initial probability. δ_{XY} denotes the transition probability from state X to state Y. $P_O^X(\alpha, \beta)$ denotes the emission probability of adjunct trees α, β at state $X \in \{M, I, D\}$.

Therefore, given an arc-annotated sequence $Q = (S_Q, P)$ of length m and a plain sequence $W = (S_W, \emptyset)$ of length n with $S_W \in V_T^*$, a TAG tree T is obtained by parsing Q using the G_{RNA}. Let $w[i, j, k, l]$ be the pair of subsequences $S_W[i+1] \dots S_W[j]$ and $S_W[k+1] \dots S_W[l]$. The size of T is denoted as L with l_1 TYPE 5 nodes and totally l_2 TYPE 2 to 4 nodes ($L = l_1 + l_2$). Let $T[p]$ with $1 \leq p \leq L$ be the subtree of T rooted at p and $v[p]$ be the label (an adjunct tree) for the position p. If p is of arity 2, the children of p are p_1 and p_2 and if p is of arity 1, the child of p is denoted as p_1. Here I do not show the DP equation in detail, but I will show some ideas about it.

Now we will consider the general recursion $P^Z(w[i, j, k, l], T[p])$ which means the probability of aligning $w[i, j, k, l]$ to subtree $T[p]$ with state Z. For simplicity's sake, we only consider the case for matched state and the probability is $P^M(w[i, j, k, l], T[p])$. Theoretically, for fixed i, j, k, l where $(0 \leq i \leq j \leq k \leq l \leq n)$, $W[i, j, k, l]$ can form all the types of the adjunct trees in G_{RNA} and the type is denoted as $v_c[q]$. Furthermore, the corresponding constructed subtree is denoted as $T_c[q]$. So we are trying to compute $P(T_c[q], T[p])$, $\forall p \in \{1 \dots L\}$. However, not all the pair of the adjunct trees can be matched in aligning two TAG trees as stated in [60]. The general idea is that the nodes of arity 1(or 2) for $T_c[q]$ can only match the nodes of arity 1(or 2) for $T[p]$. Therefore, if the type $v_c[q]$ is of arity 1, we only consider the $v[p]$, $\forall p \in \{1 \dots L\}$ where $v[p]$ belongs to the TYPE $2-4$. The equation for $P^M(w[i, j, k, l], T[p])$ is

$$\begin{aligned} P_1^M(w[i, j, k, l], T[p]) = & \max_{\substack{X \in \{M, I, D\} \\ \beta \in \mathcal{T}}} P_O^M(\beta, v(p)) \cdot \delta_{XM} \\ & \cdot P^X(i + |LU(\beta)|, j - |LD(\beta)|, k + |RD(\beta)|, l - |RU(\beta)|, T[p_1]) \end{aligned} \tag{2.10}$$

Here, $\mathcal{T}$ is a set containing all the TYPE $2-4$ adjunct trees. Let β be a simple linear adjunct tree and the active node of β is labeled with Y^*. The yield of β can be decomposed into four subsequences $|LU(\beta)|$, $|LD(\beta)|$, $|RD(\beta)|$ and $|RU(\beta)|$. Figure 2.9 explains the process of the recursion equation 2.10. The time complexity is $\mathcal{O}(l_1 n^4)$ and space complexity is $\mathcal{O}(Ln^4)$. If the type $v_c[q]$ is of arity 2, $v_c[q]$ can be $T5Ld$, $T5Lu$, $T5Rd$ and $T5Ru$. Here we only illustrate the recursion for $T5Ld$.

$$\begin{aligned} P_2^M(w[i, j, k, l], T[p]) = & \max_{\substack{X, Y \in \{M, I, D\} \\ i < r \leq j \\ i \leq s \leq r}} P_O^M(T5Ld, v(p)) \\ & \cdot \delta_{XM} \cdot P^X(w[r, j, k, l], T[q_1]) \\ & \cdot \delta_{YM} \cdot P^X(w[i, s, s, r], T[q_2]) \end{aligned} \tag{2.11}$$

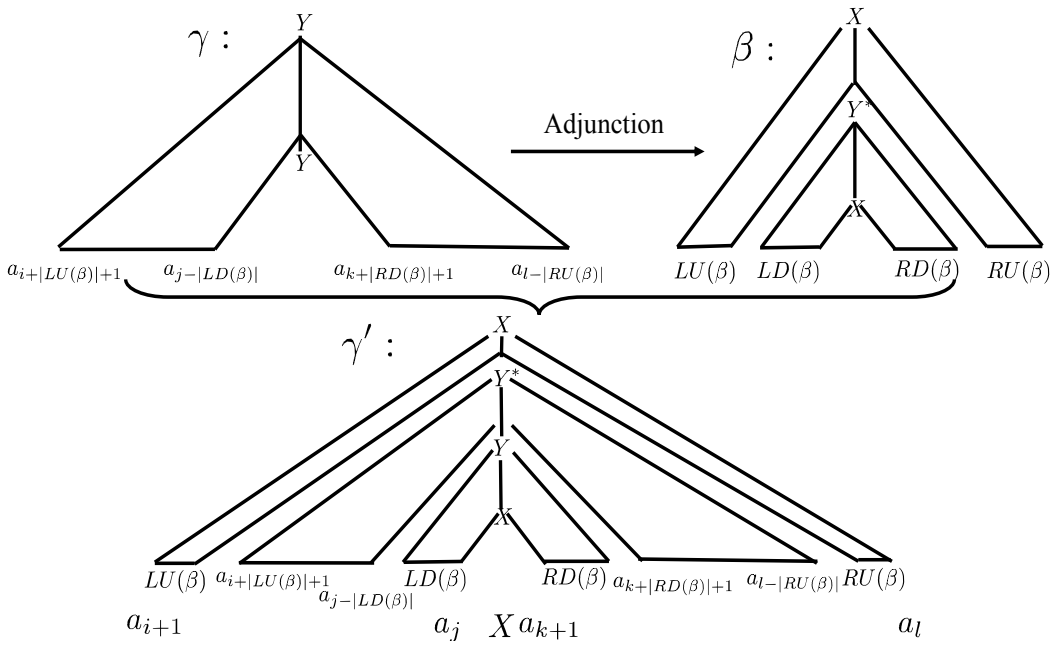


Figure 2.9: illustration of equation 2.10. γ is a tree with $\gamma(0) = Y$ and $\beta \in \mathcal{A}$ is an adjunct tree. γ' is a derived tree by adjoining γ into β.

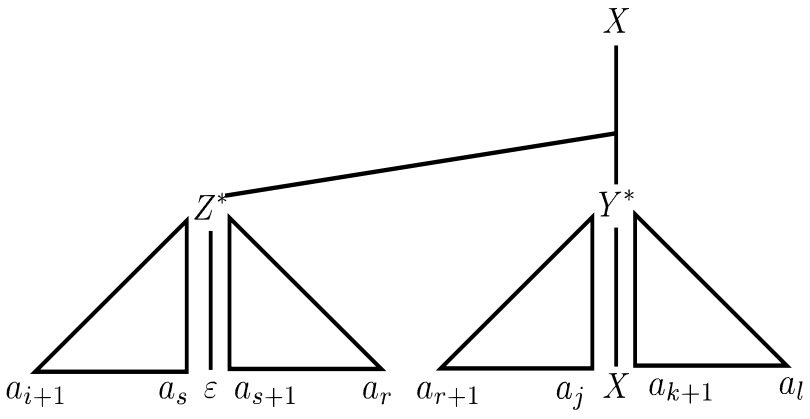


Figure 2.10: illustration of equation 2.11 if $v_c[q] = T5Ld$.

The recursion 2.11 can be easily understood from Figure 2.10. The time complexity is $\mathcal{O}(l_2 n^6)$ from the first view of the recursion 2.10. However, the complexity can be reduced to $\mathcal{O}(l_2 n^5)$ by pre-computing the term $P^X(w[i,s,s,r], T[q_2])$ [100].

2.3.3 Song's algorithm

Song *et al.* [91] illustrated an efficient parameterized algorithm for RNA pseudo-knotted structure-sequence alignment and also for searching of RNA structures in genomes. I will explain the method to calculate an optimal structure-sequence alignment from three steps:

- Construct the conformation graph H for consensus structure of an RNA family and image graph G for the target sequence.
- Formulate RNA structure-sequence alignment between the structure profile and the target sequence as a generalized subgraph isomorphism problem.
- Solve the problem using a tree decomposition based parameterized algorithm.

The basic structural units in the consensus secondary structure for an RNA family are the stems and the loops. Then a **conformation graph** H is introduced to describe the relationship among the stems and the loops. H is a mixed graph containing both undirected and directed edges between the vertices. Each vertex represents a stretch of nucleotides which forms one half of a stem. Two vertices are connected by an undirected edge if the corresponding stretches of nucleotides form a stem ($5'$ to $3'$) in the structure. Two vertices are connected by a directed edge if the corresponding stretches of nucleotides are the two ends of a loop in the structure. Figure 2.11 A) shows a simple consensus secondary structure with two parallel stems. Figure 2.11 B) illustrates the conformation graph H for the secondary structure. Technically, two additional vertices s (source) and t (sink) are added to the graph.

For the target sequence, an **image graph** G is built. The key step is to use the profile of each stem to scan the target sequence based on the covariance models. Then the alignments with the significant score between the profile stem and the base pairing regions in the target sequence are found. Finally, several base pairing regions are abstracted for one stem and each base pairing region is called an **image** of the stem. Song *et al.* defined a parameter k to be the maximum number of the base pairing

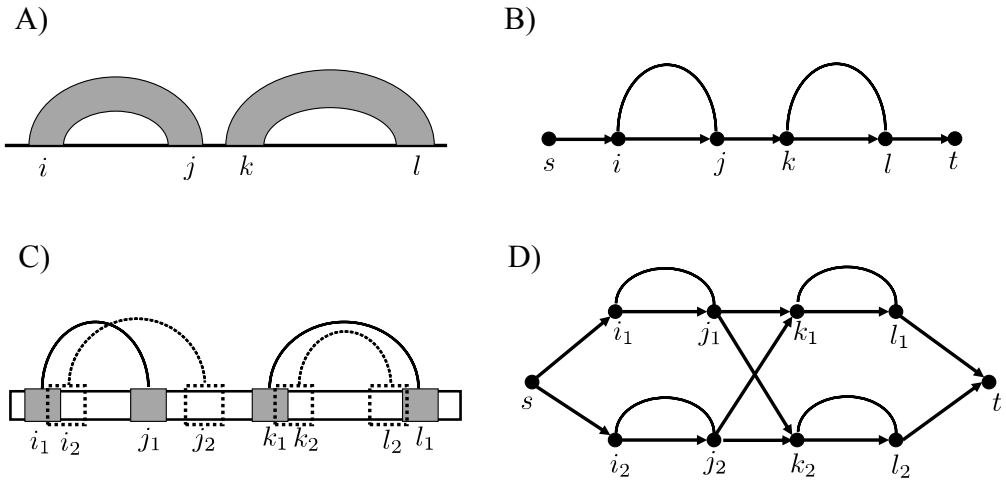


Figure 2.11: [91]. A) A consensus secondary structure for an RNA family. It has two parallel stems (i, j), (k, l). B) Conformation graph H of the RNA structure in A). C) target sequence with two images for each stem, represented by two pairs of the rectangles (one pair is filled with grey and one pair is with dotted border). The two images for the stem (i, j) are (i_1, j_1), (i_2, j_2) and the two images for the stem (k, l) are (k_1, l_1) and (k_2, l_2). D) image graph G for the target sequence.

regions for each stem. Each one of the base pairing regions is defined as a vertex in the image graph G. The image graph is also a mixed graph. Two vertices are connected by an undirected edge if the corresponding bases regions are pairing. Two vertices are connected by a directed edge if the corresponding bases regions are non-overlapping and are not images of the same stem. Figure 2.11 C) shows a target sequence. Each stem in the structure profile has two images. For the stem (i, j), the two images are (i_1, j_1), (i_2, j_2). For the stem (k, l), the two images are (k_1, l_1) and (k_2, l_2).

Given the conformation graph H for the structure profile and image graph G for the target sequence, the structure-sequence alignment can be formulated as a **generalized subgraph isomorphism problem** [91] which is to find a one-to-one mapping f from vertices in H to their images in a subgraph S of G such that

- (u, v) is an edge in H if and only if $(f(u), f(v))$ is an edge in S
- For any set of vertices in G representing overlapping regions on the target sequence, at most one of them can be selected to the subgraph S
- The total score achieves the maximum when calculating the sum of the alignment score of the stems and loops according to the mapping f.

To solve the problem, Song *et al.* developed a tree decomposition based DP algorithm to compute the optimal alignment between H and the subgraph of G. As the Song's algorithm and the Rinaudo's algorithm are similar, the detail of the algorithm will be shown in the section of the Rinaudo's algorithm. The complexity of the Song's algorithm is $\mathcal{O}(k^t n)$ where k is the maximum number of the base pairing regions for each stem, t is the tree-width of the tree decomposition of the conformation graph H and n is the number of nodes(bags) in the tree decomposition. However, this algorithm needs a heuristics preprocessing step to detect the potential k candidates for each stem which may cause inaccuracy of the alignment.

2.3.4 Rinaudo's algorithm

Recently, Rinaudo *et al.* [78] developed a fully general method for structure-sequence comparison, which is able to take as input any type of pseudoknotted structure. The inputs for the algorithm are an arc-annotated sequence $Q = (S_Q, P)$ of length m and a plain sequence $W = (S_W, \emptyset)$ of length n. The arc-annotated sequence can be referred to as the RNA graph. The algorithm also relies on the tree decomposition of the RNA graph and further dynamic programming algorithm is used to calculate an optimal alignment. Different from Song's algorithm, in Rinaudo's algorithm, each nucleotide in the query structure and target sequence is treated as a vertex in the RNA graph. The tree decomposition can be defined as follows:

Definition 4 (**Tree Decomposition of an arc-annotated sequence**)**.** A tree decomposition of an arc-annotated sequence $Q = (S_Q, P)$ is a rooted tree T whose nodes, called bags, are sets of positions in Q. Additionally, the set $\mathcal{X}$ of bags must satisfy:

- (Node coverage) $\forall s \in S, \exists X \in \mathcal{X}$ such that $s \in X$;
- (Edge coverage) $\forall (i,j) \in P, \exists X \in \mathcal{X}$ such that $\{i,j\} \subset X$. Moreover, $\forall 1 \leq i < m, \exists X \in \mathcal{X}$ such that $\{i, i+1\} \subset X$;
- (Coherence) $\forall X, X', X'' \subset \mathcal{X}$, if X'' lies on the path from X and X' on T, then $X \cap X' \subset X''$.

Figure 2.12 illustrates a possible tree decomposition for a simple pseudoknot. The **width** of a tree decomposition T is defined as $\max_{X \in \mathcal{X}} |X| - 1$. The **tree-width** tw

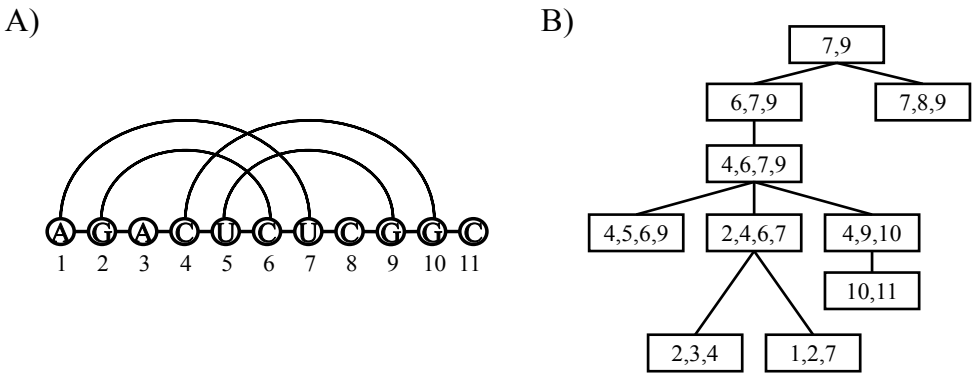


Figure 2.12: A) Arc-annotated sequence of a simple pseudoknot. B) A corresponding tree decomposition and the width of this tree decomposition is 3.

of an arc-annotated sequence Q is the minimum width over all possible tree decompositions of Q. Suppose the current bag is X, we use $p(X)$ to represent the **father** bag and $child(X)$ as its **children** bags. The **root** bag is denoted as X_0. Furthermore, we split a bag X into two parts **transition** indices $X\uparrow$ and **proper** indices $X\downarrow$.

$$X\uparrow := \{i \in X \mid i \in p(X)\}.$$
$$X\downarrow := \{i \in X \mid i \notin p(X)\}.$$

As shown in Figure 2.13 A), if the current bag X contains positions $4, 6, 7, 9$, $X\uparrow = \{6, 7, 9\}$ and $X\downarrow = \{4\}$. Furthermore, $P(X)$ is the set of **proper base pairs** of a bag involving at least one proper index. Similarly, $N(X)$ represents the set of **proper backbone interactions**, i.e. consecutive positions involving proper indices.

$$P(X) := \{(i, j) \mid i, j \in X \text{ and } i \text{ or } j \in X\downarrow\}$$
$$N(X) := \{(i, i+1) \mid i, i+1 \in X \text{ and } i \text{ or } i+1 \in X\downarrow\}$$

The notion of smooth bag is further defined by Rinaudo *et al.* for assigning consecutive positions.

Definition 5 (Smooth Bag of a Tree Decomposition [78]). Let $X \in \mathcal{X}$ be a bag in a tree decomposition for an arc-annotated sequence $Q = (S_Q, P)$. If $X \neq X_0$, $p(X)$ is the father bag. X is **smooth** iff there exist two consecutive positions i and j

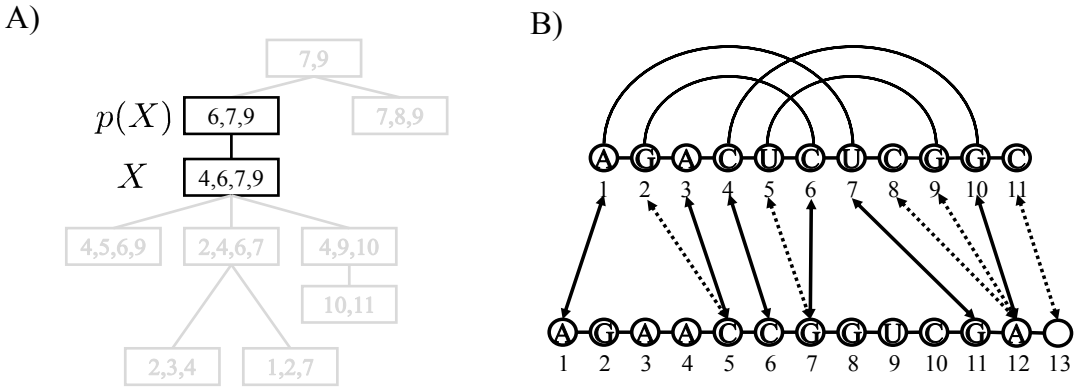


Figure 2.13: A) Illustration of the transition indices $X\uparrow$ and proper indices $X\downarrow$ of a bag. Suppose the current bag $X = \{4, 6, 7, 9\}$, $X\uparrow = \{6, 7, 9\}$ and $X\downarrow = \{4\}$. B) Representation of structure-sequence alignment. Every position in Q has a corresponding alignment triple. Operation δ distinguishes match positions and unmatched position in Q. We aggregate consecutive unmatched positions in Q to their nearest rightward matched position and a *virtual position* is added at the end of W. The alignment triple for positions $8, 10, 11$ in Q are $(8, 12, 0)$, $(10, 12, 1)$ and $(11, 13, 0)$.

such that position i is in the proper indices and position j is in the transition indices and j is not in one the children bags of X. Besides there is no i' in the proper indices 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 $(i, j) \in P$, or (2) iff the size of the root is strictly smaller than the size of one of its children.

A tree decomposition for an arc-annotated sequence $Q = (S_Q, P)$ is **smooth** if every bag $X \in \mathcal{X}$ is smooth. In this way, the Gotoh's algorithm can be used to calculate the optimal score for assignments of consecutive positions in $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$ when the assignments of the other indices in the bag are fixed.

Based on Definition 1 for structure-sequence alignment, to preserve the sequential order and make sure every position in Q has a mapping position in W, we extend the set $\mathscr{A}'_{m,n}$ to $\hat{\mathscr{A}}_{m,n} \coloneqq \{1 \dots m\} \times \{1 \dots n+1\} \times \{0, 1\}$ after adding a *virtual position* at the end of W. Additionally to θ and γ, a new operation δ is defined to get the third element of $\hat{\mathscr{A}}_{m,n}$ and δ distinguishes matched positions and unmatched positions. For a particular alignment triple $\forall a = (i, j, k) \in \hat{\mathscr{A}}_{m,n}, \delta(a) = 0$ means that position i aligns to j and position i is an unmatched position. Additionally, unmatched positions

are forced to aggregate to their right nearest positions. Moreover, $\delta(a) = 1$ means that sequence position i aligns to position j and the position i is a match position [78]. The details are shown in Figure 2.13 B).

For a particular bag X, let $\mathcal{A}_{X\uparrow}$ be all possible alignments for $X\uparrow$ and let $\mathcal{A}_{X\downarrow}$ be all possible alignments for $X\downarrow$.

$$\mathcal{A}_{X\uparrow} \coloneqq \bigcup_{i \in X\uparrow} \Big(\{i\} \times \{1 \dots n+1\} \times \{0,1\} \Big)$$
$$\mathcal{A}_{X\downarrow} \coloneqq \bigcup_{i \in X\downarrow} \Big(\{i\} \times \{1 \dots n+1\} \times \{0,1\} \Big)$$

Therefore, for $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$ and $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}$, the local bag alignment is $A_X = A_{X\uparrow} \bigcup A_{X\downarrow}$. Let us further denote by $C(X, A_{X\uparrow})$ the cost for the partial optimal alignment ending with a particular $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$. Therefore, Rinaudo's algorithm [78] can be reformulated as:

$$C(X, A_{X\uparrow}) = \min_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}} \{LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', project(X', A_X))\} \tag{2.12}$$

$$M(X, A_{X\uparrow}) = \arg\min_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}} \{LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', project(X', A_X))\} \tag{2.13}$$

Here *LCost* function is to calculate the cost of bag alignment and section 3.1.2.4 will give more details about the function. The **project**(X', A_X) operation restricts a set A_X to the elements in a bag X'. The dynamic programming traverses the tree T from the leaf bags to the root and for each bag $X \in T$ computes partial alignments. At the root bag of T, the whole alignment instances between Q and W are taken into account. The time complexity for the algorithm is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw+1})$ where tw is the tree-width of Q. As shown by Rinaudo *et al.* [78], by finding a smooth tree decomposition of Q, the time complexity of the structure-sequence alignment algorithm can be reduced in $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$.

2.3.5 Other formula

There is an easy way to understand the tree decomposition based parameterized algorithm if you are familiar with the **variable elimination algorithm** which is used to solve the maximum a posteriori (MAP) problem in Bayesian networks or

Markov networks [53, 22]. Here I give a quite easy example using Markov networks which are based on the undirected graphs. Suppose we have four random variables A, B, C, D with relationship shown in Figure 2.14. Each variable has two values, for example the variable A has values a^0 and a^1. To parameterize the graph, a function called **factor** is defined which is used to capture the affinities between related variables (which has edges between them). Let $\mathcal{D}$ be a set of random variables, a factor ϕ is defined as a function from $Val(\mathcal{D})$ to $\mathbb{R}$ where $Val(\mathcal{D})$ represents all possible values for $\mathcal{D}$. As shown in Figure 2.14, each table corresponds a factor. In the leftmost table in Figure 2.14, $\mathcal{D} = \{A, B\}$ and one possible value in $Val(\mathcal{D})$ is (a^0, b^0).

As our example does not have evidence variables, our task is that given variables X, compute $MAP(x) = \arg\max_x P_\Phi(X = x)$ and $P_\Phi(x) \propto \prod_k \phi_k(\mathcal{D}_k)$ where Φ is defined as a set of factors ϕ in the Markov network. In Figure 2.14, $\Phi = \{\phi_1, \phi_2, \phi_3\}$. Therefore, $MAP(x) = \arg\max_x \prod_k \phi_k(\mathcal{D}_k)$. To solve the problem, the exact inference method variable elimination algorithm is used here. However, one usually transfer production to summation and the equation $\arg\max \prod_k \phi_k(\mathcal{D}_k)$ are transfered to $\arg\max \sum_k \theta_k(\mathcal{D}_k)$. Therefore, in our example as shown in Figure 2.14, we need to calculate $\arg\max \sum_{k=1}^{3} \theta_k(\mathcal{D}_k)$ where $\theta = log_2\phi$. The variable elimination algorithm can be described mathematically in the following way:

$$\max_C \max_B \max_A (\theta_1(A, B) + \theta_2(B, C) + \theta_3(C, D)) \tag{2.14}$$

As none of $\theta_2(B, C)$ and $\theta_3(C, D)$ are related to $\max_A$, the formula can be reformulated as

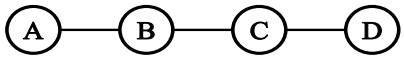
$$\max_C \max_B (\theta_2(B, C) + \theta_3(C, D) + \max_A \theta_1(A, B)) \tag{2.15}$$

$$\max_C \max_B (\theta_2(B, C) + \theta_3(C, D) + \lambda_1(B)) \tag{2.16}$$

Notice that in this step, we can eliminate the variable A by replacing $\max_A \theta_1(A, B)$ by $\lambda_1(B)$. $\lambda_1(B)$ is a function of variable B and for different values of B, we have different values. The first step involves an operation which is called **factor maximization** which is defined as follows:

Definition 6 (Factor maximization [53])**.** Let X be a set of variables, and $Y \notin X$ a variable. Let $\phi(X, Y)$ be a factor. We define the factor maximization of Y in ϕ to be a factor λ over X such that:

$$\lambda(X) = \max_Y \phi(X, Y).$$



A	B	ϕ_1	θ_1
a^0	b^0	2^7	7
a^0	b^1	2^2	2
a^1	b^0	2^3	3
a^1	b^1	2^3	3

B	C	ϕ_2	θ_2
b^0	c^0	2^7	7
b^0	c^1	$2^{4.5}$	4.5
b^1	c^0	$2^{1.2}$	1.2
b^1	c^1	2^3	3

C	D	ϕ_3	θ_3
c^0	d^0	2^1	1
c^0	d^1	2^9	9
c^1	d^0	2^2	2
c^1	d^1	$2^{3.5}$	3.5

Figure 2.14: Markov network example.

The process of the factor maximization operation is shown in Figure 2.15 A). The next step is to eliminate variable B. This step involves two factor operations, one is **factor summation** and the other one is **factor maximization** as shown in Figure 2.15 B) with equation:

$$\max_C(\theta_3(C,D) + \max_B(\lambda_1(B) + \theta_2(B,C))) \tag{2.17}$$

The last step is to eliminate C as shown 2.18

$$\max_C(\theta_3(C,D) + \lambda_2(C)) \tag{2.18}$$

The corresponding process is shown in Figure 2.16. We will eliminate variable C in this step and get a function $\lambda_3(D)$. Therefore, the maximum value is 23 when $D = d_1$. Now we execute the **backtrack** process to get the optimal value x^*. To make $\lambda_3(d^1) = 23$, we need $C = c^0$ and $D = d^1$ with $\lambda_2(c^0) = 14$ in Figure 2.16. To make $\lambda_2(c^0) = 14$, we need $B = b^0$ and $C = c^0$ with $\lambda_1(b^0) = 7$ in Figure 2.15 B). To make $\lambda_1(b^0) = 7$, we need $A = a^0$ and $B = b^0$ as shown in Figure 2.15 A). The optimal value is 23 with $A = a^0$, $B = b^0$, $C = c^0$ and $D = d^1$.

Now let us reconsider Rinaudo's algorithm. Suppose we have an arc-annotated sequence $Q = (S_Q, \emptyset)$ with AUUA and the other arc-annotated sequence $W = (S_W, \emptyset)$ with AA. We want to get an optimal alignment between Q and W. Q has four nucleotides, each nucleotide corresponds to a variable A, B, C, D. Each variable has six values which are $1, 2, 3, 4, 5, 6$. $1, 2, 3$ means that the position i of Q is aligned to the position $1, 2, 3$ of W with matched case and $4, 5, 6$ means that the position i of Q is aligned to the position $1, 2, 3$ with unmatched case. Therefore, an alignment

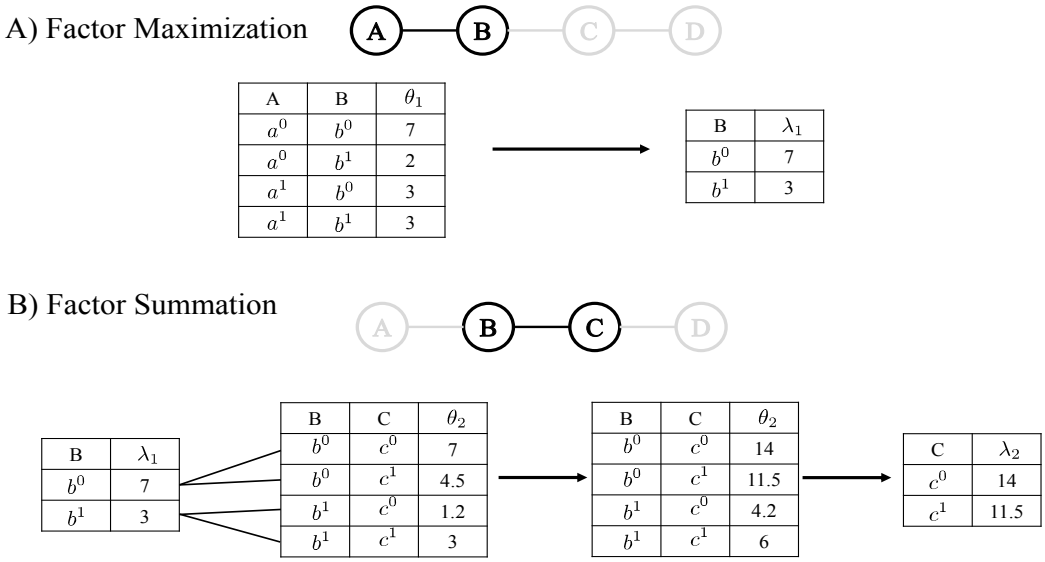


Figure 2.15: The step to eliminate variable B with factor operations: factor summation and factor maximization.

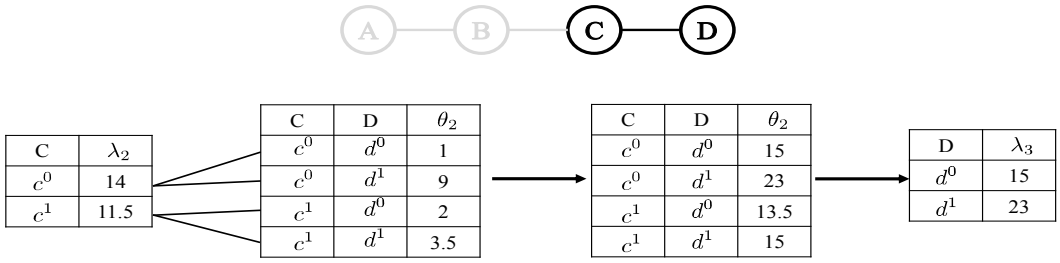


Figure 2.16: The step to eliminate variable C.

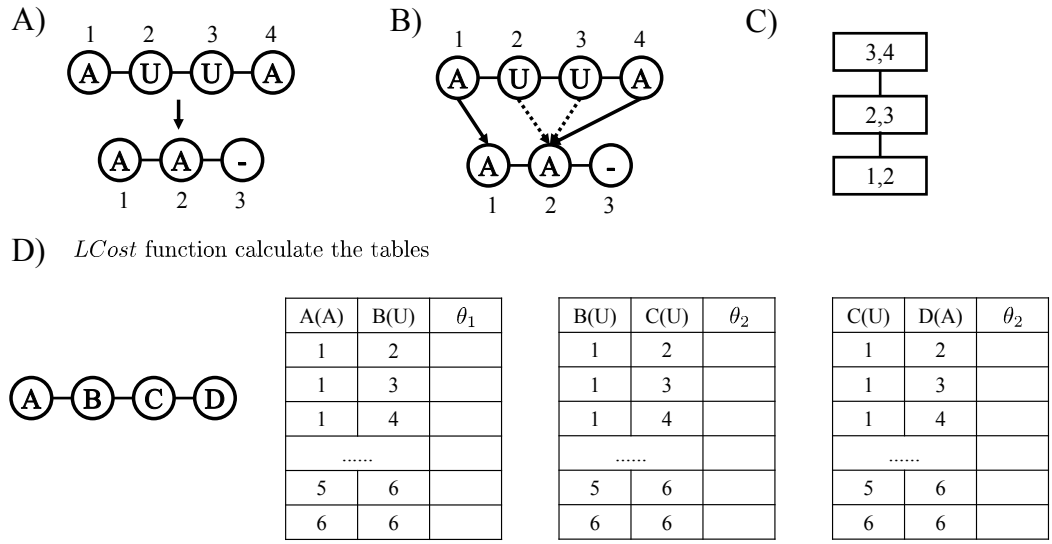


Figure 2.17: An alignment example.

with values $A = 1, B = 5, C = 5, D = 2$ can be transferred to the alignment triples $(1, 1, 1)$, $(2, 2, 0)$, $(3, 2, 0)$ and $(4, 2, 1)$ and the alignment is shown in Figure 2.17 B). The *LCost* function in the previous DP equation is to calculate the tables which determine the factor θ in the Markov network shown in Figure 2.17 D). Then operations factor summation and factor maximization are used to calculate the optimal alignment.

The function of the tree decomposition is to find the elimination ordering. Figure 2.17 C) is the tree decomposition of Q with AUUA. As previously mentioned, I split a bag X into $X\uparrow$ and $X\downarrow$ and the set $X\downarrow$ is the eliminating variable in the bag. For the bag $(1, 2)$, $X\downarrow$ is 1, for the bag $(2, 3)$, $X\downarrow$ is 2 and for the bag $(3, 4)$, $X\downarrow$ is 3. So the eliminating order is 1, 2, 3. I will show more details about the tree decomposition and the DP recursion in the next chapter.

Chapter 3

Methods

This chapter describes the algorithmic foundations of our new program `LiCoRNA` (a`Li`gnment of `Co`mplex `RNA`s).

A first set of algorithmic methods are dedicated to the, stochastic or deterministic, **generation of suboptimal alignments**. Our rationale for exploring the space of suboptimal, or near-optimal, alignments is three-fold:

- Firstly, one optimal alignment is ambiguous, meaning that several alignments that have the same score may co-exist [32]. Such co-optimal alignments may additionally feature very different correspondences, so that the choice of an arbitrary alignment within the set of feasible alignments is likely to impact greatly the quality of our conclusions;
- Secondly, one optimal alignment is only an approximation of the biologically-relevant one, ideally revealing functional homology through correspondences. Producing a set of alignments whose score is close to the optimal one therefore increases the probability to capture *true* alignments, despite neglecting some aspects of the probabilistic model [64].
- Thirdly, one optimal alignment may be sensitive to perturbations of the scoring parameters, especially gap penalties [103]. Considering a set of near-optimal solutions provides a good way to empirically assess such an instability of the prediction and, using stochastic sampling, to estimate a notion of support for individual correspondences.

Accordingly, we contribute three sets of novel algorithms: a) stochastic backtrack algorithm (Section 3.2.4), based on the computation of a pseudo-partition function (Section 3.2.3); b) Δ-suboptimal generation algorithm (Section 3.3.2); and c) K-best algorithm (Section 3.3.1).

Finally, we explore a systematic approach to assess the robustness of an optimal alignment, through its comparison with the **Maximum Expected Accuracy** (MEA) structure-sequence alignment. The MEA alignment can intuitively be compared to the center of mass of the alignment space, and is a natural generalization of the eponymous notion introduced by Do *et al.* [28] in the context of pairwise/multiple sequence alignment. Our rationale is that, if the alignment is well-defined, then the MEA alignment should be close to the optimal one. Conversely, an ill-defined optimal alignment, admitting many suboptimal alignments, is expected to be either distant from the MEA, or have poor associated accuracy.

The computation of the MEA requires the preliminary computation of the posterior probabilities for all possible correspondences of bases and base pairs. In this work, we either empirically estimate these probabilities using stochastic sampling, or compute those exactly through a custom parameterized instance of the **inside-outside algorithm** (Section 3.2.5). Finally, in Section 3.2.6, we adapt Rinaudo's algorithm to compute the MEA alignment based on the correspondence probabilities of bases and base pairs.

3.1 Model and definitions

3.1.1 Tree decomposition and its practical computation

3.1.1.1 Definitions

Let us recall the definition of tree decomposition and some notations used.

Definition 7 (Tree Decomposition of an arc-annotated sequence). A tree decomposition of an arc-annotated sequence $Q = (S_Q, P)$ is a rooted tree T whose nodes, called bags, are sets of positions in Q. Additionally, the set $\mathcal{X}$ of bags must satisfy:

- (Node coverage) $\forall s \in S, \exists X \in \mathcal{X}$ such that $s \in X$;

- (Edge coverage) $\forall (i,j) \in P, \exists X \in \mathcal{X}$ such that $\{i,j\} \subset X$. Moreover, $\forall 1 \leq i < m, \exists X \in \mathcal{X}$ such that $\{i, i+1\} \subset X$;
- (Coherence) $\forall X, X', X'' \subset \mathcal{X}$, if X'' lies on the path from X and X' on T, then $X \cap X' \subset X''$.

The **width** of a tree decomposition T is defined as $\max_{X \in \mathcal{X}} |X| - 1$. The **tree-width** tw of an arc-annotated sequence Q is the minimum width over all possible tree decompositions of Q. In the following sections, we will use a simple pseudoknot, shown in Figure 3.1, as a running example.

We split the indices found in a bag X into two parts: the **transition** indices $X\uparrow$ and the **proper** indices $X\downarrow$, respectively such that

$$X\uparrow := \{i \in X \mid i \in p(X)\} \text{ and } X\downarrow := \{i \in X \mid i \notin p(X)\}.$$

We denote by $p(X)$ the **father** bag of X in the tree decomposition, and by $P(X) := \{(i,j) \mid i,j \in X \text{ and } i \text{ or } j \in X\downarrow\}$ the set of **proper base pairs** of a bag, involving at least one proper indices. Similarly,

$$N(X) := \{(i, i+1) \mid i, i+1 \in X \text{ and } i \text{ or } i+1 \in X\downarrow\}$$

represents the set of **proper backbone interactions**, consecutive positions involving the proper indices.

3.1.1.2 Practical computation of the tree decomposition

As we treat the arc-annotated sequence Q as an RNA graph, we can use the tree decomposition of a general graph to decompose RNA graphs. However, finding the optimal tree-width and tree decomposition for a general graph is an NP-hard problem [10] and computing the exact tree-width is very computationally expensive. Fortunately, efficient heuristic algorithms have been proposed for computing upper/lower bounds on the tree-width in a constructive fashion [11, 12].

Here we use `LiBTW`, a JAVA library implementing various tree decomposition algorithms (http://www.treewidth.com/treewidth/). In particular, we recommend to use the GREEDYDEGREE and GREEDYFILLIN heuristics [102] because of their good empirical behavior, as shown in Section 4.2.

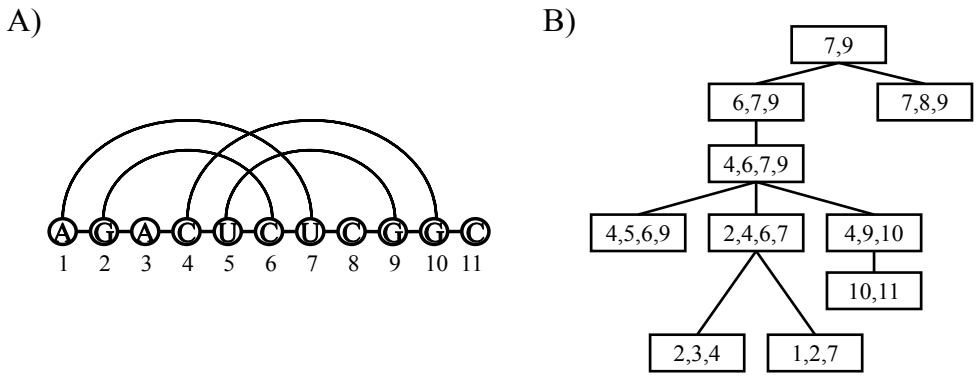


Figure 3.1: Arc-annotated sequence of simple pseudoknot and a corresponding tree decomposition. The width of this tree decomposition is 3. We treat the arc-annotated sequence as a RNA graph. Each vertex in the RNA graph exists in at least one bag. For example, vertex 3 exists in the bag $\{2, 3, 4\}$. For an edge (u, v) in the RNA graph, there is a bag containing (u, v). For example, bag $(1, 2, 7)$ contains the edge $(1, 7)$. Given three bags X, X', X'', if X'' lies on the path from X and X' on T, then $X \cap X' \subset X''$. For example, $X = \{2, 4, 6, 7\}$, $X' = \{7, 9\}$ and $X'' = \{6, 7, 9\}$, $X \cap X' = \{7\} \subset X''$.

3.1.2 Scoring scheme

3.1.2.1 RIBOSUM

Let us first describe RIBOSUM (RIBOsomal RNA SUbstitution Matrix) scoring matrices [52] which are used as parameters in our cost function.

Based on BLOSUM (BLOcks SUbstitution Matrix) matrices for protein alignment by Henikoff & Henikoff [41], Klein & Eddy proposed the RIBOSUM matrices in [52]. The data they used to generate the matrices is from high-quality alignments in European Ribosomal RNA Database [21]. The substitution matrix in RIBOSUM matrices gives the log-odds ratio for a given substitution relative to background nucleotide frequencies. For **single base substitution** matrix S, individual scores are calculated as

$$s_{ij} = \log_2 \frac{f_{ij}}{f_i \cdot f_j}.$$

where given time interval, f_{ij} is the observed frequency that nucleotide type i will change to nucleotide type j in an evolutionary process. The background frequency f_i

A)

AA	2.49														
AC	7.04	2.11													
AG	8.24	8.89	0.80												
AU	4.31	2.04	5.13	-4.49											
CA	8.84	9.37	10.41	5.56	5.13										
CC	14.37	9.08	14.50	6.71	10.45	3.59									
CG	4.68	5.86	4.57	-1.67	3.57	5.70	-5.36								
CU	12.64	10.45	10.14	5.17	8.49	5.77	4.96	2.28							
GA	6.86	9.73	8.61	5.33	7.98	12.43	6.00	7.71	1.05						
GC	5.03	3.81	5.77	-2.70	5.95	3.70	-2.11	5.84	4.88	-5.62					
GG	8.39	11.05	5.38	5.61	11.36	12.58	4.66	13.69	8.67	4.13	1.98				
GU	5.84	4.72	6.60	-0.59	7.93	7.87	0.27	5.61	6.09	-1.21	5.77	-3.47			
UA	4.01	5.32	5.43	-1.60	2.42	6.88	-2.75	4.72	5.85	-1.60	5.75	0.57	-4.97		
UC	11.32	8.67	8.87	4.81	7.08	7.40	4.91	3.83	6.63	4.49	12.01	5.30	2.98	3.21	
UG	6.16	6.93	5.94	0.51	5.63	8.41	-1.32	7.35	7.55	0.08	4.27	2.09	-1.14	4.76	-3.36
UU	9.04	7.83	11.07	2.98	8.39	5.41	3.67	5.21	11.54	3.90	10.79	4.44	3.39	5.98	4.28	0.01
	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU

B)

	A	C	G	U
A	-2.22			
C	1.86	-1.16		
G	1.46	2.48	-1.03	
U	1.39	1.05	1.74	-1.65

Figure 3.2: RIBOSUM85-60 matrix. (A) The 16×16 matrix is used to get scores for aligning base pairs. This matrix is a candidate to the base pair substitution matrix S' (B) The 4×4 matrix is used to get scores for aligning single-stranded regions. This matrix is a candidate to the base substitution matrix S.

for a nucleotide type i is probability of i in a randomly shuffled sequence.

For **base pair substitutions** matrix S', the individual score is computed by

$$s'_{i,j,k,l} = \log_2 \frac{f'_{i,j,k,l}}{f_i \cdot f_j \cdot f_k \cdot f_l}$$

In this case, nucleotide type i forms a base pair with j and k is paired to l. $f'(i,j,k,l)$ is the observed frequency that base pair $i \sim j$ change to base pair $k \sim l$ in the evolution process within a given time interval. f_i, f_j, f_k and f_l have the same meaning as before. As suggested by Klein & Eddy [52], we use RIBOSUM85-60 as shown in Figure 3.2 as the default matrix.

3.1.2.2 Cost function for structure-sequence alignment

A sequence-structure alignment can be represented as a sequence of **alignment triples**, each one an element of $\hat{\mathscr{A}}_{m,n} \coloneqq \{1 \dots m\} \times \{1 \dots n+1\} \times \{0,1\}$. A triplet $(i,j,1) \in \hat{\mathscr{A}}_{m,n}$ means that the sequence position i aligns to position j, whereas a triplet $(i,j,0) \in \hat{\mathscr{A}}_{m,n}$ does not match i to j, but constrains subsequent sequence positions to be aligned after j. To ease the writing of certain equations and algorithms,

we introduce three functions θ, γ, and δ, each accessing a specific element of a triplet, namely: $\forall a = (i, j, k) \in \hat{\mathscr{A}}_{m,n}, \theta(a) \to i, \gamma(a) \to j$ and $\delta(a) \to k$.

To score a sequence-structure alignment A, we make a slight change of equation 2.1 by adding certain restrictions on the scoring of arc-altering and arc-removing.

Definition 8 (**Cost of a structure-sequence alignment**)**.** Given an arc-annotated sequence $Q = (S_Q, P)$ and a plain sequence $W = (S_W, \emptyset)$, the cost of a structure-sequence alignment A between Q and W with minimum cost is defined as:

$$\begin{aligned} cost(A) = & \sum_{\substack{a \in A \\ \text{s.t.} \delta(a) \neq 0}} \sigma(a) + \sum_{\substack{a,b \in A \text{ s.t.} (\theta(a), \theta(b)) \in P \\ \delta(a) \neq 0, \delta(b) \neq 0}} \tau(a, b) \\ & + \sum_{\substack{a,b \in A \\ \text{s.t. } \theta(a) = i, \theta(b) = i+1}} \alpha_Q + (\gamma(b) - \gamma(a)) \cdot \beta_Q \\ & \sum_{\substack{a \in A \\ \text{s.t. } \delta(a) = 0}} \beta_W + \sum_{\substack{a,b \in A \\ \text{s.t. } \theta(a) = i, \theta(b) = i+1 \\ \delta(a) = 0, \delta(b) = 1}} \alpha_W \end{aligned} \tag{3.1}$$

where σ, τ are the cost function of base substitution and base pair substitution, separately. α_Q and β_Q are the gap open and gap extension penalties for the arc-annotated sequence and α_W and β_W are the gap open and gap extension penalties for the plain sequence.

The cost function is defined from the set of edit operations in [48]. Next, we will consider how to emulate the edit operations in our model. For free bases, the operation includes base match, base mismatch and base deletion. We used the 4×4 matrix of RIBOSUM85-60 [78] in Figure 3.2 to score **base match** and **base mismatch**. To score **base deletion** and **insertion**, we use an affine penalty function as shown in Table 3.1. Basic arc operations, such as **arc-match** and **arc-mismatch**, are also scored using the 16×16 matrix of RIBOSUM85-60, as shown in Figure 3.2.

Scoring some arc operations requires more complex strategies. Similarly to the work by Jiang *et al.* [48] and Siebert *et al.* [89], we add certain restrictions on the scoring of arc-altering and arc-removing. For the current bag X with $(i, j) \in P(X)$ and $a, b \in A_X$ s.t. $\delta(a) \neq 0$ and $\delta(b) = 0$, the score of **arc-altering** S_{aa} can be expressed, using the existing notations, as

$$S_{aa}(a) = \sigma(a) + \beta_W. \tag{3.2}$$

Parameter	Value
$\sigma(a)$	*base substitution.* Negative RIBOSUM85-60 4×4 matrix
$\tau(a, b)$	*base pair substitution.* Negative RIBOSUM85-60 16×16 matrix
λ_Q^1	$5 + 2(\ell - 1)$ for single-stranded regions in Q
λ_Q^2	$10 + 5(\ell - 1)$ for stack regions in Q
λ_W^1	$5 + 2(\ell - 1)$ for single-stranded regions in W
λ_W^2	$10 + 5(\ell - 1)$ for stack regions in W

Table 3.1: Scoring parameters in minimum cost alignment. ℓ is the length of gap. $\lambda_Q = \alpha_Q + (\ell - 1) \cdot \beta_Q$. $\lambda_W = \alpha_W + (\ell - 1) \cdot \beta_W$.

Finally, the score of **arc-removing** S_{ar} can be expressed as

$$S_{ar} = \beta_W + \beta_W. \tag{3.3}$$

The effect of this restriction is that one can evaluate both arc ends in an alignment independently for the edit operation arc-altering and arc-removing.

Since stacked regions are generally more conserved than single-stranded regions, an optimal alignment is less likely to feature gaps in stacked regions [52]. Therefore, we give two different score values for the gaps in Q with parameters α_Q^1, β_Q^1, α_Q^2 and β_Q^2. The cost of an ℓ-sized **gap in a single-stranded region** within the structure Q is then given by

$$\lambda_Q^1 := \alpha_Q^1 + \beta_Q^1 \cdot (\ell - 1),$$

whereas the **gap cost within stems** is given by

$$\lambda_Q^2 := \alpha_Q^2 + \beta_Q^2 \cdot (\ell - 1).$$

We adopt the same strategy for λ_W^1 and λ_W^2 the gap costs for the sequence W.

By default, the suggested parameters set is listed in Table 3.1, but each can be overridden by the user. As expected, the use of affine cost functions can circumvent scattered alignments, having numerous short gaps. However, setting the gap opening and extension scores to too large values also results in scattered alignments, since the structure then becomes the main contributor to the score, leading to a maximization of the number of conserved base-pairs.

3.1.2.3 Scoring a tree decomposition

Let us further denote by $C(X, A_{X\uparrow})$ the cost of a partial optimal alignment ending with a particular $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$ for a particular bag X and the DP equation [78] is

$$C(X, A_{X\uparrow}) = \min_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}} \{LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', project(X', A_X))\} \quad (3.4)$$

where $\mathcal{A}_{X\uparrow}$ represents all possible alignments for $X\uparrow$. The **project**(X', A_X) operation restricts a set A_X to the elements in a bag X'. *LCost* function is to calculate the cost of bag alignment. The dynamic programming traverses the tree T from the leaf bags to the root and at the root bag of T, the whole alignment instances between Q and W are taken into account.

Intuitively, we treat the arc-annotated sequence Q as an RNA graph and the function of tree decomposition is to generate the eliminating order for the vertex in the RNA graph (in section 2.3.5). According to the definition of tree decomposition, each vertex and each edge in the RNA graph appear in the bags in the tree decomposition. Therefore, the subsequent DP can traverse all the vertices and edges at least once. Each time we eliminate an RNA vertex, the cost function σ or β_W will be considered. Each time we eliminate the associated edges and if the edge is an interaction edge, the cost function τ, σ and β_W will be considered. If the edge is a consecutive edge, α_Q, β_Q and α_W will be considered. The *LCost* function is used to add the local cost to the cost of the partial alignment and at the same time avoid accounting for the cost items more than once for each vertex and edge.

3.1.2.4 *LCost* function

Let us focus on *LCost* function for one bag. Two consecutive alignment triples $a, b \in \hat{\mathscr{A}}_{m,n}$ with $\theta(b) - \theta(a) = 1$ and $(\theta(a), \theta(b)) \in N(X)$ are said to be in conflict which gives $+\infty$ score, if the following cases satisfy, :

- $\gamma(a) > \gamma(b)$
- $\gamma(a) = \gamma(b), \delta(a) = 1$
- $\gamma(a) < \gamma(b), \delta(a) = 0$

Two alignment triples $a, b \in \hat{\mathscr{A}}_{m,n}$ with $\theta(a) < \theta(b)$ and $(\theta(a), \theta(b)) \in P(X)$ are said be to in conflict, if the following cases satisfy:

- $\gamma(a) > \gamma(b)$
- If $\theta(b) - \theta(a) > 1$, $\gamma(a) = \gamma(b), \delta(a) = 1$
- If $\theta(b) - \theta(a) = 1$, the cases are the same as consecutive alignment triples.

Therefore, we give a definition of **valid bag alignment** A_X between a bag X and W which is not $+\infty$.

Definition 9 (**valid bag alignment**)**.** Given the current bag X and a plain sequence W, a bag alignment A_X is valid if the following cases satisfy:

- $\forall i, j \in X$, $a, b \in A_X$, s.t. $\theta(a) = i$ and $\theta(b) = j, i < j$, (1) $\gamma(a) < \gamma(b)$. (2) $\gamma(a) = \gamma(b)$, $\delta(a) = 0$.
- For $(i, j) \in P(X)$, $a, b \in A_X$, s.t. $\theta(a) = i$ and $\theta(b) = j$, a and b is not in conflict.
- For $(i, i+1) \in N(X)$, $a, b \in A_X$, s.t. $\theta(a) = i$ and $\theta(b) = i + 1$, a and b is not in conflict.
- For $i \in t$, $a \in A_X$, s.t. $\theta(a) = i$ and $\gamma(a) = n + 1, \delta(a) = 0$.

The *LCost* function gives $+\infty$ for bag alignment that is not valid and calculates the local contribution of valid bag alignment A_X to get the partial alignment score $C(X, A_{X\uparrow})$. The *LCost* function is defined as follows:

$$
\begin{aligned}
LCost(A_{X\uparrow}, A_{X\downarrow}) = & \sum_{\substack{a \in A_{X\downarrow} \\ \text{s.t. } \delta(a) \neq 0}} \sigma(a) + \sum_{\substack{(i,j) \in P(X), a, b \in A_X \\ \text{s.t. } \delta(a), \delta(b) \neq 0 \\ \text{and } \theta(a)=i, \theta(b)=j}} \tau(a, b) \\
& \sum_{\substack{(i,i+1) \in N(X), a, b \in A_X \\ \text{s.t. } \theta(a)=i, \theta(b)=i+1}} \alpha_Q + (\gamma(b) - \gamma(a)) \cdot \beta_Q \\
& \sum_{\substack{a \in A_{X\downarrow} \\ \text{s.t. } \delta(a)=0}} \beta_W + \sum_{\substack{(i,i+1) \in N(X), a, b \in A_{X\downarrow} \\ \text{s.t. } \theta(a)=i, \theta(b)=i+1 \\ \delta(a)=0, \delta(b)=1}} \alpha_W
\end{aligned} \tag{3.5}
$$

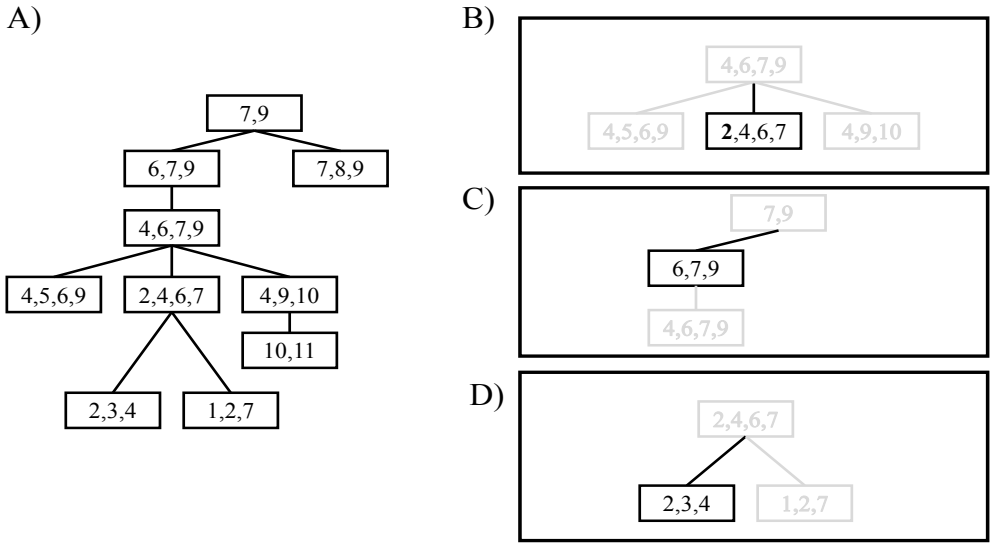


Figure 3.3: (A) tree decomposition example. (B) For $X = \{2, 4, 6, 7\}$ and $X\downarrow = \{2\}$ where $(2, 6) \in P$, function τ is considered. (C) For $X = \{6, 7, 9\}$ and $X\downarrow = \{6\}$, neither function τ nor function σ will be considered. (D) For $X = \{2, 3, 4\}$ and $X\downarrow = \{3\}$, function σ will be considered.

σ, τ are the cost function of base substitution and base pair substitution, separately. α_Q and β_Q are the gap open and gap extension penalty for arc-annotated sequence and α_W and β_W are the gap open and gap extension penalty for the plain sequence.

Though given *LCost* function, the calculation of base substitution σ function and base pair substitution τ function for the current bag alignment A_X are still tricky. For $a \in A_{X\downarrow}$, there are three cases to consider for σ function and τ function and the cases are shown in Figure 3.3.

- $\theta(a)$ (get the first element of a) does not form base pair in the arc-annotated sequence, we only consider the function σ.
- $\theta(a)$ forms base pair with j in Q and $j \in X$, we only consider the function τ.
- $\theta(a)$ forms base pair with j in Q, but $j \notin X$. Neither function σ nor function function τ will be considered.

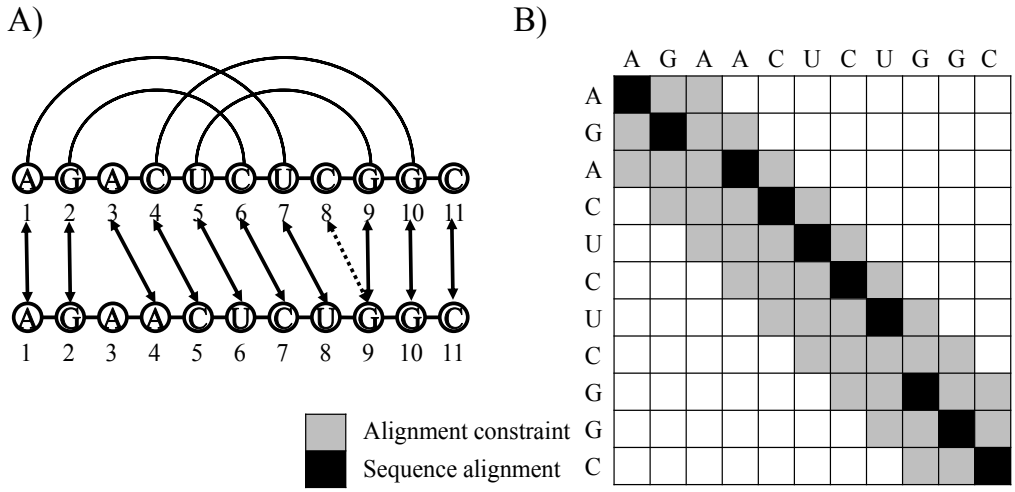


Figure 3.4: (A) An example alignment. (B) Illustration of N constraint alignment. The black squares shows the alignment position in (A) and the final alignment (A) lie inside the constraint scanned region (shows in gray).

3.1.3 Banded dynamic programming

To accelerate the dynamic programming algorithm without losing too much accuracy, we implemented a banded dynamic programming heuristics, as described by Harmanci *et al.* [40]. Banded DP is a heuristics method which is achieved by trading optimality for speed. In banded DP, to solve the alignment problem, one uses some fast methods to identify the band (the scanned region in the target sequence) for each position in the query sequence, and calculate the optimal alignment within that band for each position. The optimal alignment is more likely missing when the evolutionary distance between query and target sequence increases.

In our case, for position i in Q which is supposed to align to position j in W, the following constraint must be satisfied:

$$\left| \frac{n \times i}{m} - j \right| \leq N \tag{3.6}$$

where N is a constraint set by the user, which implicitly restricts to which sequence positions each structure position can be matched, m is the length of the arc-annotated sequence Q, and n is the length of the plain sequence W.

By default, N is set to the difference between the two sequence lengths. The parameter N restricts the length of insertions within homologous sequences which is reasonable for comparing close homologs. However, it may hurt the quality of our results in cases where long domains are inserted throughout evolution, so N can also be set interactively by the user to detect and/or work around such issues. As shown in equation 3.6, the parameter N reduces the computational cost by restricting the scanned region for position i in Q which is illustrated in Figure 3.4. More specifically, using the banded DP reduces the time complexity to $\mathcal{O}(|\mathcal{X}| \cdot N^{tw})$.

3.2 Probabilistic structure-sequence alignment

In [61], McCaskill proposed a cubic time algorithm to compute the partition function which assumes the energy of each secondary structure of a given RNA sequence is weighted by the corresponding Boltzmann probability. The partition function was subsequently used to calculate the base pairing probability for every base pair in the RNA sequence. In 1995, Miyazawa computed the partition function and probabilities of residue-residue correspondence for alignment problem [62]. Later, in 2002, M*ü*ckstein *et al.* [63] presented the stochastic backtracking algorithm to generate pairwise sequence alignments from the Boltzmann conditional probability distribution based on the partition function. Here, we adopt a similar approach to explore the solution space of structure-sequence alignment.

To that purpose, we generalize the stochastic backtracking algorithm to generate an ensemble of suboptimal structure-sequence alignments. This algorithm includes two basic steps: (1) the **forward step** computes the partition function for the partial alignments $M(X, A_{X\uparrow})$; (2) a **stochastic backtrack** step generates a statistically representative sample of the alignments. However, in order to directly derive from the main steps in Rinaudo's algorithm, we need to ascertain that its underlying DP decomposition respects some notions of **completeness** and **unambiguity** [75]. Here, we first need to introduce the concept of **derivation** to define completeness and unambiguity for DP recursions.

3.2.1 Derivation and derivation tree

To describe the derivation process associated with a tree decomposition, we define a **state space**

$$\mathcal{Q} = \{(X, A_{X\uparrow}) \mid X \in \mathcal{X}, A_{X\uparrow} \in \mathcal{A}_{X\uparrow}\}.$$

A **state derivation** is then of the form:

$$q \xrightarrow[s]{A_{X\downarrow}} q'_1, q'_2, \ldots, q'_c \tag{3.7}$$

where $q, q'_1, q'_2, \ldots, q'_c$ are states belong to $\mathcal{Q}$, X is a bag in the tree decomposition, $A_{X\uparrow}$ is a particular alignment set for transition indices $X\uparrow$ and $A_{X\downarrow}$ is a particular alignment set for proper indices $X\downarrow$ and s is the score of the transition. Intuitively, derivations correspond to the assignment of the proper indices for a bag, leading to a local contribution s to the overall score, followed by a transition towards a set of children states.

In the context of Rinaudo's algorithm, a state derivation for a particular state $(X, A_{X\uparrow})$ is of the form:

$$\begin{aligned}(X, A_{X\uparrow}) &\xrightarrow[s]{A_{X\downarrow}} (X'_1, B_1), \ldots (X'_c, B_c), \text{ if } child(X) \neq \emptyset \\ \text{or } (X, A_{X\uparrow}) &\xrightarrow[s]{A_{X\downarrow}} \emptyset, \text{ if } child(X) = \emptyset\end{aligned} \tag{3.8}$$

where $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}$, $c = |child(X)|$, X'_1, …, $X'_c \in child(X)$ are the children bags of X in the tree decomposition, and $B_i = project(X'_i, A_X)$ is the alignment set for $X'_i \cap X$. $s = LCost(A_{X\uparrow}, A_{X\downarrow})$ is the local score. Theoretically, there are $(2\cdot(n+1))^{|X\downarrow|}$ distinct derivations from state $(X, A_{X\uparrow})$ where $|S_W| = n$.

A state derivation is **valid** if $A_X = A_{X\uparrow} \cup A_{X\downarrow}$ is a valid bag alignment (defined in Section 3.1.2.4). The set of admissible values $A_{X\downarrow}$ within a context $A_{X\uparrow}$ is denoted as $\mathcal{A}_{X\downarrow}(X, A_{X\uparrow}) \subseteq \mathcal{A}_{X\downarrow}$. Here, a state $(X_0, \emptyset)$ represents the start state, reminding that X_0 is the root bag in the tree decomposition, and the father of X_0 is $p(X_0) = \emptyset$. Figure 3.5 shows an example of such a derivation.

A **derivation tree** from a state $(X, A_{X\uparrow})$ is an ordered, rooted tree where each node in the tree corresponds to a state and each edge corresponds to a $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$ with score s. In order to represent a tree, we extend the notations introduced to depict

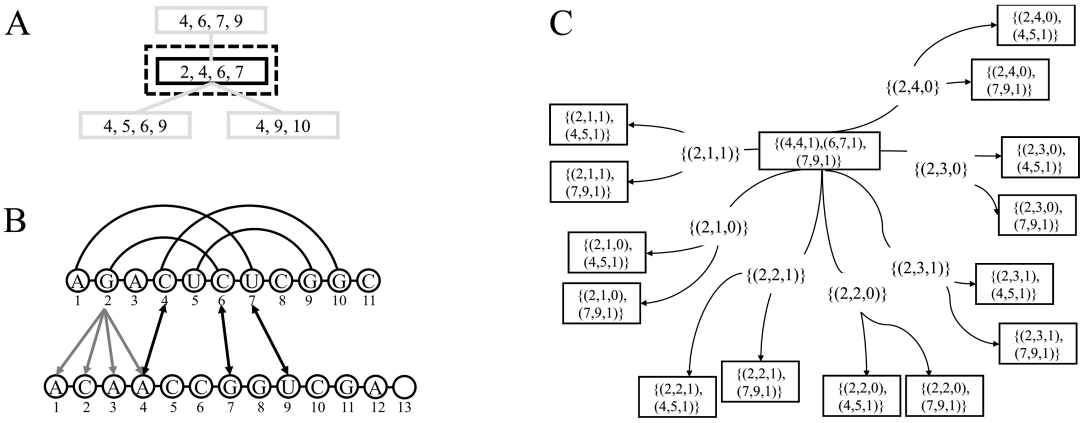


Figure 3.5: Illustration of the concept of derivation, starting from a state $(X, A_{X\uparrow})$. **(A)** Part of the tree decomposition in Figure 3.1. Here we consider the bag with indices $\{2, 4, 6, 7\}$. **(B)** In this case, $X\downarrow = \{2\}$, $X\uparrow = \{4, 6, 7\}$ and we assume $A_{X\uparrow} = \{(4, 4, 1), (6, 7, 1), (7, 9, 1)\}$. To make a valid bag alignment, position 2 can map to position $\{1, 2, 3\}$ with matched or unmatched case and map to position 4 with only unmatched case. **(C)** derivation representations for different $A_{X\downarrow}$.

state derivations, and denote recursively a derivation tree as

$$(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t_c, \text{ if } child(X) \neq \emptyset$$
$$\text{or } (X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} \emptyset, \text{ if } child(X) = \emptyset$$

where, for each $i \in [1, c]$, t_i is a derivation tree rooted in (X'_i, B_i), and the $\emptyset$ notation is now abused to represent the empty tree.

$\mathcal{D}(X, A_{X\uparrow})$ is the **set of derivation trees** starting from state $(X, A_{X\uparrow})$ and $\mathcal{D}(X, A_{X\uparrow})$ can be recursively described as

$$\mathcal{D}(X, A_{X\uparrow}) = \left\{(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} \emptyset, \text{ if } child(X) = \emptyset\right\} \tag{3.9}$$
$$\bigcup_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} \left\{(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t_c \text{ if } child(X) \neq \emptyset\right\}$$

where $(t_1, \cdots, t_c) \in \mathcal{D}(X'_1, B_1) \times \cdots \times \mathcal{D}(X'_c, B_c)$.

3.2.2 Completeness and unambiguity of Rinaudo's DP scheme

Now let us move our focus on completeness and unambiguity of Rinaudo's algorithm. Given Q and W, the search space contains all the alignments satisfying Definition 1.

The search space is denoted as $\mathcal{S}$ and derivation space from the DP recursion is $\mathcal{D}$. A DP recursion is unambiguous if and only if any pair of distinct derivations in $\mathcal{D}(X_0, \emptyset)$ originating from state $(X, \emptyset)$ give rise to different alignments in the search space $\mathcal{S}$. It is complete if and only if every alignment in the search space $\mathcal{S}$ corresponds to at least one derivation in $\mathcal{D}(X_0, \emptyset)$. Following the idea in [17], we define a function $\Phi : \mathcal{D}_I \rightarrow \mathcal{S}_I$ to represent the correspondence between an instance I in derivation space S and an instance I in search space $\mathcal{S}$. Therefore, if the DP recursion is unambiguous and completeness, the function Φ is injective and surjective. $\mathcal{S}(X, A_{X\uparrow})$ is used to represent all valid alignments between the subset of positions appearing in the descendants of X in the tree and W within the context $A_{X\uparrow}$.

Proposition 1. The DP recursion of Rinaudo's algorithm is unambiguous: for any derivation trees D_1 and D_2 starting from state $(X, \emptyset)$, $D_1 \neq D_2 \implies \Phi(D_1) \neq \Phi(D_2)$.

Proof. We will prove it by induction. In other words, we want to prove that for a given bag X, any possible alignment set $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$ for set $X \uparrow$ and any derivation trees $D_1 \in \mathcal{D}(X, A_{X\uparrow})$ and $D_2 \in \mathcal{D}(X, A_{X\uparrow})$ starting from the state $(X, A_{X\uparrow})$, $D_1 \neq D_2 \implies \Phi(D_1) \neq \Phi(D_2)$.

Base case. We consider the case where the bag X is a leaf bag in the tree decomposition. The statement is obviously satisfied. The leaf bag does not have children. For different derivation trees D_1 and D_2, they have different alignment sets $A_{X\downarrow}$ and $A'_{X\downarrow}$. Obviously different sets $A_{X\downarrow}$ and $A'_{X\downarrow}$ correspond to different alignments for the indices in $X\downarrow$. $\Phi(D_1)$ and $\Phi(D_2)$ contains the alignment triple for the indices in X. So $\Phi(D_1) = A_{X\uparrow} \cup A_{X\downarrow}$, $\Phi(D_2) = A_{X\uparrow} \cup A'_{X\downarrow}$ corresponds two different alignments in set $\mathcal{S}(X, A_{X\uparrow})$.

Induction hypothesis. Given each child bag X, for any possible alignment set $A_{X\uparrow}$ and any derivation trees D_1 and D_2 from state $(X, A_{X\uparrow})$, $D_1 \neq D_2 \implies \Phi(D_1) \neq \Phi(D_2)$.

Induction argument. Given father bag X, for any possible alignment set $A_{X\uparrow}$, we have state $(X, A_{X\uparrow})$. Reminding that the set of derivation trees starting from state $(X, A_{X\uparrow})$ can be represented as $\mathcal{D}(X, A_{X\uparrow}) = \bigcup_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} \{(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t_c\}$ where $(t_1, \cdots, t_c) \in \mathcal{D}(X'_1, B_1) \times \cdots \times \mathcal{D}(X'_c, B_c)$, different derivation trees D_1 and D_2 from state $(X, A_{X\uparrow})$ can differ in two cases. The first is $A_{X\downarrow}$ and $A'_{X\downarrow}$ are different. The alignment for positions $X\downarrow$ will be considered for $\Phi(D_1)$ and $\Phi(D_2)$ in $\mathcal{S}(X, A_{X\uparrow})$. So $\Phi(D_1)$ and $\Phi(D_2)$ are different. The second case is that the derivation trees for one or more children X' originating from state (X', B) where $B = project(X', A_X)$

are different. Suppose the derivation trees for ith child are different in D_1 and D_2. Then the derivation of D_1 can be represented as $(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t_i, \cdots, t_c$ and the derivation of D_2 can be represented as $(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t'_i, \cdots, t_c$. According to the induction hypothesis, $t_i \neq t'_i \implies \Phi(t_i) \neq \Phi(t'_i)$ and $\Phi(t_i) \subseteq \Phi(D_1)$, $\Phi(t'_i) \subseteq \Phi(D_1)$. Therefore, $\Phi(D_1) \neq \Phi(D_2)$ This completes the proof of unambiguity of Rinaudo's algorithm. □

Next we prove the completeness of Rinaudo's algorithm.

Proposition 2. The DP recursion of Rinaudo's alogrithm is complete: for any alignment A in the search space $\mathcal{S}$ between Q and W, there exists a derivation tree D starting from state $(X, \emptyset)$ so that $A = \Phi(D)$.

Proof. Recalling that given a bag X, we denote the search space from state $(X, A_{X\uparrow})$ by $\mathcal{S}(X, A_{X\uparrow})$ between the subset of positions appearing in the descendants of X in the tree decomposition and W ending with $A_{X\uparrow}$. We want to prove that for any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$, there exists a derivation tree D starting from state $(X, A_{X\uparrow})$ so that $A = \Phi(D)$.

Base case. We consider the case where X is a leaf bag in the tree decomposition. The statement holds obviously. Given any $A_{X\uparrow}$, we want to prove for any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$, there exists a derivation tree D starting from the state $(X, A_{X\uparrow})$ so that $A = \Phi(D)$. A valid alignment for the leaf bag with the state $(X, A_{X\uparrow})$ can be represented as $A = A_X = A_{X\uparrow} \cup A_{X\downarrow}$, $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$. Now we consider the set of derivation trees starting from state $(X, A_{X\uparrow})$, $\mathcal{D}(X, A_{X\uparrow}) = \bigcup_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} \xrightarrow[s]{A_{X\downarrow}} \emptyset$. So $|S(X, A_{X\uparrow})| = |\mathcal{D}(X, A_{X\uparrow})|$.

Induction hypothesis. Given each child bag X and a particular $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$, for any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$, there exists a derivation tree D starting from the state $(X, A_{X\uparrow})$ so that $A = \Phi(D)$.

Induction argument. Given father bag X and a particular $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$, for any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$ can be decomposed into several parts $A = A_{X\uparrow} \cup A_{X\downarrow} \cup \mathcal{S}(X'_1, B_1) \cup \ldots \cup \mathcal{S}(X'_c, B_c)$. The set of derivation trees from state $(X, A_{X\uparrow})$ can be represented as $\mathcal{D}(X, A_{X\uparrow}) = \bigcup_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} \{(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} t_1, \cdots, t_c\}$ where $(t_1, \cdots, t_c) \in \mathcal{D}(X'_1, B_1) \times \cdots \times \mathcal{D}(X'_c, B_c)$. Similar to the base case, we will consider all $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$ in our derivation. According to induction hypothesis, for

any alignment $A' \in \mathcal{S}(X'_i, B_i)$ of the child bag, there exists a derivation tree D' starting from the state (X'_i, B_i) in derivation space $\mathcal{D}(X'_i, B_i)$ so that $A' = \Phi(D')$. This completes the proof of completeness of Rinaudo's algorithm based on the fact of one-to-one correspondence. □

3.2.3 Computing the partition function

As explained earlier [62], in a thermodynamic interpretation of the alignment problem, the alignment score can be treated as negative energy and the partition function for the alignment A s.t. $\forall a \in A, a \in \hat{\mathscr{A}_{m,n}}$ between Q and W can be defined as

$$\mathcal{Z} = \sum_{\substack{A \\ \text{s.t. } \forall a \in A, a \in \hat{\mathscr{A}_{m,n}}}} e^{\frac{-C(A)}{Rt}} = \sum_{\substack{A \\ \text{s.t. } \forall a \in A, a \in \hat{\mathscr{A}_{m,n}}}} e^{-\beta C(A)} \tag{3.10}$$

where $\beta = \frac{1}{Rt}$. Here, t is analogous to the temperature in Kelvin. The constant R, the correspondence of Boltzmann constant, is set according to the substitution matrix. As mentioned in the previous section, the substitution matrix we used is a RIBOSUM matrix which is defined as a log-odds matrix and the individual score in single base substitution matrix S and base pair substitution matrix S' is calculated by

$$s_{ij} = \log_2 \frac{f_{ij}}{f_i \cdot f_j}$$
$$s'_{i,j,k,l} = \log_2 \frac{f'_{i,j,k,l}}{f_i \cdot f_j \cdot f_k \cdot f_l}$$

Therefore, the substitution score equals to a constant factor times log-odds ratio of the substitution. In the case of RIBOSUM matrix, R is calculated by

$$R = \frac{1}{\log_e 2} \approx 1.44279$$

The probability for a particular alignment A s.t. $\forall a \in A, a \in \hat{\mathscr{A}_{m,n}}$ satisfies:

$$\mathbb{P}(A) = \frac{1}{\mathcal{Z}} e^{-\beta C(A)}$$

The sum over the probabilities of all possible alignments between Q and W is 1,

$$\sum_{\substack{A \\ \text{s.t. } \forall a \in A, a \in \hat{\mathscr{A}_{m,n}}}} \mathbb{P}(A) = 1$$

The Boltzmann probability distribution gives a probability for every alignment, and therefore statistically characterizes the ensemble. Usually, parameter t in β is used to govern the Boltzmann distribution. When $t \to 0$, for the alignments with score bigger than the optimal alignment, the probability $\mathbb{P}$ approaches 0 and the probability of the optimal alignment approaches $1/u$ where u is the number of distinct alignments with the minimum score. If $t = 1$, we get the "true" probability which means that the distribution is only affected by the score of a particular alignment. If $t \to \infty$, all the valid alignments satisfying Definition 1 have the same probability which means that Boltzmann probability distribution is uniform.

The partition function can be easily computed using Rinaudo's algorithm by changing summation operation to multiplication operation and the minimum operation into summation operation as shown

$$\mathcal{Z}(X, A_{X\uparrow}) = \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \prod_{X' \in child(X)} \mathcal{Z}(X', project(X', A_X)) \quad (3.11)$$

There are still two cases to be considered. (1) If the current bag X is the leaf bag in the tree decomposition, the set $child(X) = \emptyset$. Then all the terms $\mathcal{Z}(X', project(X', A_X))$ are set to be 1 in equation 3.11. (2) If the bag alignment A_X is not a valid alignment, then the term $e^{-\beta \cdot LCost(A_{X\uparrow}, A_{X\downarrow})}$ is set to be 0. If the bag alignment is not valid, the alignment between Q and W derived from the invalid bag alignment does not satisfy Definition 1. This kind of alignment does not contribute weight to the partition function.

Next we prove the correctness of equation 3.11 for partition function.

Proposition 3. The equation 3.11 correctly computes the partition function associated with each state in $\mathcal{Q}$.

Proof. We will prove it by induction. Recalling that $\mathcal{S}(X, A_{X\uparrow})$ is the search space for partial alignments between the subset of positions appearing in the descendants of X in the tree decomposition and W ending with $A_{X\uparrow}$. We want to prove that for a given state $(X, A_{X\uparrow})$, $\mathcal{Z}(X, A_{X\uparrow}) = \sum_{A \in \mathcal{S}(X, A_{X\uparrow})} e^{-\beta C(A)}$.

Base case. X is the leaf bag. The sole weight contributes to an alignment A in $\mathcal{S}(X, A_{X\uparrow})$ is the term $e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})}$. As we have already restricted $A_{X\uparrow}$ in an

alignment A, to find a different alignment A', we have to consider different alignment triples for positions in proper indices $X\downarrow$. Therefore, one has

$$\mathcal{Z}(X, A_{X\uparrow}) = \sum_{A \in \mathcal{S}(X, A_{X\uparrow})} e^{-\beta C(A)} = \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})}$$

where $\forall A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$, $A_X = A_{X\uparrow} \cup A_{X\downarrow}$ is a valid alignment.

Induction hypothesis. Suppose X is not a leaf bag and the claim holds for all bags with fewer descendants.

Induction argument. Recall that X'_i with $i \leq c \coloneqq |child(X)|$ is one child of X in the tree decomposition. Any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$ can be decomposed into several parts $A = A_{X\uparrow} \cup A_{X\downarrow} \cup A'_1 \cup \ldots \cup A'_c$ where $A'_i \in \mathcal{S}(X'_i, B_i)$. Therefore, the alignments in the search space $\mathcal{S}(X, A_{X\uparrow})$ can be first decomposed according to the alignment triple $A_{X\downarrow}$.

$$
\begin{aligned}
\mathcal{Z}(X, A_{X\uparrow}) &= \sum_{A \in \mathcal{S}(X, A_{X\uparrow})} e^{-\beta C(A)} \\
&= \sum_{\substack{A = A_{X\uparrow} \cup A_{X\downarrow} \cup A'_1 \cup \ldots \cup A'_c \\ A'_i \in \mathcal{S}(X'_i, B_i) \\ A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})}} e^{-\beta(LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{i=1}^{c} C(A'_i))} \\
&= \sum_{\substack{A_{X\downarrow} \\ \text{s.t. } A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})}} \sum_{\substack{A'_1 \\ \text{s.t. } A'_1 \in \mathcal{S}(X'_1, B_1)}} \cdots \sum_{\substack{A'_c \\ \text{s.t.} A'_c \in \mathcal{S}(X'_c, B_1)}} \\
&\quad \sum_{A = A_{X\uparrow} \cup A_{X\downarrow} \cup A'_1 \ldots A'_c} e^{-\beta(LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{i=1}^{c}(A'_i))} \\
&= \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \sum_{A'_1 \in \mathcal{S}(X'_1, B_1)} e^{-\beta C(A'_1)} \\
&\cdot \ldots \cdot \sum_{A'_c \in \mathcal{S}(X'_c, B_c)} e^{-\beta C(A'_c)} \\
&= \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \sum_{A'_1 \in \mathcal{S}(X'_1, B_1)} e^{-\beta C(A'_1)} \\
&\cdot \ldots \cdot \sum_{A'_{c-1} \in \mathcal{S}(X'_{c-1}, B_{c-1})} e^{-\beta C(A'_{c-1})} \cdot \mathcal{Z}(X'_c, B_c) \\
&= \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \sum_{A'_1 \in \mathcal{S}(X'_1, B_1)} e^{-\beta C(A'_1)} \\
&\cdot \ldots \cdot \mathcal{Z}(X'_c, B_c) \cdot \sum_{A'_{c-1} \in \mathcal{S}(X'_{c-1}, B_{c-1})} e^{-\beta C(A'_{c-1})} \\
&= \sum_{A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})} e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \prod_{i=1}^{c} \mathcal{Z}(X'_i, B_c)
\end{aligned}
\tag{3.12}
$$

□

The complexity of computing the partition function is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ where $|\mathcal{X}|$ is the number of bags in the tree decomposition, tw is the tree-width of tree decomposition and $n = |W|$.

3.2.4 Stochastic backtrack algorithm

One application of partition function is to generate the optimal and suboptimal alignments using the stochastic backtracking algorithm. Due to the fact that Rinaudo's

algorithm is unambiguous and complete, we can use the stochastic backtracking algorithm to randomly generate structure-sequence alignment under the Boltzmann conditional probability distribution. The algorithm starts at state $(X_0, \emptyset)$ and following the inverting ordering of DP equation. That is to say, starting from start state $(X_0, \emptyset)$, choosing one of the alignment set $A_{X\downarrow}$ under the context $A_{X\uparrow}$ according to the probability which is proportional to the weight contribution of $\mathcal{Z}(X, A_{X\uparrow})$. After such a step, we know alignment set $A_{X\downarrow}$ and we move to the children states $(X'_1, project(X'_1, A_X))$,..., $(X'_c, project(X'_c, A_X))$. The backtracking procedure is finished when all the bags are visited once according to the inverting ordering of DP equation. The process of stochastic backtracking algorithm is shown here,

Procedure Stochastic-Backtracking

Let $X_0, \ldots, X_{n-1}$ be a inverting topological ordering of the bags, X_0 is the root bag;
A=$\emptyset$;
for $i \leftarrow 0 \ldots n-1$ **do**
  $A_{X_i\uparrow} \leftarrow project(X_i, A)$;
  sample alignment set $A_{X_i\downarrow}$ in the context $A_{X_i\uparrow}$ based on $\mathcal{Z}(X_i, A_{X_i\uparrow})$;
  $A \leftarrow A_{X_i\downarrow}$;
end
return A

As shown in the algorithm, the partition function $\mathcal{Z}(X, A_{X\uparrow})$ for partial alignment has already been calculated and we already know the alignment set $A_{X\uparrow}$. Each $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$ with the valid derivation $(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} (X'_1, B_1), \ldots (X'_c, B_c)$ is a potential choice. For each derivation, the corresponding conditional probability based on the partition function $\mathcal{Z}(X, A_{X\uparrow})$ is calculated by:

$$p_i = \frac{e^{-\beta LCost(A_{X\uparrow}, A^i_{X\downarrow})} \cdot \prod_{X' \in child(X)} \mathcal{Z}(X', project(X', A_X))}{\mathcal{Z}(X, A_{X\uparrow})} \tag{3.13}$$

Proposition 4. The stochastic backtracking algorithm samples an alignment A in the search space S under the Boltzmann distribution.

Proof. We will prove it by induction. Recalling that $\mathcal{S}(X, A_{X\uparrow})$ is the search space for partial alignments between the subset of positions appearing in the descendants of X in the tree decomposition and W ending with $A_{X\uparrow}$. We want to prove that for

a given state $(X, A_{X\uparrow})$, the probability for an alignment A in search space $\mathcal{S}(X, A_{X\uparrow})$ is $\mathbb{P}(A) = \frac{e^{-\beta C(A)}}{\mathcal{Z}(X, A_{X\uparrow})}$.

Base case. X is a leaf bag in tree decomposition. For any state $(X, A_{X\uparrow})$, an alignment $A \in \mathcal{S}(X, A_{X\uparrow})$ and $A = A_X = A_{X\uparrow} \cup A_{X\downarrow}$, the generating probability is

$$\frac{e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})}}{\mathcal{Z}(X, A_{X\uparrow})} = \frac{e^{-\beta C(A)}}{\mathcal{Z}(X, A_{X\uparrow})}$$

where $A_{X\downarrow} \subseteq A$.

Induction hypothesis. Given each child bag X and any particular $A_{X\uparrow} \in \mathcal{A}_{X\uparrow}$, any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$, the probability of the alignment is $\mathbb{P}(A) = \frac{e^{-\beta C(A)}}{\mathcal{Z}(X, A_{X\uparrow})}$.

Induction argument. Recall that X'_i with $i \leq c \coloneqq |child(X)|$ is one child of X in the tree decomposition. Any alignment $A \in \mathcal{S}(X, A_{X\uparrow})$ can be decomposed into several parts $A = A_{X\uparrow} \cup A_{X\downarrow} \cup A'_1 \cup \ldots \cup A'_c$ where $A'_i \in \mathcal{S}(X'_i, B_i)$. The induction hypothesis ensures that our algorithm generates partial alignment A'_i for child bag X'_i with Boltzmann probability. Moreover, we have

$$\begin{aligned} &\frac{e^{-\beta LCost(A_{X\uparrow}, A_{X\downarrow})} \cdot \prod_{i=i}^{c=|child(X)|} \mathcal{Z}(X'_i, B_i)}{\mathcal{Z}(X, A_{X\uparrow})} \\ &\cdot \frac{e^{-\beta C(A'_1)}}{\mathcal{Z}(X'_1, B_1)} \cdot \ldots \cdot \frac{e^{-\beta C(A'_c)}}{\mathcal{Z}(X'_c, B_c)} = \frac{e^{-\beta C(A)}}{\mathcal{Z}(X, A_{X\uparrow})} \end{aligned} \tag{3.14}$$

This completes the proof.

□

The complexity of the forward step is the same as Rinaudo's algorithm which is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ where $|\mathcal{X}|$ is the number of bags in the tree decomposition, tw is the tree-width of tree decomposition and $n = |W|$. In the backward step, the algorithm visits each bag once and each time considers all the $A_{X\downarrow}$ with the fact that the corresponding derivation $(X, A_{X\uparrow}) \xrightarrow[s]{A_{X\downarrow}} (X'_1, B_1), \ldots (X'_c, B_c)$ is a valid derivation. Besides, we can always get the tree decomposition with $|A_{X\downarrow}| = 1$ for all the bags. Furthermore, for the ease of comparing the complexity of K-best alignment algorithm in the further section, we generate K-stochastic alignment. Therefore, the complexity for stochastic backtracking algorithm is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw} + K \cdot |\mathcal{X}| \cdot n)$.

3.2.5 Inside-outside algorithm

Using the partition function, one can also compute the match probability between Q and W. $\mathbb{P}(S_Q[i] \sim S_W[p])$ ($\mathbb{P}(i \sim p)$ for short) represents the **Boltzmann match probability** that $S_Q[i]$ aligns to $S_W[p]$ with matched case given that i is in the single-stranded region. Suppose the set $\mathcal{S}(i,p) = \{A | a = (i,p,1) \in A \text{ and } a \in \hat{\mathscr{A}}_{m,n}\}$ represents all the possible alignments where $S_Q[i]$ aligns to $S_W[p]$ with matched case in the search space $\mathcal{S}$. Then the probability $\mathbb{P}(i \sim p)$ is calculated by

$$\mathbb{P}(i \sim p) = \frac{\sum\limits_{A \in \mathcal{S}(i,p)} e^{-\beta C(A)}}{\mathcal{Z}} \tag{3.15}$$

Similarly, $Prob((i,j) \sim (p,q))$ represents the base pair Boltzmann match probability that $S_Q[i]$ aligns to $S_W[p]$ with matched case, $S_Q[j]$ aligns to $S_W[q]$ with matched case given that nucleotides $S_Q[i]$, $S_Q[j]$ form base pair. $\mathcal{S}((i,j) \sim (p,q))$ represents all the corresponding possible alignments. Then

$$\mathbb{P}((i,j) \sim (p,q)) = \frac{\sum\limits_{A \in \mathcal{S}((i,j) \sim (p,q))} e^{-\beta C(A)}}{\mathcal{Z}} \tag{3.16}$$

Here one can use the inside-outside algorithm to calculate the match probability following the idea in [75]. In [75], Ponty *et al.* pointed out the notion of derivations can be formalized as hyperpaths in a directed hypergraph and calculated the probability of each hyper-edge using the inside-outside algorithm. To simplify the expression of the algorithm, we also assume that for each bag $X \in \mathcal{X}$, $|A_{X\downarrow}| = 1$ without loss of generality. The inside algorithm follows DP recursion 3.11 to compute the partition function where $\mathcal{Z}(X, A_{X\uparrow})$ represents the partition function for the partial alignment from the leaf bags ending with $A_{X\uparrow}$. Unlike inside algorithm traversing the tree decomposition from the leaf bags to the root bag, outside algorithm calculates from the root bag to the leaf bags. We use $\hat{\mathcal{Z}}(X, A_{X\uparrow})$ to represent the partition function for the partial alignment from the root bag ending with $A_{X\uparrow}$. The difference is shown in Figure 3.6 C).

Now we need to consider how to calculate $\hat{\mathcal{Z}}(X, A_{X\uparrow})$. We first see an example as shown in Figure 3.6 D). The current bag is $X = \{2,4,6,7\}$ and we consider a particular state $(X, A_{X\uparrow})$ where $A_{X\uparrow} = \{(4,6,1),(6,8,1),(7,9,1)\}$. The father bag is $p(X) = \{4,6,7,9\}$ and alignment for the father bag is $A_{p(X)} = \{(4,6,1),(6,8,1),(7,9,1),(9,?,?)\}$. The alignment triple for position 9 can be $(9,10,1)$, $(9,12,1)$, etc. if

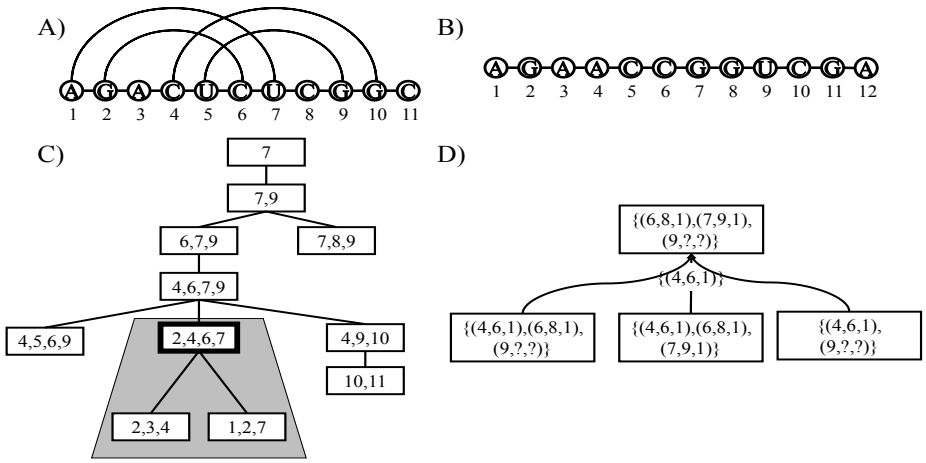


Figure 3.6: Illustration of the inside-outside algorithm **(A)** An arc annotated-sequence sequence Q. **(B)** A plain sequence W. **(C)** Illustration of $\mathcal{Z}(X, A_{X\uparrow})$ and $\hat{\mathcal{Z}}(X, A_{X\uparrow})$ for partial alignment where $X = \{2, 4, 6, 7\}$. $\mathcal{Z}(X, A_{X\uparrow})$ shows the partition function for the partial alignment between the positions appearing in the descendants (shown in grey region) of X and W. $\hat{\mathcal{Z}}(X, A_{X\uparrow})$ represents the partition function for partial alignment between the positions in the other bags (outside the grey regions) and W. **(D)** Illustration of the equation for calculating $\hat{\mathcal{Z}}(X, A_{X\uparrow})$

$A_{p(X)}$ is a valid bag alignment. Therefore we need to consider all possible valid bag alignments $A_{p(X)}$ for father bag. Then, for the particular state $(X, A_{X\uparrow})$, we have

$$\begin{aligned}\hat{\mathcal{Z}}(X, A_{X\uparrow}) = & \sum_{\substack{A_{p(X)} \\ \text{s.t. } A_{X\uparrow}=project(X,A_{p(X)})}} \hat{\mathcal{Z}}(p(X), A_{p(X)\uparrow}) \\ & \cdot e^{-\beta LCost(A_{p(X)\uparrow}, A_{p(X)\downarrow})} \\ & \cdot \prod_{\substack{X' \in child(p(X)) \\ s.t. X' \neq X}} \mathcal{Z}(X', project(X', A_{p(X)})) \end{aligned} \tag{3.17}$$

Again, we need to consider the boundary cases. (1) If the current bag X is the root bag in the tree decomposition, then $p(X) = \emptyset$, then the term $\hat{\mathcal{Z}}(p(X), A_{p(X)\uparrow})$ is set to 1 in equation 3.17. (2) If the father bag alignment is not a valid alignment, then the term $e^{-\beta LCost(A_{p(X)\uparrow}, A_{p(X)\downarrow})}$ is set to 0. Following the idea in [75], we can calculate the probability for each hyperedge in the derivation hyperpath. However, our aim is to calculate the base match/mismatch probability $\mathbb{P}(i \sim p)$ and base pair match/mismatch probability $\mathbb{P}((i, j) \sim (p, q))$. To calculate $\mathbb{P}((i, j) \sim (p, q))$, we need to focus on the bag X where $(i, j) \in P(X)$ and to calculate $\mathbb{P}(i \sim p)$, we need to focus on the bag X where $i \in X\downarrow$. In this example, we labeled the bags that used to

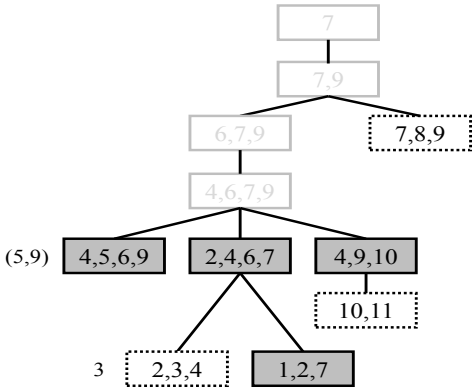


Figure 3.7: Illustration of the bags needed to calculate match/mismatch probability. The bag with grey background is to calculate base pair match/mismatch probability for example, the bag $\{4,5,6,9\}$ is to calculate $\mathbb{P}((5,9) \sim (p,q))$. The bags within dotted rectangle is to calculate base match/mismatch probability. For example, the bag $\{2,3,4\}$ is to calculate $\mathbb{P}(3,p)$.

calculate the match/mismatch probability as shown in Figure 3.7. For example, the bag $\{4,5,6,9\}$ is used to calculate $\mathbb{P}((5,9) \sim (p,q))$.

To calculate $\mathbb{P}((i,j) \sim (p,q))$, the first step is to locate the bag X where $(i,j) \in P(X)$ in the tree decomposition. A_X is denoted as the alignment of the bag X, then

$$\mathbb{P}((i,j) \sim (p,q)) = \sum_{\substack{A_X \\ s.t.(i,p,1)\in A_X \\ (j,q,1)\in A_X}} e^{-\beta LCost(project(X\uparrow,A_X),project(X\downarrow,A_X))} \cdot \frac{\hat{\mathcal{Z}}(X, project(X\uparrow, A_X)) \prod_{X'\in child(X)} \mathcal{Z}(X', project(X', A_X))}{\mathcal{Z}} \tag{3.18}$$

To calculate $\mathbb{P}(i \sim p)$, The first step is also to locate the X where $i \in X\downarrow$. Then we use a similar equation 3.18 to calculate the base match/mismatch probability.

$$\mathbb{P}(i \sim p) = \sum_{\substack{A_X \\ s.t.(i,p,1)\in A_X}} e^{-\beta LCost(project(X\uparrow,A_X),project(X\downarrow,A_X))} \cdot \frac{\hat{\mathcal{Z}}(X, project(X\uparrow, A_X)) \prod_{X'\in child(X)} \mathcal{Z}(X', project(X', A_X))}{\mathcal{Z}} \tag{3.19}$$

Equations 3.11 and 3.17 have the same complexity as Rinaudo's algorithm which is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ where $|\mathcal{X}|$ is the number of bags in the tree decomposition, tw is

the tree-width of the tree decomposition and $n = |W|$. To calculate $\mathbb{P}(i \sim p)$ or $\mathbb{P}((i,j) \sim (p,q))$, one first needs to focus on the particular bags X and consider all valid bag alignments that satisfy certain constraints. The time complexity of this step is $\mathcal{O}(n^{tw+1})$. Therefore, the overall complexity is $\mathcal{O}(\max(|\mathcal{X}|, n) \cdot n^{tw})$.

3.2.6 Maximum expected accuracy alignment

The former cost function gives an optimal alignment, or in other words, highest probability alignment. An alternative approach is to compute maximum expected accuracy alignment (MEA) [28, 45]. To compute the pairwise MEA alignment, we need two steps: (1) Compute the posterior probability for bases and base pairs. (2) Run Rinaudo's algorithm using these posterior probabilities as substitution score and no gap penalties to get the pairwise MEA alignment.

The first step is to compute the posterior probabilities. Though the inside-outside algorithm can calculate the exact base match probability and base pair match probability, they cost too much time. Another choice is to use stochastic backtracking procedure to gather 1000 alignment samplings, and based on the samplings, one can calculate the posterior probabilities of the match matrices between the two sequences $Q : (S_Q, P)$ and $W(S_W, \emptyset)$. In general, for a base index i in the single-stranded region of Q, one match matrix $\mathbb{P}(i,p)$ will be calculated and three matrices $\mathbb{P}((i,j) \sim (p,q))$, $\mathbb{P}((i,j) \sim (-,q))$ and $\mathbb{P}((i,j) \sim (p,-))$ should be calculated for a base pair $(i,j) \in P$.

The second step is to execute Rinaudo's algorithm and before that we need the objective function to calculate MEA alignment. Consider A^* to be the *true* alignment between Q and W. Then the accuracy of an alignment A with A^* is the number of same aligned pairs of nucleotides, divided by the length of the shorter sequence.

$$\begin{aligned} acc(A, A^*) &= \frac{1}{\min(m,n)}(A \cap A^*) \\ &= \frac{1}{\min(m,n)} \sum_{i \sim p \in A} 1\{i \sim p \in A^*\} \end{aligned}$$

where $m = |Q|$ and $n = |W|$. There are four cases for $S_Q[i]$. One is $S_Q[i]$ is free and $S_Q[i]$ aligns to $S_W[p]$. One is $S_Q[i]$ forms base pair with $S_W[j]$ and $S_Q[i]$, $S_Q[j]$ aligns to $S_W[p]$, $S_W[q]$, respectively. One is $S_Q[i]$ forms base pair with $S_W[j]$ and only $S_Q[i]$ aligns to $S_W[p]$. The last one is One is $S_Q[i]$ forms base pair with $S_W[j]$ and only

$S_Q[j]$ aligns to $S_W[q]$. Then

$$acc(A, A^*) = \frac{1}{\min(m,n)}\Big(\sum_{\substack{(i\sim p)\in A\\ \text{i unpaired}}} 1\{i \sim p \in A^*\} + \sum_{\substack{(i,j)\sim(p,q)\in A\\ (i,j)\in P}} 2\{(i,j)\sim(p,q)\in A^*\}$$
$$\sum_{\substack{(i,j)\sim(p,-)\in A\\ (i,j)\in P}} 1\{(i,j)\sim(p,-)\in A^*\} + \sum_{\substack{(i,j)\sim(-,q)\in A\\ (i,j)\in P}} 1\{(i,j)\sim(-,q)\in A^*\}\Big) \tag{3.20}$$

Since A^* is not known, it is not possible to compute the accuracy in a predictive context. However, we can calculate the expected accuracy by considering all possible alignments between Q and W as shown in [28]. By using partition function, we can calculate the probability for every possible alignment in search space S and the probability for a particular alignment $\tilde{A}$ is denoted by $\mathbb{P}(\tilde{A}|Q,W)$. Therefore, the expected accuracy of alignment A is

$$E(acc(A, A^*)) = \frac{1}{\min(m,n)}\sum_{\tilde{A}\in S}\mathbb{P}(\tilde{A}|Q,W)\cdot\Big(\sum_{\substack{i\sim p\in A\\ \text{i unpaired}}} 1\{i\sim p\in\tilde{A}\}$$
$$+\sum_{\substack{(i,j)\sim(p,q)\in A\\ (i,j)\in P}} 2\{(i,j)\sim(p,q)\in\tilde{A}\} + \sum_{\substack{(i,j)\sim(p,-)\in A\\ (i,j)\in P}} 1\{(i,j)\sim(p,-)\in\tilde{A}\}$$
$$+\sum_{\substack{(i,j)\sim(-,q)\in A\\ (i,j)\in P}} 1\{(i,j)\sim(-,q)\in\tilde{A}\}\Big) \tag{3.21}$$

Then the objective function of MEA for a particular alignment $A \in \hat{\mathscr{A}}_{m,n}$ is

$$E(acc(A)) = \frac{1}{\min(m,n)}\Big(2\gamma\cdot\sum_{(i,j)\sim(p,q)\in A}\mathbb{P}((i,j)\sim(p,q))$$
$$+\sum_{(i,j)\sim(-,q)\in A}\mathbb{P}((i,j)\sim(-,q))$$
$$+\sum_{(i,j)\sim(p,-)\in A}\mathbb{P}((i,j)\sim(p,-))$$
$$+\sum_{\substack{i\sim p\in A\\ i\text{ unpaired}}}\mathbb{P}(i\sim p)\Big) \tag{3.22}$$

where $\gamma > 0$ is a parameter with default value 1.

3.3 Enumerating suboptimal alignments

The concept of near optimal or suboptimal alignment comes into play for one single reason, the best alignment or the set of optimal alignments may not reflect biological processes. Comparing with stochastic backtracking procedure which can generate duplicate alignments as mentioned before, the algorithms that we will introduce is used to enumerate distinct near-optimal alignments. One algorithm to enumerate Δ near-optimal alignments and the other algorithm is to enumerate K-best near-optimal alignments.

The idea of generating Δ near-optimal alignment algorithm is from the work of Waterman and Byers in [106]. They first designed the algorithm to find suboptimal paths for shortest path problem and they extended the algorithm to find suboptimal sequence alignments within a set range from optimal alignment. The idea is to treat the derivation trees or alignments in DP matrix as $(s, t)-$ paths in a directed acyclic graph (DAG) with weights on its edges because of the linearity of the recursion function. In [118], Wuchty *et al.* extended this algorithm to enumerate suboptimal RNA folding within a range δ from the minimum free energy folding.

While conceptually very close, the generation of K-best near-optimal alignments requires considerably more involved strategies in order to achieve a reasonable complexity, as shown in Huang *et al.* [46]. In [46], Huang *et al.* formalized Viterbi algorithm (dynamic programming) under the directed hypergraph framework and gave an efficient algorithm for computing K best parsing. Moreover, Becker *et al.* [5] have implemented this algorithm to generate sub-optimal HMM (Hidden Markov Model) alignments. In [75], Ponty *et al.* pointed out the derivation from DP matrix can be formalized as hyperpaths in a directed hypergraph for RNA folding problem.

Before showing the Δ near-optimal alignment algorithm and K-best near-optimal alignment algorithm, it is necessary to first explain the backtracking procedure which is a key step to understand these two near-optimal alignment algorithms. As we have defined previously, $M(X, A_{X\uparrow})$ is the optimal partial alignment between the positions appearing in the descendants of X and W ending with $A_{X\uparrow}$ and the corresponding optimal score is $C(X, A_{X\uparrow})$. For understanding the further algorithms, two new notations are introduced. One is ξ which is a stack of state $(X, A_{X\uparrow})$. The other one is A which is a set of triplets and in other words, it stores the information about partial alignment. The normal backtracking procedure is sketched in Algorithm 1.

Algorithm 1: Backtracking Procedure($C(X, A_{X\uparrow})$)

$\xi = \{(X_0, A_{X_0\uparrow})\}$ and $A = \emptyset$;
while $\xi \neq \emptyset$ **do**
  $(X, A_{X\uparrow}) \leftarrow \mathsf{pop}(\xi)$;
  if ($child(X) = \emptyset$ *and* $A_{X\downarrow}$ *with* $C(X, A_{X\uparrow}) = LCost(A_{X\uparrow}, A_{X\downarrow})$) **then**
    $A \leftarrow A_{X\downarrow}$;
  end
  if $\left(\begin{array}{l} child(X) \neq \emptyset \text{ and } A_{X\downarrow} with \\ C(X, A_{X\uparrow}) = LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', B)) \end{array} \right)$ **then**
    /* $B = project(X', A_{X\uparrow} \cup A_{X\downarrow})$ */
    $A \leftarrow A_{X\downarrow}$;
    foreach $X' \in child(X)$ **do**
      $\mathsf{push}((X', B)) \rightarrow \xi$;
    end
  end
end
return A

The backtracking procedure starts with the start state $(X_0, \emptyset)$ which is stored in a stack ξ and an empty set A. Pop the state from stack ξ and assign the first element of the state to X and the second element to $A_{X\uparrow}$. Following equation 2.12, find the alignment set for $X \downarrow$ that is consistent with $C(X, A_{X\uparrow})$. If X is a leaf bag, then we only need to push the alignment set $A_{X\downarrow}$ into A. If X is not a leaf bag, we also need to push the child state (X', B) where $B = project(X', A_{X\uparrow} \cup A_{X\downarrow})$ into stack ξ. The procedure continues until the stack ξ is empty and a complete optimal alignment between Q and W is constructed in A.

3.3.1 Δ near-optimal alignment

As mentioned before, the idea behind the algorithm for solving Δ near-optimal alignment is from the work of Waterman and Byers in [107] and Wuchty *et al.* [118]. Given the parameter Δ defined by the user, our aim is to find all the near-optimal alignments with the costs less than $C_{\min} + \Delta$ where $C_{\min} = C(X_0, \emptyset)$ is the minimum score. In other words, the Δ near-optimal algorithm will generate every near-optimal alignment in the Δ neighborhood.

In Δ near-optimal alignment algorithm, besides the stack ξ containing the information

about states $(X, A_{X\uparrow})$ in backtracking procedure, a new stack η is defined. Each element of η contains three items of information (ξ, A, δ): (1) the state stack ξ; (2) the alignment set A; (3) the current distance δ with $C_{\min}$ derived from the start state $(X_0, \emptyset)$. The Δ near-optimal alignment is shown in Algorithm 2.

As the backtracking procedure, the algorithm also begins with the start state $(X_0, \emptyset)$. Then pop one element (ξ, A, δ) from η. Assign the first element of ξ to X and the second element of ξ to $A_{X\uparrow}$. Instead of only finding the optimal $A_{X\downarrow}$ that satisfies the equation shown below,

$$C(X, A_{X\uparrow}) = LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', B)$$

where $B = project(X', A_{X\uparrow} \cup A_{X\downarrow})$, we need to find all the $A_{X\downarrow}$ that satisfies

$$\delta' = LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', B) + \delta - C(X, A_{X\uparrow}) \leq \Delta \tag{3.23}$$

It means that if δ' with $A_{X\downarrow}$ is greater than Δ, A with $A_{X\downarrow}$ inserted will not be considered any more and if δ' with $A_{X\downarrow}$ is less than Δ, after some operations to ξ and A, (ξ, A, δ') will be pushed into the stack η for further calculation. When the stack η is empty, it means that we have found all the Δ near-optimal alignments.

Proposition 5. The score of the selected alignment A satisfying equation 3.23 is less than $C_{\min} + \delta$.

Proof. I will give a brief proof based on the popping element (ξ, A, δ). The state $(X,\ A_{X\uparrow})$ is the popping state from ξ. We define the score of the popping alignment A from start state $(X_0, \emptyset)$ to the state $(X, A_{X\uparrow})$ as $\bar{C}(X, A_{X\uparrow})$. Then we have the equation

$$\bar{C}(X, A_{X\uparrow}) + C(X, A_{X\uparrow}) + \sum_{(X', A_{X'\uparrow}) \in \xi} C(X', A_{X'\uparrow}) - C_{min} = \delta \tag{3.24}$$

Then we put equation 3.24 into equation 3.23, for $A_{X\downarrow}$ satisfying equation 3.23, we will have

$$\bar{C}(X, A_{X\uparrow}) + \sum_{(X', A_{X'\uparrow})\in\xi} C(X', A_{X'\uparrow}) + LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X'\in child(X_i)} C(X', B) - C_{\min} \leq \Delta. \tag{3.25}$$

This completes the proof. □

The complexity of Rinaudo's algorithm is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ where $|\mathcal{X}|$ is the number of bags in the tree decomposition, tw is the tree-width of the tree decomposition and $n = |W|$ is the length of the PLAIN sequence. For a generated alignment, Algorithm 2 visits each bag once and each time consider all the $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$. Besides we always get the tree decomposition with $|A_{X\downarrow}| = 1$ for all bags without loss of generality. Furthermore, we assume that the number of alignments within range δ from the optimal alignments is exactly K. Therefore, the complexity for Algorithm 2 is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw} + K \cdot |\mathcal{X}| \cdot n)$.

When $\Delta = 0$, the Δ near-optimal algorithm will produce all the optimal alignments. If Δ is set big enough, the algorithm will generate all the alignments in the search space. However, this is not recommended, because it will take quite a long time.

3.3.2 K-best suboptimal alignment

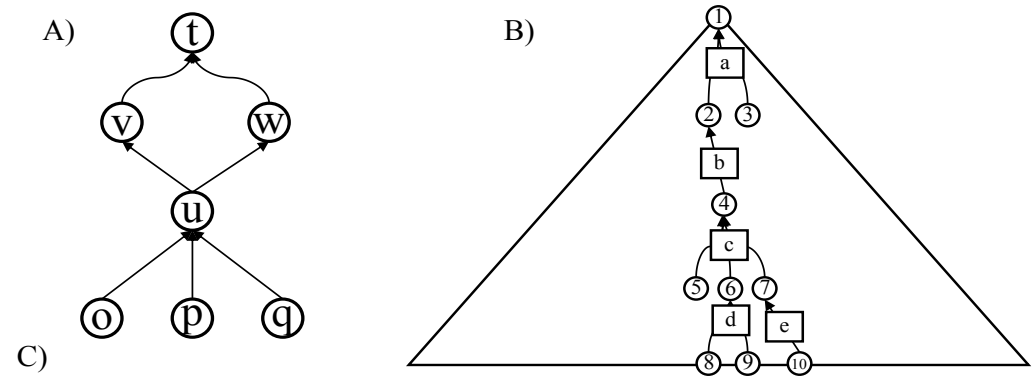
The idea behind this algorithm for solving K-best near-optimal is introduced in the work of Huang *et al.* [46] and the K-best alignment problem can be formulated as:

Definition 10 (K **best Structure-Sequence Alignment Problem**)**.** Given an arc-annotated sequence Q, a plain sequence W and a positive integer K, the K best structure-sequence alignment problem is to find K best alignments between Q and W ordered by cost.

Now we have changed our strategy. Instead of requiring Δ as input, we ask the user to input the parameter K. To understand the algorithm, we need to explicitly model derivation trees of Rinaudo's algorithm to hyperpaths in an ordered hypergraph.

Algorithm 2: Δ-suboptimal alignment$(C(X, A_{X\uparrow})\ )$

push $(\{(X_0, A_{X_0\uparrow})\}, \emptyset, 0)$ into stack η ;
while $\eta \neq \emptyset$ **do**
  $(\xi, A, \delta) \leftarrow \mathsf{pop}(\eta)$;
  if $\xi = \emptyset$ **then**
    Output suboptimal alignment and continue ;
  end
  $(X, A_{X\uparrow}) \leftarrow \mathsf{pop}(\xi)$;
  if $child(X) = \emptyset$ **then**
    foreach $A_{X\downarrow}$ *within context* $A_{X\uparrow}$ **do**
      $\delta' = LCost(A_{X\uparrow}, A_{X\downarrow}) - C(X, A_{X\uparrow}) + \delta$;
      if $\delta' \leq \Delta$ **then**
        $A \leftarrow A_{X\downarrow}$;
        $\mathsf{push}((\xi, A, \delta')) \rightarrow \eta$;
      end
    end
  end
  if $child(X) \neq \emptyset$ **then**
    foreach $A_{X\downarrow}$ *within context* $A_{X\uparrow}$ **do**
      $\delta' = LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', B) - C(X, A_{X\uparrow}) + \delta$;
      /* $B = project(X', A_{X\uparrow} \cup A_{X\downarrow})$ */
      if $\delta' \leq \Delta$ **then**
        $A \leftarrow A_{X\downarrow}$;
        foreach $X' \in child(X)$ **do**
          $\mathsf{push}((X', B)) \rightarrow \xi$;
        end
        $\mathsf{push}((\xi, A, \delta')) \rightarrow \eta$;
      end
    end
  end
end



C)

1	2	3	4	5	6	7	8	9	10
{∅, ∅}	{(7,9,1), (9,10,1)}	{(7,9,1), (9,10,1)}	{(6,81), (7,9,1), (9,10,1)}	{(4,6,1), (6,81), (9,10,1)}	{(4,6,1), (6,81), (7,9,1)}	{(4,6,1), (9,10,1)}	{(2,2,1), (4,6,1)}	{(2,2,1), (7,9,1)}	{(10,11,1)}

a	b	c	d	e
{(7,9,1),(9,10,1)}	{(6,8,1)}	{(4,6,1)}	{(2,2,1)}	{(10,11,1)}

Figure 3.8: (A) Illustration of a hypergraph. t is a target vertex and o, p, q are the source vertices. (B) Treating a derivation tree as a hyperpath in the ordered hypergraph. (C) The contents in the circles and rectangles in (B).

Definition 11 (**Ordered hypergraph** [46])**.** An ordered hypergraph $\mathcal{H}$ is a tuple $< V, E, t, \pi >$. Let V be a finite set of vertices and E a finite set of hyperarcs. π is a set of weights. Each hyperedge $e \in E$ is defined as a triple $e =< t(e), h(e), f(e) >$ where $h(e) \in V$ is its head and $t(e) \subseteq V$ is an ordered list of vertices. t is a distinguished vertex called target vertex. For $e \in E$, if $|t(e)| = 0$, then $h(e)$ is called a source vertex. $f : \mathbb{R}^{|t(e)|} \rightarrow \mathbb{R}$ is a weight function.

Figure 3.8 A) shows an example hypergraph. B) illustrates that a derivation tree can be treated as a hyperpath in the hypergraph. A vertex in $\mathcal{H}$ corresponds to a state $(X, A_{X\uparrow})$. Theoretically, there are $(2 \cdot (n+1))^{|X\downarrow|}$ hyperarcs coming into it from the children states $(X', project(X', A_X))$. The weight for each hyperarc is $LCost(A_{X\uparrow}, A_{X\downarrow})$. The source vertex corresponds to the state $(X, A_{X\uparrow})$ where X is a leaf bag and $A_{X\uparrow}$ is alignment set for transition indices of X. The target vertex corresponds to state $(X_0, \emptyset)$ where X_0 is the root bag. As we have proved that Rinaudo's algorithm is unambiguous and complete, there is a one-one correspondence between an instance I in the derivation space $\mathcal{D}$ and an instance I in the search space $\mathcal{S}$. Therefore, suboptimal structure-sequence alignments are in fact the suboptimal hyperpaths in the hypergraph.

To make the algorithm work, we need the hypergraph to be monotonic. Actually, the monotonicity is relevant to the optimal substructure property in dynamic programming. Otherwise, if the hypergraph is not monotonic, we can not get one optimal solution.

Definition 12 (**monotonicity of hypergraph** [46])**.** A hypergraph is monotonic if there is total ordering $\preceq$ on $\mathbb{R}$ such that every weight function $f : \mathbb{R}^{|t(e)|} \to \mathbb{R}$ in $\mathcal{H}$ is monotonic for each of its arguments.

Therefore, for a particular $e \in E$, if $x_i \preceq x_i'$ where $0 \leq i \leq |t(e)|$, $f(x_1, \ldots, x_i, \ldots, x_{|t(e)|})$ $\preceq f(x_1, \ldots, x_i', \ldots, x_{|t(e)|})$. As defined before, the equation to calculate the corresponding score is:

$$LCost(A_{X\uparrow}, A_{X\downarrow}) + \sum_{X' \in child(X)} C(X', project(X', A_X))\}$$

Therefore, for a particular $A_{X\downarrow}$ which corresponds to a particular hyperarc, the cost function f is of the form $a + x_1 + x_2 + \ldots + x_c$ and the function is strictly monotonic in all arguments x_i. That is to say, Rinaudo's algorithm satisfies the monotonicity of the hypergraph.

Another requirement for the hypergraph is acyclic. The hypergraph is **acyclic** if and only if any vertex $v \in V$ is shown once in any derivation tree with target vertex t. The **reason** is that if the hypergraph is acyclic, we can use the topological ordering to traverse the hypergraph. The topological order is obvious in Rinaudo's algorithm. That is to say, we can calculate the score for the partial alignment for the current bag X after calculating the alignments for the child bags $X' \in child(X)$. After knowing all these, we adopt Huang's algorithm [46].

To further present the K-best algorithm, we denote $C^1(X, A_{X\uparrow})$ and $D^1(X, A_{X\uparrow})$ as the optimal score and the corresponding optimal derivation tree from the state $(X, A_{X\uparrow})$. The derivation tree $D^1(X, A_{X\uparrow})$ can be represented as

$$\begin{aligned} &\{\langle A_{X\downarrow}, D^1(X_1', B_1) \ldots D^1(X_c', B_c)\rangle\} \text{ if } child(X) \neq \emptyset \\ \text{or } &\{\langle A_{X\downarrow}, \emptyset\rangle\}, \text{ if } child(X) = \emptyset \end{aligned}$$

Furthermore, $C^r(X, A_{X\uparrow})$ and $D^r(X, A_{X\uparrow})$ are illustrated to represent the r-th best score and the corresponding r-th best derivation tree. Therefore, our aim is to find the $D^1(X_0, \emptyset)$, ..., $D^K(X_0, \emptyset)$ with score $C^1(X_0, \emptyset)$, ..., $C^K(X_0, \emptyset)$. Figure 3.9 illustrates

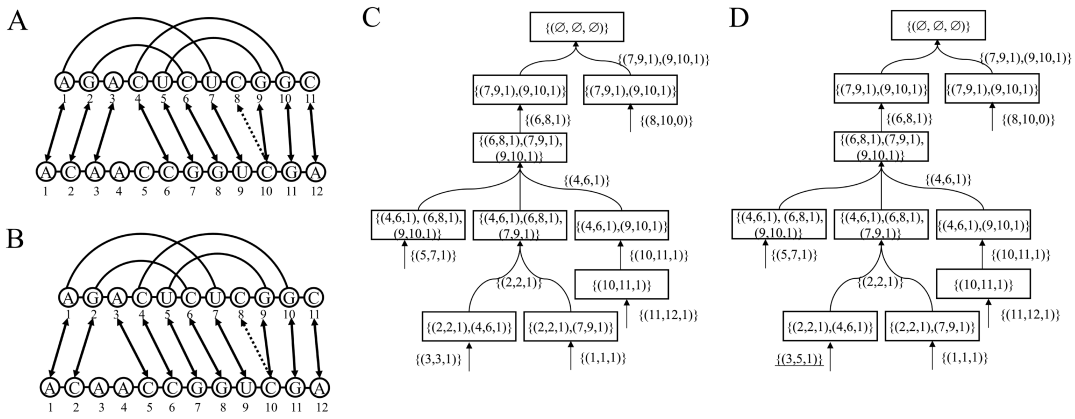


Figure 3.9: Alignments and the corresponding derivation trees (A) 1-best alignment; (B) 2-best alignment; (C) derivation tree producing alignment in (A); (D) derivation tree producing alignment in (B). The only difference between 1-best and 2-best alignment is the alignment triple for the position 3 in Q.

the derivation trees for the 1-best alignment and 2-best alignment for the example sequences Q and W. The only difference between 1-best and 2-best alignment is the alignment triple for the position 3 in Q.

3.3.2.1 Recurrence equation

Let us consider which derivation tree should be considered as the candidate to $D^r(X, A_{X\uparrow})$. Different from $D^1(X, A_{X\uparrow})$, ..., $D^{r-1}(X, A_{X\uparrow})$, the best candidate to $D^r(X, A_{X\uparrow})$ can be chosen in the set

$$\{\langle A_{X\downarrow}, D^{j_1}(X'_1, B_1) \dots D^{j_c}(X'_c, B_c)\rangle \mid A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow}), 1 \leq j_1 \leq r, \dots, 1 \leq j_c \leq r\} \tag{3.26}$$

where $D^{j_1}(X'_1, B_1)$ represents the j_1-best derivation tree from the state (X'_1, B_1) and X'_1 is one child bag and $B_1 = project(X'_1, A_X)$. Obviously, to compute $D^r(X, A_{X\uparrow})$ we do not need to consider the derivation trees for the children states with indices $j_1 > r$, ..., $j_c > r$, because we can always find the r-best derivation tree with $j_1 \leq r, \dots, j_c \leq r$. From the fact that the relationship of the bags in the tree decomposition are fixed, once $A_{X\downarrow}$ is selected, the states for the children bags (X'_1, B_1), ..., (X'_c, B_c) are all fixed. The only thing we care about is indices j_1, ... , j_c. Therefore, to simply the notation, we set $\boldsymbol{j}= (j_1, \dots, j_c) \in \{1, 2, \dots, K\}^c$ and the set of derivation trees

in 3.26 can be reformulated as

$$\{\langle A_{X\downarrow}, \boldsymbol{j}\rangle \mid A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow}), 1 \leq j_1 \leq r, \ldots, 1 \leq j_c \leq r\} \tag{3.27}$$

Therefore, we can use $\langle A_{X\downarrow}, \mathbf{1}\rangle$ instead of $\langle A_{X\downarrow}, D^1(X'_1, B_1) \ldots D^1(X'_c, B_c)\rangle$ to represent the optimal derivation tree. The optimal derivation tree $\langle A_{X\downarrow}, \mathbf{1}\rangle$ has the optimal score $C^1(X, A_{X\uparrow})$.

Furthermore, we define a new set $\mathcal{M}^r(X, A_{X\uparrow})$ where $D^r(X, A_{X\uparrow})$ is the best derivation tree among them. Actually, $\mathcal{M}(X, A_{X\uparrow})$ will be implemented as a priority queue. $D^{r-1}(X, A_{X\uparrow})$ is the best derivation tree in the set $\mathcal{M}^{r-1}(X, A_{X\uparrow})$. $\mathcal{M}^r(X, A_{X\uparrow})$ is derived from $\mathcal{M}^{r-1}(X, A_{X\uparrow})$ by some operations as shown below. We assume that $D^{r-1}(X, A_{X\uparrow})$ corresponds to the derivation tree $\langle A_{X\downarrow}, \boldsymbol{j}\rangle$=$\langle A_{X\downarrow}, j_1, \ldots, j_c\rangle$. A vector $\boldsymbol{e}^i$ is defined whose elements are all 0 except $e^i_i = 1$. We define the **neighbors** of $D^{r-1}(X, A_{X\uparrow})$ to be a set $\{\langle A_{X\downarrow}, \mathbf{j} + \mathbf{e}^i\rangle, \forall i \leq c\}$. According to the monotonicity of the hypergraph, we have the partial order.

$$C(\langle A_{X\downarrow}, \mathbf{j}\rangle) \leq C(\langle A_{X\downarrow}, \mathbf{j} + \mathbf{e}^i\rangle), \forall i \leq c \tag{3.28}$$

This partial order shows the potential search directions and $\mathcal{M}^r(X, A_{X\uparrow})$ can be constructed by deleting $D^{r-1}(X, A_{X\uparrow})$ and inserting its neighbors:

$$\mathcal{M}^r(X, A_{X\uparrow}) = (\mathcal{M}^{r-1}(X, A_{X\uparrow}) - D^{r-1}(X, A_{X\uparrow})) \bigcup \{\langle A_{X\downarrow}, \mathbf{j} + \mathbf{e}^i\rangle \mid \forall i \leq c\} \tag{3.29}$$

After that, the derivation tree in set $\mathcal{M}^r(X, A_{X\uparrow})$ with minimum score would be the r-best derivation tree $D^r(X, A_{X\uparrow})$.

3.3.2.2 Algorithm

The general idea is that *LazyKthBest* assumes an initial DP has been implemented and a DP table has been filled. In this way, we can find the 1-best derivation tree for each cell (state) in the DP table. *LazyKthBest* starts from the root bag with state $(X_0, \emptyset)$ to the leaf bag. For a particular state $(X, A_{X\uparrow})$ in the state space $\mathcal{Q}$, *LazyKthBest* is recursively called only as necessary. Therefore, for that state $(X, A_{X\uparrow})$ we suppose *LazyKthBest* is called r times and in other words, we have found the r-best derivation tree for the particular state $(X, A_{X\uparrow})$. As shown before, $D^r(X, A_{X\uparrow})$ is the best derivation tree in the set $\mathcal{M}^r(X, A_{X\uparrow})$. We also define

$\mathcal{E}(X, A_{X\uparrow}) = \{D^1(X, A_{X\uparrow}), D^2(X, A_{X\uparrow}), \ldots, D^{r-1}(X, A_{X\uparrow})\}$ to store the extracted $r-1$ derivations and $\mathcal{E}(X, A_{X\uparrow})$ is implemented as vector. The following algorithm can find $D^2(X_0, \emptyset)$, ..., $D^K(X_0, \emptyset)$ one after another.

Now we go into details. In procedure *LazyKthBest*, $\mathcal{M}(X, A_{X\uparrow})$ is not defined for every cell (state) in the DP matrix (memory saving) and when it is necessary (*LazyKthBest* visits), it is initialized with K first derivations $< A_{X\downarrow}, \mathbf{1} >$ with different $A_{X\downarrow}$ satisfying $A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})$ ordered by the score as shown in procedure *GetCandidates*. While the number of elements in $\mathcal{E}(X, A_{X\uparrow})$ is less than r which is required by *LazyKthBest*, the last element from $\mathcal{E}(X, A_{X\uparrow})$ which we suppose as $\langle A_{X\downarrow}, \mathbf{j}\rangle$ is obtained and the procedure *LazyNext* is called to insert the neighbors of $\langle A_{X\downarrow}, \mathbf{j}\rangle$ which is $\{\langle A_{X\downarrow}, \mathbf{j} + \mathbf{e}^i | \ 1 \leq i \leq c\}$ into $\mathcal{M}(X, A_{X\uparrow})$. In detail, procedure *LazyNext* ask every child state its j_i'-best derivation tree by calling the procedure *LazyKthBest*. Then this recursive process ends when the current bag is the leaf bag and returns the j_i'-best derivation of the children states upwards. Then we extract the optimal element if it exists in $\mathcal{M}(X, A_{X\uparrow})$ to $\mathcal{E}(X, A_{X\uparrow})$ (*Extrac-Min*). However, if $\mathcal{M}(X, A_{X\uparrow})$ is empty, break the procedure. It means that there is no more other derivation tree coming into state $(X, A_{X\uparrow})$. In other words, all possible derivations tree coming into state $(X, A_{X\uparrow})$ are now in vector $\mathcal{E}(X, A_{X\uparrow})$.

Algorithm 3: Recursive enumeration of the K-best alignments

Compute $D^1(X, A_{X\uparrow})$ for all bags X, for all possible bag alignment sets A_X using Rinaudo's algorithm ;
for $n \coloneqq 2$ **to** K **do**
  `LazyKthBest(`$(X_0, \emptyset)$, K, n`)` ;
end
return $\{D^1(X_0, \emptyset), D^2(X_0, \emptyset), \ldots D^K(X_0, \emptyset)\}$;

The DP algorithm of Rinaudo *et al.* runs in time $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$. Here tw is the tree-width of the tree decomposition T, and W is the plain sequence with length $n = |S_W|$. The computation of the K best alignment tree requires $K \cdot |\mathcal{X}|$ calls to *LazyKthBest*. Each call also needs the operations on $\mathcal{M}(X, A_{X\uparrow})$ which is supported by priority queue, like insertion of new elements and selection/deletion of the best element. Each operation is performed in logarithmic time relative to the number of elements in the priority queue. So the time required by the whole algorithm is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw} + K \cdot |\mathcal{X}| \cdot \log K)$.

Procedure LazyKthBest($(X, A_{X\uparrow})$, K, n)

if $\mathcal{M}(X, A_{X\uparrow})$ *is not defined* **then**
  `GetCandidates(`$(X, A_{X\uparrow})$, K`)`
end
while $|\mathcal{E}(X, A_{X\uparrow})| < n$ **do**
  if $|\mathcal{E}(X, A_{X\uparrow})| > 0$ **then**
    $\langle A_{X\downarrow}, \mathbf{j}\rangle \leftarrow \mathcal{E}_{|\mathcal{E}(X,A_{X\uparrow})|}(X, A_{X\uparrow})$;
    `LazyNext(`$\mathcal{M}(X, A_{X\uparrow})$, $A_{X\downarrow}$, $\boldsymbol{j}$, K`)` ;
  end
  if $|\mathcal{M}(X, A_{X\uparrow})| > 0$ **then**
    append Extrac-Min($\mathcal{M}(X, A_{X\uparrow})$) to $\mathcal{E}(X, A_{X\uparrow})$;
  end
  else
    break
  end
end

Procedure GetCandidates($(X, A_{X\uparrow})$, K)

$temp \leftarrow \{\langle A_{X\downarrow}, \mathbf{1}\rangle \mid \forall A_{X\downarrow} \in \mathcal{A}_{X\downarrow}(X, A_{X\uparrow})\}$;
$\mathcal{M}(X, A_{X\uparrow}) \leftarrow$ top K derivations starting from state $(X, A_{X\uparrow})$ in $temp$;
HEAPIFY($\mathcal{M}(X, A_{X\uparrow})$);

Procedure LazyNext($\mathcal{M}(X, A_{X\uparrow})$, $A_{X\downarrow}$, $\mathbf{j}$, K)

for $i \leftarrow 1 \ldots c$; `//` $c = |child(t)|$
do
  $\mathbf{j}' \leftarrow \mathbf{j} + \mathbf{e}^i$;
  `LazyKthBest(`$(X'_i, B), K, j'_i$`)` ;
  `/*` $B = project(X'_i, A_{X\uparrow} \cup A_{X\downarrow})$ `*/`
  if $j'_i \leq |\mathcal{E}(X'_i, B)|$ *and* $\langle A_{X\downarrow}, \boldsymbol{j}'\rangle$ *is not in* $\mathcal{M}(X, A_{X\uparrow})$ *and* $\mathcal{E}(X, A_{X\uparrow})$ **then**
    Insert $(\mathcal{M}(X, A_{X\uparrow}), \langle A_{X\downarrow}, \mathbf{j}'\rangle)$;
  end
end

Chapter 4

Results

The first experiment mainly answers the following questions: What is the typical tree-width of a pseudoknotted structure, especially in the presence of pseudoknots? Can accurate tree-decomposition be computed in a reasonable time in for RNA graphs? To answer the question, we test the tree decomposition algorithms on PseudoBase and PDB database and the details are shown in section 4.2.

The second experiment mainly answers the following question: How does the predictive capacity of the accuracy of `LiCoRNA` compare with that of other state-of-the-art programs? We test `LiCoRNA` on RFAM pseudoknotted families and compare the performance of `LiCoRNA` with three other state-of-the-art software `PAL`, `PSTAG` and `profile-csHMMs`. The details are shown in section 4.3.

The third experiment mainly answers the following questions: What is the difference between the predicted optimal alignment and the reference alignment in the benchmark? How far down the list of suboptimal alignments can the reference alignment be found? How trustworthy is a structure-sequence alignment prediction? To answer these questions, we generate 1000-best suboptimal alignments using K-best suboptimal alignments and sample an ensemble of alignments using stochastic sampling algorithm when $t = 1.5$. The details are shown in section 4.4 and 4.5.

The fourth experiment mainly answers the following questions: Covariance models do not consider pseudoknots when aligning, which may lead to misalign when building the full alignment. Does `LiCoRNA`'s support of pseudoknots of arbitrary complexity translate into better performances? Conversely, is the lack of support for complex pseudoknots detrimental to the quality of the alignments? We test `LiCoRNA` on the

full alignments for the RFAM pseudoknotted families and we hope that a systematic realignment will allow to reveal or refute an evolutionary pressure towards the preservation of a functional pseudoknot. The details are shown in section 4.6.

Unless specified, runtimes reported in the following correspond to tests that were performed on a PC with CPU Intel(R) Core (TM) i7-4770 and 16 GB RAM, under a Linux Ubuntu 16.04.1 LTS (64bit) operating system.

4.1 Using `LiCoRNA`

`LiCoRNA` is a software for pairwise structure-sequence alignment in RNA featuring pseudoknots of arbitrary complexity. The software consists of 9612 lines of C++, and is freely accessible at:

https://licorna.lri.fr

`LiCoRNA` currently contains three different programs:

- `alignerDP` (Maximum/Suboptimal Parsimony Alignment): implementation of Rinaudo's DP algorithm to compute the optimal structure-sequence alignment. Besides, it can also generate the K-best suboptimal alignments where K is set by users.

```
$ alignerDP
OPTION
-i input path of the file with RNA folded sequence.
-d input path of the file with tree decomposition.
-s input path of the file with target sequence.
-c maximum insertion length.
-n number of suboptimal alignments, the default value is 5
(If the user does not use -n option, n is set to be 5).
```

- `alignerSB` (Stochastic Backtracking): generates a representative sample of alignments according to the Boltzmann conditional probability distribution.

```
$ alignerSB
OPTION
-i input path of the file with RNA folded sequence.
-d input path of the file with tree decomposition.
-s input path of the file with target sequence.
-c maximum insertion length.
-n number of sampling alignments, the default value is 100.
-t temperature.
```

- `alignerMEA` (Maximum Expected Accuracy sequence-structure alignment): computes an alignment with maximum expected accuracy over all alignments that are sampled by the stochastic backtracking procedure.

```
$ alignerMEA
OPTION
-i input path of the file with RNA folded sequence.
-d input path of the file with tree decomposition.
-s input path of the file with target sequence.
-c maximum insertion length.
-t temperature.
```

In program `alignerSB`, the sampling process for each bag can be done quite simply as follows: We generate a probability p' uniformly from the interval $[0, 1]$. Then the interval is partitioned into n sub-intervals: $[0, p_1)$, $[p_1, p_1 + p_2), \ldots$, and each sub-interval corresponds a sample $A^1_{X\downarrow}$, $A^2_{X\downarrow}, \ldots$. If p' is in the ith interval, then the sampled alignment is $A^i_{X\downarrow}$. The uniform distribution generator is mt19937 in library C++11.

Here is an example of an input file for the arc-annotated sequence Q:

```
AGACUCUCGGC
<<.((>>.)).
```

and an example of an input file for plain sequence W:

```
ACAACCGGUCGA
```

There are several points which should be considered about the input file:

- Accepted symbols for sequence are: A, C, G and U;
- Accepted symbols for structure are: ., [and], (and), < and >, { and }. If the four paired symbols are still not enough for you, then the paired symbols with A and a, B and b, C and c, ..., Z and z are also available. Generally, the base pairs with different symbols are crossing;
- All programs support any kind of pseudoknots;
- All programs support Watson-Crick base pairs and G-U base pairs. Non-canonical base-pairs are currently unsupported.

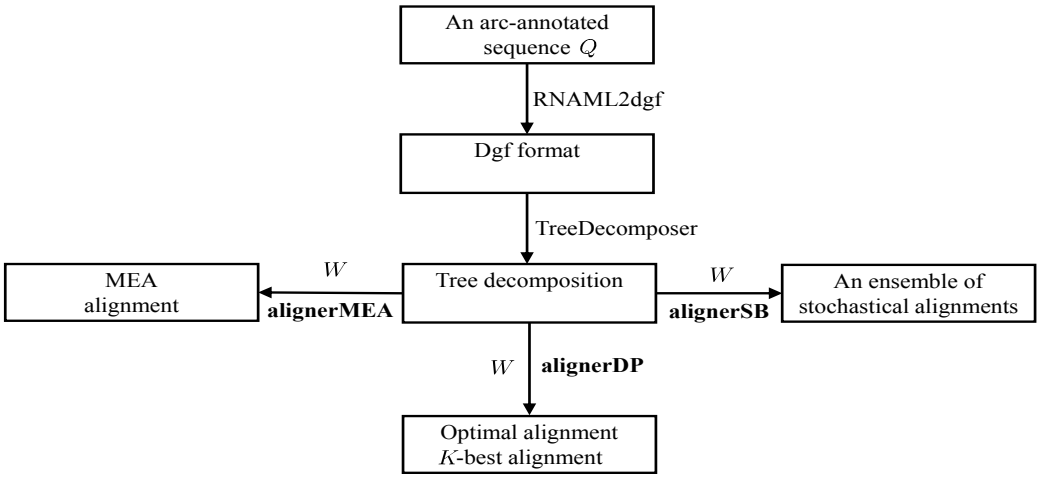


Figure 4.1: The workflow of `LiCoRNA`. The program RNAML2dgf transfers the input file for Q to dgf format. Program TreeDecomposer which is written in JAVA gives the tree decomposition of Q. Then users can use any program alignerMEA, alignerSB and alignerDP according to their needs. W is the input plain sequence.

The workflow for program `LiCoRNA` is shown in Figure 4.1. One more thing to say is that users can also use other tree decomposition algorithms, as long as they format the tree decomposition as the output of `TreeDecomposer` program.

4.2 The tree-width of pseudoknotted RNAs is typically small

The tree-width of a tree decomposition directly affects the complexity of Rinaudo's algorithm as shown on page 33. However, finding the tree decomposition with the minimal tree-width is an NP-hard problem [11, 12]. The current algorithms for computing the exact tree-width are very expensive and only work for specific classes of graphs. Fortunately, many heuristic methods are available, whose performance depends on the type of input data. In order to compare their performance on graphs associated with RNA structures, we benchmarked four tree decomposition algorithms available in our program TreeDecomposer based on the LibTW library [102]:

- GreedyDegree [7];
- GreedyFillin [51];

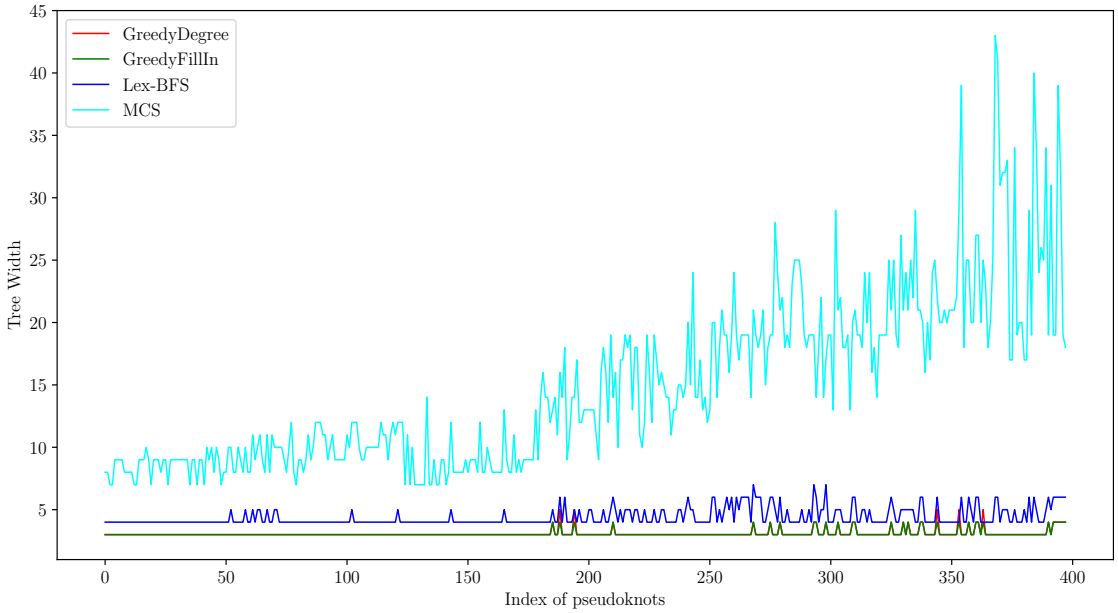


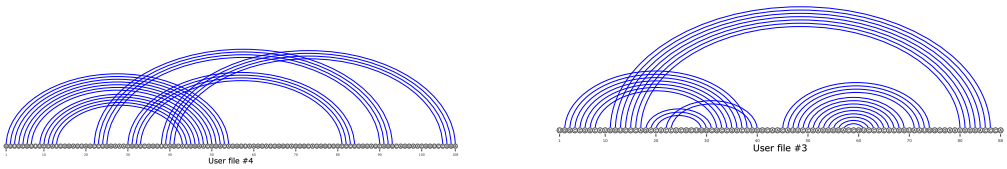
Figure 4.2: Tree-width comparison for algorithms GreedyDegree, GreedyFillin, Lex-BFS, MCS in PseudoBase.

- Lex-BFS (Lexicographic Breadth First Search) [81];
- MCS (Maximum Cardinality Search) [95].

4.2.1 Datasets

A **PseudoBase dataset** consists of non-redundant 398 pseudoknots, and was retrieved from the PseudoBase [101] on June 12th, 2017. These structural models were obtained from experimental methods, crystallography, NMR or through high quality sequence comparisons. Some pseudoknots are stored over several segments, and structure and sequence information within the missing regions. To make full use of those pseudoknots, we merged those segments into a complete sequence, based on the rationale that the tree-width of the graph is not affected by the merge.

A second **PDB dataset** includes all the canonical and non-canonical interactions, categorized into 12 families within the Leontis/Westhof classification [56]. The 1338 non-redundant tertiary structures containing RNA strands were downloaded from the



(A) PKB71 (B) PKB75

Figure 4.3: With GreedyFillin, Tree-width for PKB71 is 5 and tree-width for PKB75 is 5. With GreedyDegree, tree-width for PKB71 is 6 and tree-width for PKB75 is 5.

PDB database [82] on June 12th, 2017. First, we used RNAVIEW [119] to generate 2-dimensional displays of RNA/DNA secondary structures with tertiary interactions. The interactions that RNAVIEW can detect are the 12 families edge-to-edge base pairs as shown in Section 1.2. The output for RNAVIEW was stored in XML format. Therefore, we had 1014 XML files by using RNAVIEW because no base-pairs were found in the other 324 PDB structures and we ignored them.

The **PseudoBase dataset** and **PDB dataset** are only used in this experiment, because the corresponding database PseudoBase and PDB only provide separate sequences and the corresponding structures and they do not provide alignment information. In the next few experiments, we will construct dataset from RFAM database. In each RFAM family, we have the alignment information and also the consensus structure.

4.2.2 Results

We tested the four tree decomposition algorithms on the PseudoBase dataset, and compared the performances by showing the tree-width. The **computation time** for the four algorithms is typically less than 1 second for one RNA structure and the time difference between the four algorithms can be ignored. The comparisons of the tree-width are shown in Figure 4.2. In detail, the x-axis is labeled with the index of pseudoknots ordered by the length and y-axis is tree-width calculated by the four algorithms. Overall, GreedyDegree and GreedyFillin clearly outperform the other two algorithms, but yield almost identical results. Furthermore, almost all the RNA graphs in **PseudoBase dataset** have tree-width 4. Figure 4.3 illustrates two examples whose tree-widths are more than 4.

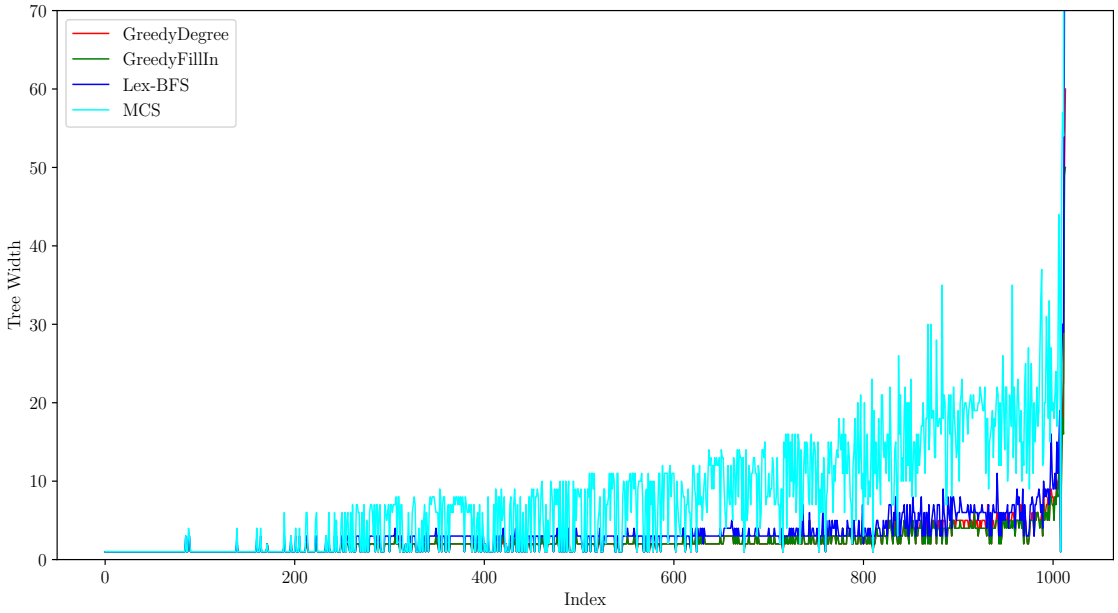


Figure 4.4: Tree-width comparison for algorithms GreedyDegree, GreedyFillin, Lex-BFS, MCS on the PDB dataset, including all non-canonical interactions.

Finally, we tested the four tree decomposition algorithms on the **PDB dataset**, and the result is shown in Figure 4.4. The result is consistent with the analysis in 2014 [25]. Also, the index of the PDB file is ordered by their sequence length. We restrict the upper value of the y-axis to be 70, because the widths calculated by MCS are much bigger than widths of former RNA structures which will make the difference for the four tree decomposition undistinguishable for the former RNA structures. It is clear that the tree decompositions calculated by GreedyDegree and GreedyFillin have relatively smaller tree-widths. The tree-width for GreedyDegree and GreedyFillin are usually less than 10. Figure 4.5 shows an example whose tree-width is 9 by using GreedyFillin.

RFAM	Len	seed/total	RFAM	Len	seed/total	RFAM	Len	seed/total
RF00009	189 – 440	116/1103	RF00010	281 – 520	458/6660	RF00011	291 – 417	114/1352
RF00023	230 – 437	477/6350	RF00024	382 – 559	37/198	RF00028	206 – 488	12/71430
RF00030	183 – 562	66/862	**RF00041**	119 – 131	60/168	**RF00094**	87 – 100	33/598
RF00140	88 – 119	39/1017	**RF00165**	62 – 64	14/538	**RF00176**	89 – 92	18/82
RF00216	300 – 304	23/163	**RF00233**	77 – 86	28/130	**RF00259**	166 – 169	5/116
RF00261	215 – 229	11/48	RF00373	233 – 475	70/371	**RF00381**	56 – 59	13/328
RF00390	23	6/26	RF00458	187 – 202	7/23	**RF00499**	102 – 115	5/32
RF00505	64 – 66	5/405	**RF00507**	79 – 87	23/555	**RF00622**	74 – 78	12/101
RF01050	1158 – 1220	13/84	**RF01072**	29 – 33	25/281	RF01073	61 – 62	7/7454
RF01074	40	4/27	**RF01075**	96	2/26	**RF01076**	72	2/213
RF01077	66 – 67	4/51	**RF01078**	56 – 59	3/5	**RF01079**	39	2/18
RF01080	32	4/19	**RF01081**	26	3/14	**RF01082**	25	2/16
RF01083	21	4/84	**RF01084**	120 – 141	8/361	**RF01085**	116 – 118	2/8
RF01087	148 – 151	6/131	**RF01088**	67 – 68	3/20	**RF01089**	121 – 128	7/71
RF01090	66 – 69	7/74	**RF01091**	61	2/11	**RF01092**	61	2/13
RF01093	60	12/145	**RF01094**	118	2/2	**RF01095**	56	2/4
RF01096	55 – 56	2/56	RF01097	52	4/2021	**RF01098**	49 – 50	2/27
RF01099	48	32/17550	**RF01100**	40	2/4	**RF01101**	40	3/41
RF01102	36 – 37	2/7	**RF01103**	29	2/12	**RF01104**	29	3/12
RF01105	27	2/5	**RF01106**	25	2/5	**RF01107**	27	2/12
RF01108	26	2/6	**RF01109**	21	2/12	**RF01111**	21	4/10
RF01113	23	2/13	**RF01114**	22	2/40	RF01577	615 – 626	2/14
RF01689	114 – 150	144/266	**RF01704**	52 – 93	627/859	RF01715	204 – 213	6/6
RF01725	87 – 139	437/773	**RF01726**	57 – 81	126/321	**RF01735**	112 – 125	30/151
RF01745	169 – 218	189/351	**RF01761**	71 – 106	118/134	**RF01768**	62	7/768
RF01775	144 – 155	7/129	**RF01785**	27	2/11	RF01786	83 – 103	54/299
RF01788	188 – 195	5/714	RF01807	196 – 219	12/35	RF01831	86 – 143	97/610
RF01833	55	4/28	**RF01834**	35	5/29	**RF01840**	54	14/103
RF01849	293 – 592	111/2078	RF01850	232 – 360	7/51			

Table 4.1: 86 Pseudoknotted RFAM families. Len: the length range. seed: number of seed sequences. total: number of full sequences. Family names with underline are used to evaluate the performance of `LiCoRNA`. Family names in bold are used for testing the correctness of `INFERNAL`.

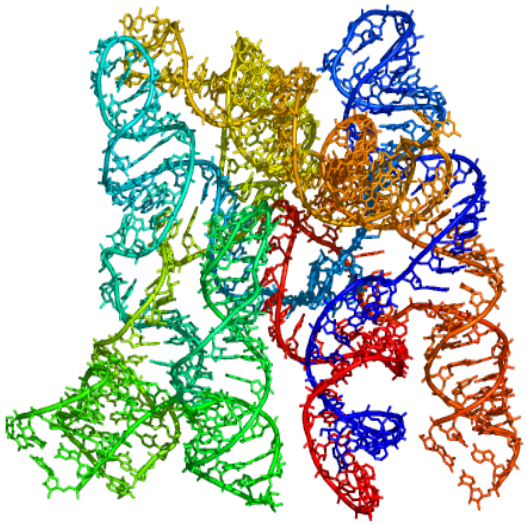


Figure 4.5: In this ribonuclease P RNA from Bacillus stearothermophilus (PDB ID: 2A64), the tree decomposition returned by the GreedyFillin heuristics on the secondary structure, including all non-canonical interactions, has width 9!

4.3 Predictive accuracy of LiCoRNA

This section first describes a benchmark which contains 86 pseudoknotted families from the RFAM database [65]. Next, we illustrate the performance of our software, called LiCoRNA (aLignment of Complex RNAs), based on Rinaudo's algorithm [78] which performs a structure-sequence alignment for pseudoknots of arbitrary complexity by comparison with other state-of-the-art softwares.

4.3.1 Dataset

The **dataset** we used for testing and analysis is the set of pseudoknotted families in RFAM database [65]. It is necessary to explain some notations about RFAM database. Each family in RFAM has two multiple sequence alignments. The seed alignment is a hand-curated alignment of known members of the family. Then INFERNAL is used to build a covariance model from the seed alignment. Finally, the full alignment is generated by searching the rfamseq sequence dataset based on the covariance model. Here we used RFAM 11.0, because since RFAM 12.0, they no longer provide full

alignment for each family. There are 86 pseudoknotted families which are annotated as pseudoknots in RFAM database as shown in Table 4.1 and the corresponding RNA names are list in Table A.1 in appendix A. Among the 86 families, 21 families with the number of seed sequence more than 10 and maximum sequence length less than 200 were selected. Furthermore, the seed alignments in these families were treated as the benchmark, because the seed alignment in each RFAM family is a hand-curated alignment which contains representative sequences in the family and the consensus structure. For families with the number of seed sequences more than 15, we use the tool rnazSelectSeqs.pl to select 15 maximally-divergent sequences [105] to reduce the computation time. These families are underlined in Table 4.1.

4.3.2 Competitors

State-of-the-art softwares for the pseudoknotted structure-sequence alignment problem include the `PAL` [39] and `PSTAG` [60] algorithms, which are described in Section 2.3. Additionally, our competitors also include `profile-csHMMs` (profile context-sensitive Hidden Markov Models), described by Yoon and Vaidyanathan [120]. It uses context sensitive grammars to represent pseudoknotted RNA structures. Based on that, a DP scheme, named *sequential component adjoining* algorithm, calculates the optimal structure-sequence alignment.

We used `LiCoRNA`, `PAL`, `PSTAG`, `profile-csHMMs` to predict the pairwise alignments on the 21 families. In other words, for each family in our benchmark, we chose in turn each of its members, along with its pseudoknotted consensus, as the query structure to predict the secondary structure of the other members.

4.3.3 Evaluation metrics

The **evaluation metrics** we use to indicate the accuracy of an alignment are Average Fractional Identity (AFI) which only considers the alignment character, and sensitivity/Positive Predictive Value(PPV) analysis which considers the structure character of the target sequence predicted by the alignment.

The Fractional Identity represents the alignment identity between the predicted and reference alignments, which is the number of identities divided by the length of the alignment. This parameter is calculated by the tool `CompalignP` that is distributed

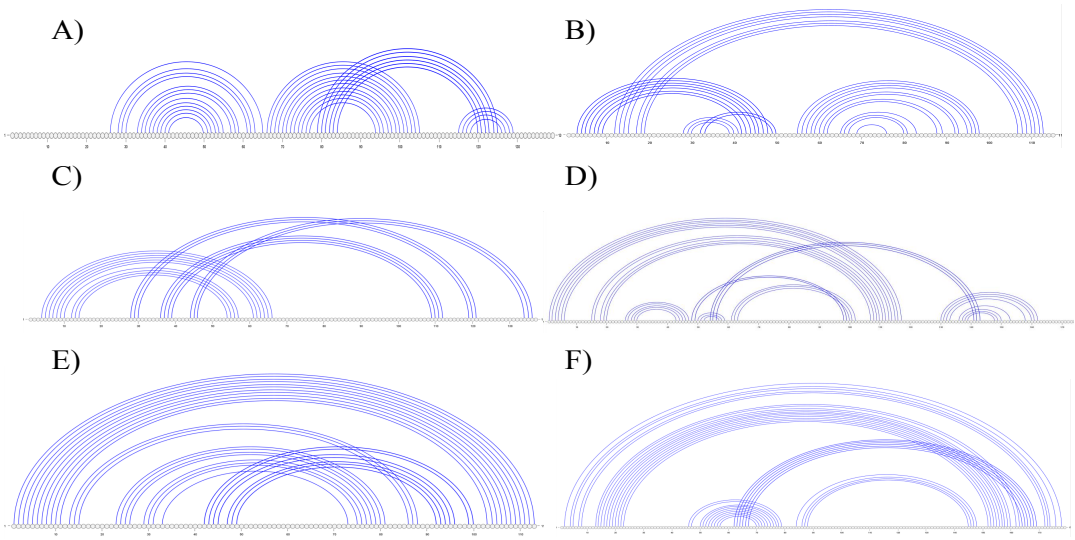


Figure 4.6: The secondary structure of the consensus structure of the six families with tree-width 4.A) RF00041; B) RF00094; C) RF00140; D) RF01689; E) RF01786; F) RF01831.

with BRAliBase 2.1 [115]. Good alignment performance is demonstrated by being close to 1. Besides, according to the alignment between query and target sequence, the structure of target sequence is predicted. The predicted structure is evaluated by PPV and sensitivity:

$$PPV = \frac{TP}{TP + FP} \tag{4.1}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{4.2}$$

where TP(true positive) represents the number of correctly predicted base pairs, FP(false positive) represents the number of predicted base pairs which are not in the reference structure, and FN (false negative) represents the number of base pairs in the reference structure that are not predicted. The fact that the parameter PPV and sensitivity are close to 1 indicates good performance.

4.3.4 Results

Table 4.2 lists the test results for the 21 pseudoknotted families. Besides, Figure 4.7 shows the alignment performances for `LiCoRNA`, `PAL`, `PSTAG`, `profile-csHMMs`. Figure 4.8 and Figure 4.9 show the sensitivity and the PPV analysis for the four softwares.The sequence identity which is calculated by esl-alipid program included in

RFAM	Len	Avg Id	tw	LiCoRNA			PAL			PSTAG			profile-csHMMs		
				SN	PPV	AFI	SN	PPV	AFI	SN	PPV	AFI	SN	PPV	AFI
RF00041	120 - 130	83.40%	4	**0.895**	**0.940**	**0.934**	-	-	-	-	-	-	-	-	-
RF00094	87 - 93	86.11%	4	**0.911**	**0.931**	0.881	-	-	-	0.899	0.913	0.887	0.908	0.921	**0.896**
RF00140	99 - 115	77.66%	4	**0.903**	**0.941**	**0.826**	-	-	-	-	-	-	-	-	-
RF00165	62 - 64	68.42%	3	**0.963**	**0.976**	**0.916**	0.888	0.920	0.827	0.945	0.952	0.872	0.958	0.966	0.884
RF00176	89 - 92	93.28%	3	0.956	0.960	**0.973**	-	-	-	**0.973**	**0.978**	0.970	0.959	0.963	0.965
RF00233	78 - 86	73.72%	3	**0.914**	**0.935**	**0.954**	0.856	0.880	0.887	0.909	0.901	0.924	0.898	0.915	0.932
RF00381	56 - 59	84.19%	3	0.950	0.991	**0.966**	**0.956**	**1.000**	0.963	0.935	0.967	0.928	0.945	0.982	0.930
RF00507	79 - 87	73.97%	3	**0.931**	0.982	0.952	0.927	**0.994**	**0.967**	0.915	0.926	0.858	0.916	0.947	0.868
RF00622	74 - 78	84.50%	3	**0.927**	**0.970**	**0.980**	-	-	-	0.908	0.949	0.945	0.912	0.951	0.946
RF01072	29 - 33	80.89%	3	0.973	0.999	0.974	**0.975**	**1.000**	**0.980**	0.966	0.991	0.928	0.961	0.987	0.923
RF01093	60	76.41%	3	0.906	0.955	0.983	**0.945**	**1.000**	**0.998**	0.891	0.914	0.787	0.883	0.921	0.848
RF01099	48	86.15%	3	**0.936**	**1.000**	**1.000**	0.892	0.976	0.971	0.934	0.991	0.993	0.935	0.998	0.997
RF01689	125 - 134	83.26%	4	**0.957**	**0.986**	**0.966**	-	-	-	-	-	-	-	-	-
RF01704	56 - 64	75.50%	3	0.981	0.999	**0.933**	**0.982**	**1.000**	0.921	0.970	0.986	0.911	0.975	0.992	0.918
RF01725	90 - 128	84.02%	3	**0.974**	**0.998**	**0.938**	-	-	-	-	-	-	-	-	-
RF01726	59 - 61	79.02%	3	**0.966**	**0.999**	**0.950**	-	-	-	0.957	0.974	0.892	0.959	0.982	0.917
RF01735	112 - 125	77.23%	3	**0.947**	**0.963**	**0.906**	0.900	0.955	0.902	-	-	-	-	-	-
RF01761	93 - 100	80.15%	3	0.970	**0.980**	**0.933**	0.963	0.975	0.926	-	-	-	**0.972**	0.975	0.931
RF01786	83 - 86	64.85%	4	0.906	0.964	0.942	0.888	**0.975**	**0.951**	**0.907**	0.924	0.880	0.897	0.943	0.907
RF01831	96 - 103	74.71%	4	**0.961**	**0.993**	**0.960**	-	-	-	-	-	-	0.960	0.989	0.950
RF01840	54	88.14%	3	**0.945**	0.991	0.995	-	-	-	0.944	0.988	0.988	**0.945**	**0.997**	**0.996**

Table 4.2: Pairwise tests: Statistics for Sensitivity, PPV and AFI based on 21 Pseudoknotted RFAM families for LiCoRNA, PAL, PSTAG and Profile-csHMMs. SN represents Sensitivity, PPV for Positive Predictive Value, AFI for Average Fractional Identity, Len for the length range.The highest values of parameters SN, SP, AFI for each family are labeled in bold.

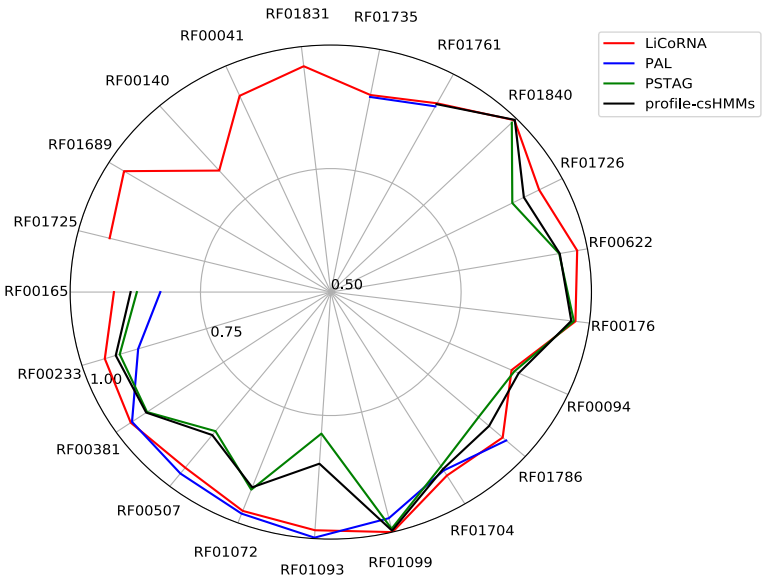


Figure 4.7: The AFI of `LiCoRNA`, `PAL`, `PSTAG`, `profile-csHMMs` for the 21 pseudoknotted families.

INFERNAL ranges from 64.85% (RF01786) to 93.28% (RF00176). The tree decomposition method we used is GreedyFillIn. There are six families with tree-width 4 and the consensus structures are shown in 4.6. '$-$' symbols in Table 4.2 mean that this family can not be predicted by a particular program. PSTAG can not be executed for families RF00041, RF00140, RF01689, RF01725, RF01735, RF01761 and RF01831. Part or all the pairwise sequence alignments for families RF00041, RF00140, RF01689, RF01725 and RF01735 can not be predicted by profile-csHMMs. PAL can be used to deal with simple pseudoknots only. Then we get two conclusions: (1) It is obvious that only `LiCoRNA` can deal with all the families.(2) Some pseudoknots with tree-width 4 can also be predicted by other three programs, especially the family RF01786.

Besides, from Table 4.2, `LiCoRNA` shows generally equivalent or better results than its competitors for almost all the families, especially for the parameter AFI. For one family, it is possible that the best performance programs according to different parameters *sensitivity*, PPV and AFI are different. For example, in family RF00094, `LiCoRNA` performs best for parameter *sensitivity* and PPV. However, for AFI, program `profile-csHMMs` performs best. This indicates that in the alignment region for stems, `LiCoRNA` performs best, while when considering the whole alignment includ-

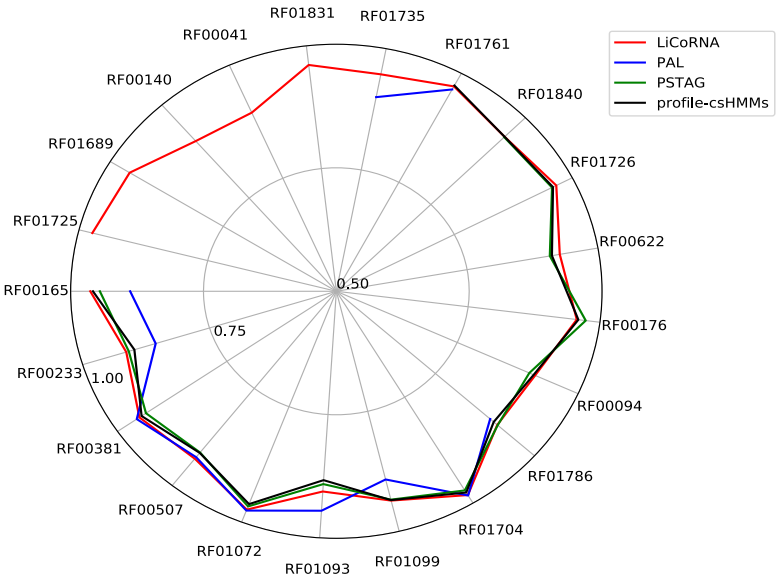


Figure 4.8: The sensitivity of `LiCoRNA`, `PAL`, `PSTAG`, `profile-csHMMs` for the 21 pseudoknotted families.

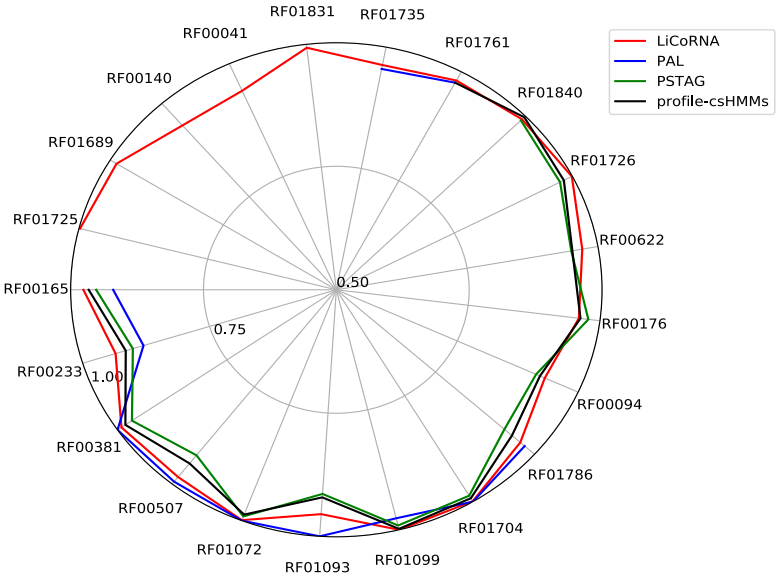


Figure 4.9: The PPV of `LiCoRNA`, `PAL`, `PSTAG`, `profile-csHMMs` for the 21 pseudoknotted families.

RFAM	tw	Time (s)			
		LiCoRNA	PAL	PSTAG	profile-csHMMs
RF00041	4	575.77	-	-	-
RF00094	4	372.54	-	53.65	1.51
RF00140	4	8348.09	-	-	-
RF00165	3	72.75	1.87	7.89	0.33
RF00176	3	52.23	-	57.56	0.53
RF00233	3	23.24	0.20	34.04	1.89
RF00381	3	71.64	1.32	3.62	0.45
RF00507	3	87.08	4.54	34.27	3.23
RF00622	3	59.37	-	20.81	0.63
RF01072	3	21.07	0.08	0.14	0.06
RF01093	3	56.62	1.36	4.58	0.22
RF01099	3	31.21	0.58	1.34	0.07
RF01689	4	731.88	-	-	-
RF01704	3	46.59	1.60	6.07	0.71
RF01725	3	2841.8	-	-	-
RF01726	3	27.24	-	5.80	0.21
RF01735	3	106.36	0.34	-	-
RF01761	3	35.89	0.77	-	2.03
RF01786	4	1805.76	5.99	42.54	0.81
RF01831	4	2883.84	-	-	2.26
RF01840	3	45.16	-	2.86	0.09

Table 4.3: Time comparison for pairwise test for LiCoRNA, PAL, PSTAG and profile-csHMMs.

ing the loop regions, profile-csHMMs performs better. It must be noted that some structures predicted by PAL were corrected by deleting some base pairs. Since the input query structure for PAL is always the same as consensus structure even though for some base pairs in the query structure are neither WatsonCrick nor Wobble base pairs. Deleting such base pairs in the query structure and the resulting base pairs in the predicted structure for target sequence makes the results comparable with the results of other three programs.

Table 4.3 depicts the average computation time for an alignment within each RFAM family for the four tools. Comparing with PAL, PSTAG, profile-csHMMs, LiCoRNA uses more time. There are two main factors that affect the running time of LiCoRNA.

One is tree-width (tw). The complexity of Rinaudo's algorithm can be reduced to $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ if we can find a smooth tree decomposition [78]. However, the current generated tree decompositions are not smooth. This is the main reason that `LiCoRNA` uses more time. The second factor is the length difference between Q and W because of the use of banded DP strategy with complexity $\mathcal{O}(|\mathcal{X}| \cdot N^{tw})$ where N is the length difference between query structure and target sequence. This can explain why family RF01725 cost so much time even though the tree-width is 3 for the consensus structure.

4.4 Analyzing near optimal solutions

Aside from the former accuracy parameters sensitivity/PPV and AFI, K-best algorithm can also be used to check the accuracy of `LiCoRNA`. If the reference alignment is always in the K best suboptimal alignments, or the score distance between reference and optimal alignment is small, we can also prove the accuracy of `LiCoRNA`.

4.4.1 Dataset

The **dataset** we used is the 21 RFAM pseudoknotted families which is the same as the dataset for testing the accuracy of `LiCoRNA`.

4.4.2 Reference alignments have quasi optimal scores

We first directly compare the optimal alignment with the reference alignment by calculating

$$\frac{Score_{ref} - Score_{opt}}{MinLength}$$

for each pairwise alignment in each family where $Score_{ref}$ represents the score of the reference alignment and $Score_{opt}$ represents the score of the optimal alignment. This parameter describes the cost distance between reference and optimal alignment per position. $Score_{ref}$ and $Score_{opt}$ are both calculated by our tool `calAligncost` in `LiCoRNA`. Figure 4.10 illustrates $\frac{Score_{ref}-Score_{opt}}{MinLength}$ for each family.

These are the three reasons that make pairwise alignments predicted by `LiCoRNA` different from those in RFAM pseudoknotted families. One is the use of banded DP

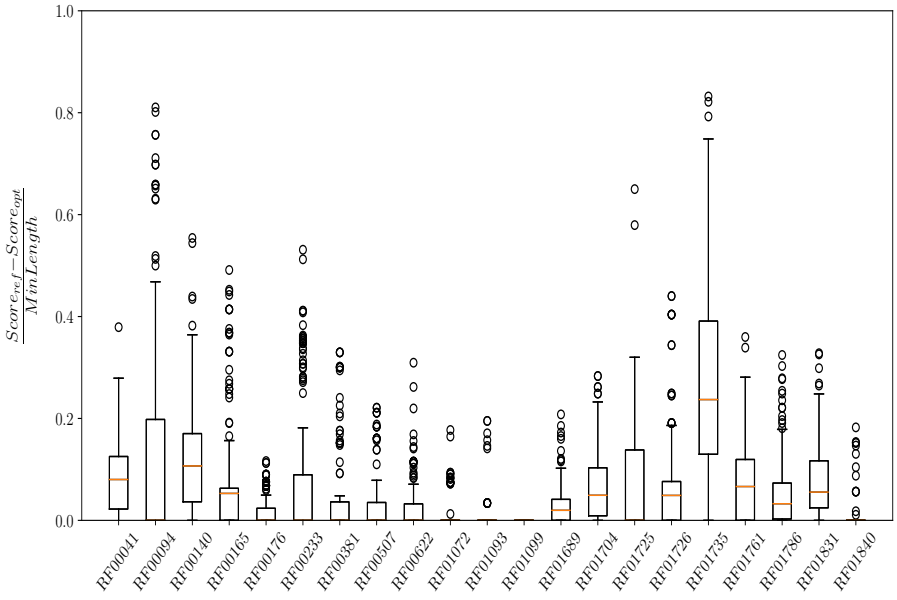


Figure 4.10: $\frac{Score_{ref}-Score_{opt}}{MinLength}$ for each family where $Score_{ref}$ represents the score of the reference alignment and $Score_{opt}$ represents the score for the optimal alignment. The parameter can be negative and in order to make the figure neat, we do not draw 4 extreme negative values in family RF00041. The 4 negative values are -0.237, -0.406, -1.113, -1.290.

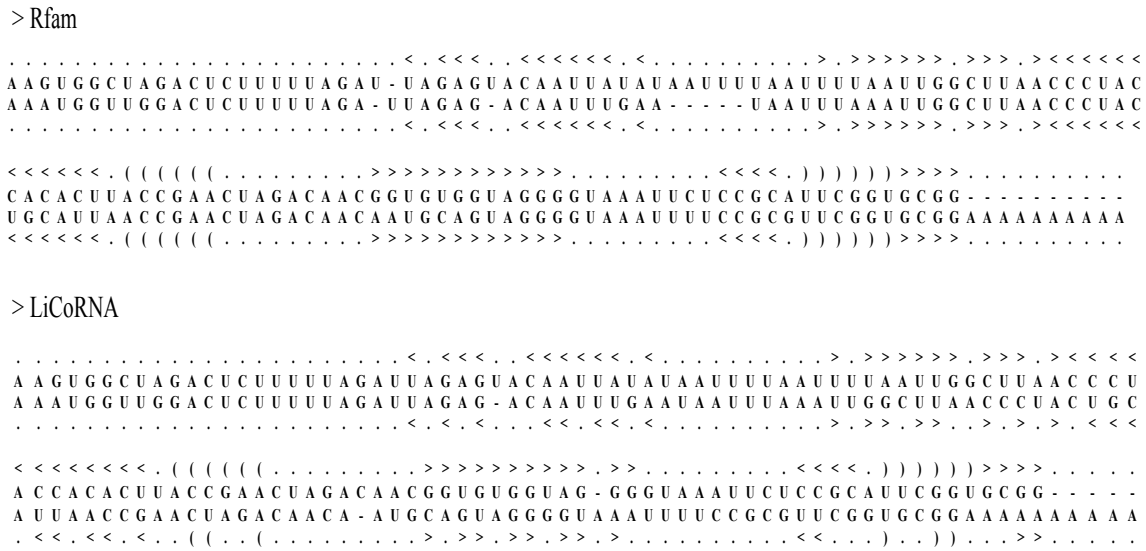


Figure 4.11: Pairwise sequence alignments from RFAM database and predicted by `LiCoRNA` in family RF00041. In this case, $\frac{Score_{ref} - Score_{opt}}{MinLength}$ is negative with value -1.290.

as discussed in Section 3.1.3. Therefore, our method is heuristic and not guaranteed to return the absolute best alignment. In this case, the reference alignment may no longer be part of the search space, and its score may be smaller than the cost of the returned optimal alignment. An example is showed in Figure 4.11

The second reason is that there are many open gaps in RFAM pairwise alignment. One example which is from family RF01735 is illustrated in Figure 4.12. Family RF01735 is a conserved RNA motif named eps-Associated RNA element (EAR) which is associated with exopolysaccharide (eps) or capsule biosynthesis genes. The EAR RNA consists of five helical segments (P1-P5) and a pseudoknot and EAR presents a variable length stem-loop region near P3 [47]. The benchmark shows a scattered alignment in this region, with multiple gap openings. As our scoring function uses an affine gap penalty model, such an alignment is heavily penalized and therefore largely suboptimal.

The third reason is that there are more gaps in the structural regions, but the optimal alignment mode of `LiCoRNA` produces alternative alignments with fewer gaps in the structural region. One example which is from family RF01735 is shown in Figure 4.13. Those two pairwise alignments have almost the same number of gaps

Figure 4.12: Pairwise sequence alignments from RFAM database and predicted by `LiCoRNA` in family RF01735. The RFAM pairwise alignment have multiple gap openings.

and the alignment predicted by `LiCoRNA` has less gaps in the structural regions and more gaps in the loop region.

Based on the second and third reason, we hypothesize that the seed alignment of the RFAM family RF01735 may be wrong in this region and could probably be refined, a task that we cannot currently address automatically, but belongs to the perspectives of our work. However, for the first reason, we can solve it by setting K properly and we will talk about it in the next section.

4.4.3 Reference alignment are not far down the list of suboptimal alignments

As we have shown that the optimal alignment sometimes is not the reference alignment and the score of the reference alignment is close to optimal, it is possible to find the reference alignment near the optimal alignment. Then how to set the parameter K properly so that the user can find the reference alignment? To give an intuition about the value of K, we set $K = 1000$ subjectively and check the presence of the reference alignment in the near 1000 alignments.

Therefore, we generated 1000 suboptimal alignments for each pairwise comparison in 21 RFAM families. Furthermore, we also calculated the percentage of the presence of reference alignment in K suboptimal alignments for each family. Suppose the

```
> Rfam
. . < < < < < < < . . ( ( ( ( . ( ( ( . . < < < . ( ( . . . . . > > > > > > > > > ) ) . . . . < < < < < < < < . . < < < . <
G U G G C C G G C A U G G U C - C C A - - G C C U C C C C G C U G G C G C C G G C U G G G C A A C A U U C C G A - - G G G G A
U C G G G U C G G C A U G G G A U C U C C A C C U C C U C G C - G G U C C G A C C U G G G - - - C A U - C C G U A A G G A G A
. . . > . . . > > > . > > > > . > > > > . . . . . . . . ) ) ) ) ) ) ) . .
C C G U - - - C C C - U C G G U A A U G G C G A A - - - U G G G A C C C A
G C G U A G C U C C C U C G G - - A U G G C A A A G G G G A G A U C C A C C

> LiCoRNA
. . < < < < < < < . . . . . ( ( ( ( ( ( ( < < < . ( ( . . . . . > > > > > > > > > ) ) . . . . < < < < < < . < < < < < . . .
G U G G C C G G C - A U - - G G U C C C A G C C U C C C C G C U G G C G C C G G C U G G G C A A C A U U C C - G A G G G G - -
U C G G G U C G G C A U G G G A U C U C C A C C U C C U C G C - G G U C C G A C C U G G G - - - C A U C C G U A A G G A G A G
. < . . . > > > > > > > > . > > > > . . . . . . ) ) ) ) ) ) ) . . . .
- A C C G U C C C U C G G U A A U G G C G A A - U G G G A C C C A - -
C G U A G C U C C C U C G - G A U G G C A A A G G G G A G A U C C A C C
```

Figure 4.13: Pairwise sequence alignments from RFAM database and predicted by `LiCoRNA` in family RF01735. The pairwise alignment predicted by `LiCoRNA` has less gaps in the structural region.

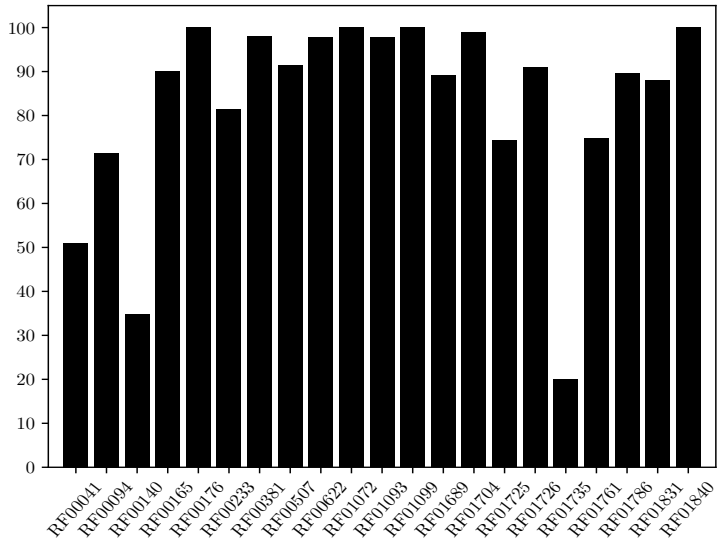


Figure 4.14: Average percentage of the presence of reference alignment in 1000 sub-optimal pairwise alignments in each family.

alignment in RFAM family is the biological alignment, we ask the question that if we can find it near the optimal alignment. Figure 4.14 lists the percentage of presence of reference alignments for the 21 RFAM families. As expected, we can find the reference alignment in the 1000 suboptimal alignments.

However, family RF01735, RF00140, RF00041 show a relative low percentage ($< 50\%$) because of the same reason discussed in Section 4.4.2. Besides that, setting $K = 1000$ is enough to get the reference alignment.

4.5 Stochastic sampling enables the detection of ambiguously-aligned regions

Compared with the K-best algorithm, the Stochastic backtracking algorithm can generate the suboptimal alignments with repetition. However, this algorithm can generate different alignments by setting different pseudo-temperatures (parameter t) and these sampling alignments are the representations of the alignment ensemble at a particular t. Partition function and stochastic pairwise alignment algorithm are useful in many different contexts, such as test whether the alignment is well-defined and the reliability of the aligned nucleotides as shown in [63]. We will show an example to explain how to use stochastic sampling to detect the ambiguously-aligned regions.

4.5.1 Evaluation metrics

Reliability can be numerically quantified for each position in query structure Q by the **positional entropy** metric. For any given position i in Q, the entropy $H(i)$ associated with a position i is computed as

$$H(i) = -\sum_{k=1}^{n} p_{ik} \cdot \log_2(p_{ik}) - q_i \cdot \log_2(q_i)$$

where n is the length of the target sequence, p_{ik} is the probability that the position i of Q aligns with the position k of W, and q_i is the probability that the position i aligns with a gap. A positional entropy of 0 for a position i means that the position i always aligns to the same position in W.

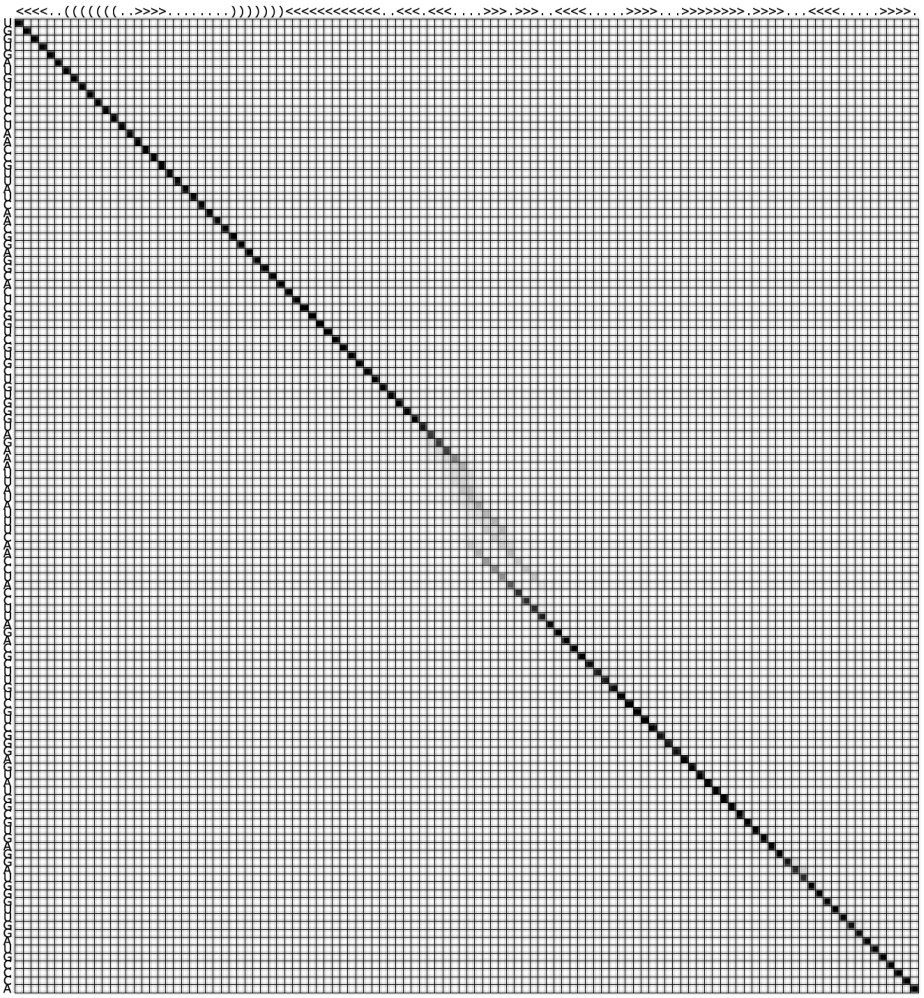


Figure 4.15: Dot plot of the alignment between query structure ACNG01000079.1/ 252537-252424 and target sequence AAJM01000088.1/ 3912-4034 in RF01735 when $t = 1.5$. The horizontal axis of the dot plot is the query structure. The darker the square, the higher match probability between the corresponding positions.

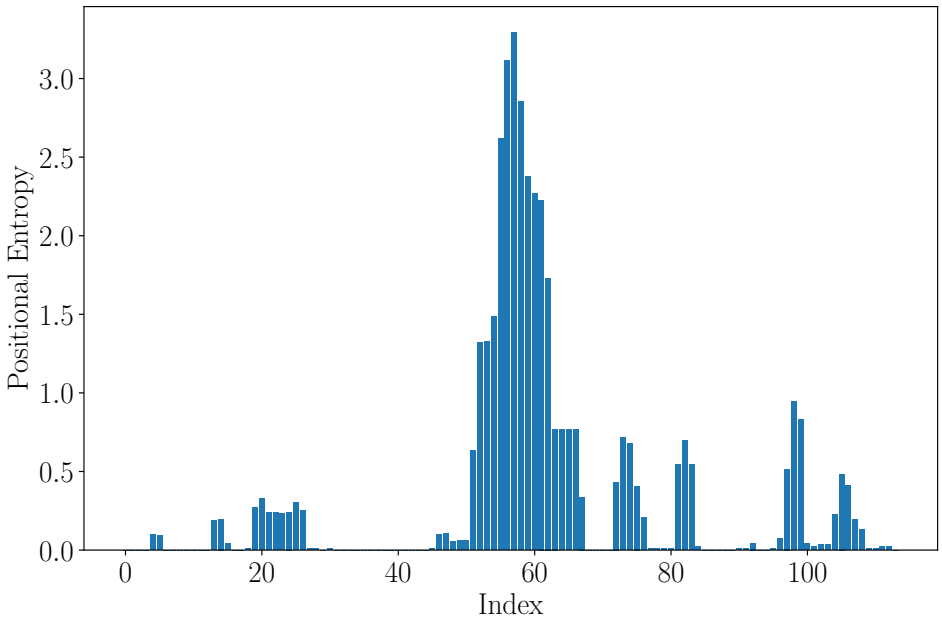


Figure 4.16: Positional entropy for each position in query structure ACNG01000079.1/ 252537-252424 in family RF01735. The target sequence is AAJM01000088.1/ 3912-4034.

4.5.2 An example

As shown in Figure 4.14, the percentage of the presence of reference alignment in 1000 suboptimal alignments for family RF01735 is low. To find the ambiguously-aligned regions, we generate 1000 stochastic pairwise alignments between query structure ACNG01000079.1/ 252537-252424 and target sequence AAJM01000088.1/ 3912-4034 in RF01735 when $t = 1.5$. We first calculated the Boltzmann match probabilities. Figure 4.15 illustrates match probabilities for the aligned nucleotides. The horizontal axis of the dot plot is the query structure. The darker the square, the higher match probability between the corresponding position. The region with lower match probabilities shows several different possibilities to align the two subsequences. This means that the alignment in this region is less reliable than that in other regions.

To be more precise, we calculate the positional entropy for every position of ACNG01 000079.1/ 252537-252424, as shown in Figure 4.16. The higher the positional entropy, the more ambiguous the position is. So the positions near 60 ambiguously aligns to positions in target sequence which can also be seen in Figure 4.15.

4.6 Structure-based realignment of RFAM families improves support for pseudoknotted base-pairs

As `LiCoRNA` shows a good performance and due to the fact that `INFERNAL` does not support pseudoknots, we used `LiCoRNA` to curate RFAM full pseudoknotted alignments. We hope that a systematic realignment will allow to reveal or refute an evolutionary pressure towards the preservation of a functional pseudoknot. However, since our scoring scheme is not purely based on an evolutionary model, we simply inquire whether the compatibility with pseudoknots, which are the object of an *ad hoc* manual annotation in RFAM alignments, can be increased by explicitly considering them during the alignment phase.

4.6.1 Dataset

To avoid misalignment in RFAM families due to the limitation of covariance models, we considered all the 86 pseudoknotted families. Then we did the preprocessing procedures to reduce the data. Finally, 60 families in bold were selected in Table 4.1. We did not run all the families mainly because the pairwise alignment costs too much time for some families. The preprocessing procedure includes:

- Exclude the families that the maximum sequence length more than 200.
- In one family, use rnazSelectSeqs.pl to select sequences where each pairwise sequence identity is less than 99%.
- Exclude the families where the running time for the pairwise alignment is more than 700 s (RF00140, RF01689, RF01831, RF01725, RF01786).
- Exclude the families that contain more than 500 sequences (RF01073, RF01788, RF01097).

4.6.2 Evaluation metrics

The parameters we use is to compute the **base pair conservation** and sequence identity. In one family, we choose one sequence with the same number of base pairs

as the consensus structure and treat the structure of it as the reference structure. Furthermore, we use LiCoRNA to align the reference sequence with other members in the family. The **base pair conservation** for base pair with index i in the reference structure is represented as BPC^i. Therefore, $BPC^i_{\texttt{RFAM}}$ and $BPC^i_{\texttt{LiCoRNA}}$ are the ith base-pair percentage for alignments in RFAM and predicted by LiCoRNA. To represent the base pair conservation for all base pairs, we average the BPC^i with equation $\frac{\sum_{i=1}^{v} BPC^i}{v}$ where v is the number of base pairs in the reference structure and we denote the average value of BPC^i with $avgBPC$.

The **base pair conservation increment** for base pair with index i is defined as $BPCI^i = BPC^i_{\texttt{LiCoRNA}} - BPC^i_{RFAM}$. Similarly, $AvgBPCI = \frac{\sum_{i=1}^{v} BPCI^i}{v}$ is the average value of the base pair conservation increment for all base pairs in the reference structure. The sequence identity is calculated by esl-alipid program in INFERNAL. The assumption behind this test is that an alignment is considered better when its predicted structure has more base pairs, and its sequence identity is comparable to that of the RFAM alignment.

4.6.3 Results

Table 4.4 lists $AvgBPC$ of all base pairs for the RFAM alignments and those predicted by LiCoRNA for **RFAM dataset 60**. Three different columns $S\%$, $P\%$ and $T\%$ are the $AvgBPC$ values of base pairs when the base pairs are in the secondary structure region, pseudoknotted region and both of the regions. Figure 4.17 illustrates the families that at least one of the columns $S\%$, $P\%$ and $T\%$ is different in Table 4.4. The $AvgBPCI$ values of all base pairs for each family is showed in Figure 4.17 and the result shows that the structures abstracted from the alignments using LiCoRNA have relatively more base pairs than the structures abstracted from RFAM alignments without losing sequence identity except one family RF00499. The reason is that the consensus structure for RF00499 has 38 base pairs in the secondary structure region and 5 base pairs in the pseudoknot region and one fewer base pairs in the pseudoknot region can reduce the percentage significantly.

Let us first see an example to show the comparison in detail. RF01089 has 27 base pairs in the consensus secondary structure, including 19 secondary base pairs and 8 pseudoknot base pairs and there are 56 selected sequences in this family. We calculate the BCP^i value of a base pair with index i for the alignments predicted by LiCoRNA with/without pseudoknot base pairs in the reference structure and those of RFAM.

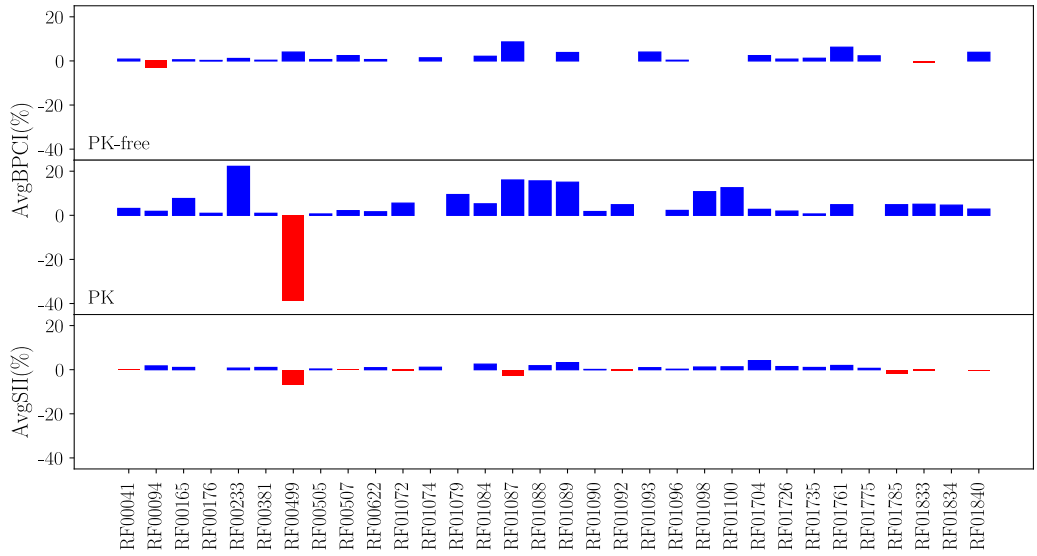


Figure 4.17: $AvgBPCI$ of base pairs in the reference structure and sequence identity for families that at least one of the columns $S\%$, $P\%$ and $T\%$ is different in Table **??** in appendix material. $AvgBPCI$: The average of base pair conservation increment of all base pair for each family. $AvgSII$: the average of sequence identity increment with equation $AvgSII = \frac{SI_{\texttt{LiCoRNA}} - SI_{RFAM}}{v}$ where SI represents sequence identity and v is the number of sequence for one family. BPs: base pairs. Blue columns mean that `LiCoRNA` is better and red columns mean that RFAM is better.

RFAM	Len	tw	#BP			RFAM/LiCoRNA					
			#S	#P	#T	S%	P%	T%	Avg Id%	Avg #open_gaps	Avg #gaps
RF00041	64 – 130	4	29	6	35	0.908/**0.916**	0.905/**0.936**	0.907/**0.920**	84.7/84.5	1.65/1.69	5.24/4.56
RF00094	84 – 100	4	22	9	31	**0.962**/0.930	0.975/**0.993**	**0.965**/0.948	70.8/72.5	7.22/6.69	12.6/9.82
RF00165	57 – 64	3	10	8	18	0.981/**0.986**	0.895/**0.971**	0.942/**0.980**	55.7/56.7	2.60/2.25	2.68/2.33
RF00176	34 – 95	3	22	3	25	0.896/**0.898**	0.739/**0.748**	0.877/**0.880**	92.4/92.4	1.03/1.05	11.9/11.9
RF00233	46 – 87	3	20	3	23	0.899/**0.910**	0.684/**0.906**	0.871/**0.909**	67.8/68.5	2.96/3.05	8.77/7.77
RF00381	35 – 65	3	12	6	18	0.904/**0.907**	0.926/**0.935**	0.911/**0.916**	81.8/82.8	1.06/1.15	2.78/2.46
RF00390	22 – 23	3	4	3	7	0.875/0.875	0.889/0.889	0.881/0.881	92.0/92.0	0.167/0.167	0.167/0.167
RF00499	31 – 117	3	38	5	43	0.546/**0.586**	**0.889**/0.500	0.572/**0.580**	80.3/73.5	3.43/4.77	47.5/42.3
RF00505	62 – 66	3	7	9	16	0.980/**0.986**	0.989/**0.995**	0.985/**0.991**	88.6/88.9	0.667/0.714	0.714/0.714
RF00507	50 – 87	3	11	8	19	0.950/**0.974**	0.793/**0.814**	0.880/**0.903**	68.5/68.4	0.781/1.03	2.52/2.77
RF00622	62 – 79	3	15	8	23	0.892/**0.898**	0.955/**0.971**	0.914/**0.923**	80.5/81.4	1.06/1.21	3.32/2.90
RF01072	22 – 33	3	4	6	10	0.954/0.954	0.643/**0.698**	0.767/**0.800**	85.3/84.8	0.514/0.586	2.14/2.29
RF01074	40	3	5	3	8	0.957/**0.971**	1.000/1.000	0.973/**0.982**	85.0/86.1	0.444/0.444	0.444/0.444
RF01075	95 – 96	3	26	5	31	1.000/1.000	1.000/1.000	1.000/1.000	90.8/90.8	1.5/1.5	1.5/1.5
RF01076	63 – 72	3	12	7	19	0.991/0.991	1.000/1.000	0.995/0.995	91.8/91.8	0.0204/0.0204	0.184/0.184
RF01077	66 – 67	3	18	4	22	0.980/0.980	0.926/0.926	0.971/0.971	93.1/93.1	0.118/0.118	0.118/0.118
RF01078	56 – 59	3	10	5	15	0.980/0.980	0.880/0.880	0.947/0.947	91.6/92.3	1.2/1.4	1.8/2
RF01079	19 – 39	3	4	4	8	1.000/1.000	0.719/**0.813**	0.859/**0.906**	98.1/98.1	0.5/0.75	4.88/4.88
RF01080	32	3	7	4	11	0.943/0.943	1.000/1.000	0.964/0.964	88.1/88.1	0.0/0.0	0.0/0.0
RF01081	26	3	5	4	9	0.950/0.950	0.938/0.938	0.944/0.944	86.5/86.5	0.0/0.0	0.0/0.0
RF01082	25	3	6	4	10	0.967/0.967	1.000/1.000	0.980/0.980	86.4/86.4	0.0/0.0	0.0/0.0
RF01083	21	3	5	3	8	1.000/1.000	0.667/0.667	0.875/0.875	95.2/95.2	0.0/0.0	0.0/0.0
RF01084	84 – 144	3	37	6	43	0.907/**0.928**	0.782/**0.834**	0.890/**0.915**	54.2/56.7	6.92/6.06	16.7/15.4
RF01085	115 – 118	3	32	6	38	0.992/0.992	0.958/0.958	0.987/0.987	96.5/95.7	0.75/0.75	1.0/1.0
RF01087	41 – 153	3	18	10	28	0.776/**0.862**	0.612/**0.772**	0.717/**0.830**	88.2/85.5	2.28/3.72	22.8/22.8
RF01088	65 – 68	3	18	5	23	0.981/0.981	0.800/**0.956**	0.942/**0.976**	90.9/92.7	0.333/1.44	0.444/1.56
RF01089	41 – 130	3	19	8	27	0.796/**0.834**	0.616/**0.766**	0.743/**0.813**	58.7/61.9	4.70/5.36	15.1/15.8
RF01090	51 – 69	3	10	8	18	0.965/0.965	0.880/**0.897**	0.928/**0.935**	90.4/90.5	0.173/0.261	0.913/0.913
RF01091	61	3	12	5	17	1.000/1.000	1.000/1.000	1.000/1.000	95.9/95.9	0.0/0.0	0.0/0.0
RF01092	61	3	7	7	14	1.000/1.000	0.952/**1.000**	0.976/**1.000**	92.9/92.3	0/0.333	0/0.333

RFAM	Len	tw	#BP			RFAM/LiCoRNA					
			#S	#P	#T	S%	P%	T%	Avg Id%	Avg #open_gaps	Avg #gaps
RF01093	59 – 63	3	11	5	16	0.831/**0.871**	0.986/0.986	0.879/**0.907**	76.1/77.0	0.140/0.953	0.256/1.19
RF01094	118	3	11	5	16	1.000/1.000	1.000/1.000	1.000/1.000	93.6/93.6	0.0/0.0	0.0/0.0
RF01095	56	3	12	3	15	1.000/1.000	1.000/1.000	1.000/1.000	99.1/99.1	0.0/0.0	0.0/0.0
RF01096	28 – 57	3	10	7	17	0.935/**0.938**	0.824/**0.846**	0.889/**0.900**	92.6/92.8	0.923/0.846	3.50/3.27
RF01098	42 – 50	3	5	7	12	1.000/1.000	0.833/**0.940**	0.903/**0.965**	89.1/90.3	0.333/0.833	0.8333/1.33
RF01099	38 – 50	3	5	6	11	0.922/0.923	0.960/0.960	0.943/0.943	90.4/90.5	0.0176/0.0276	0.0603/0.0704
RF01100	40	3	7	4	11	1.000/1.000	0.875/**1.000**	0.955/**1.000**	97.5/98.8	0.0/1.0	0.0/1.0
RF01101	37 – 40	3	3	5	8	1.000/1.000	1.000/1.000	1.000/1.000	93.3/93.3	0.1/0.1	0.3/0.3
RF01102	36 – 37	3	6	5	11	0.875/0.875	0.625/0.625	0.761/0.761	82.7/82.7	0.625/0.625	0.625/0.625
RF01103	29	3	5	6	11	1.000/1.000	1.000/1.000	1.000/1.000	96.6/96.6	0.0/0.0	0.0/0.0
RF01104	29	3	3	7	10	1.000/1.000	0.976/0.976	0.983/0.983	91.4/91.4	0.0/0.0	0.0/0.0
RF01105	27	3	4	5	9	1.000/1.000	1.000/1.000	1.000/1.000	98.2/98.2	0.0/0.0	0.0/0.0
RF01106	25	3	3	7	10	1.000/1.000	1.000/1.000	1.000/1.000	96.0/96.0	0.0/0.0	0.0/0.0
RF01107	27	3	4	4	8	1.000/1.000	1.000/1.000	1.000/1.000	97.5/97.5	0.0/0.0	0.0/0.0
RF01108	26	3	5	5	10	0.960/0.960	0.680/0.680	0.820/0.820	88.5/88.5	0.0/0.0	0.0/0.0
RF01109	21	3	3	5	8	1.000/1.000	0.800/0.800	0.875/0.875	95.2/95.2	0.0/0.0	0.0/0.0
RF01111	21	3	3	5	8	1.000/1.000	1.000/1.000	1.000/1.000	97.6/95.2	0.0/0.0	0.0/0.0
RF01113	23	3	3	4	7	1.000/1.000	0.917/0.917	0.952/0.952	94.2/94.2	0.0/0.0	0.0/0.0
RF01114	22	3	3	4	7	1.000/1.000	0.833/0.833	0.905/0.905	95.5/95.5	0.0/0.0	0.0/0.0
RF01704	28 – 89	3	13	5	18	0.878/**0.902**	0.938/**0.965**	0.894/**0.919**	54.0/58.1	2.99/2.46	6.71/5.28
RF01726	49 – 93	3	8	4	12	0.915/**0.923**	0.939/**0.958**	0.923/**0.934**	72.8/74.2	1.16/1.23	3.33/3.19
RF01735	60 – 128	3	36	4	40	0.949/**0.961**	0.958/**0.964**	0.950/**0.962**	81.4/82.4	2.38/1.96	7.73/7.09
RF01761	47 – 119	3	18	4	22	0.864/**0.926**	0.875/**0.923**	0.866/**0.926**	69.4/71.3	4.38/3.40	11.0/7.68
RF01768	62	3	11	7	18	0.986/0.986	0.929/0.929	0.964/0.964	92.8/92.8	0.0/0.0	0.0/0.0
RF01775	143 – 155	3	29	7	36	0.960/**0.983**	1.000/1.000	0.968/**0.986**	90.5/91.1	3.08/3.42	6.33/5.83
RF01785	26 – 27	3	6	3	9	1.000/1.000	0.714/**0.762**	0.905/**0.921**	94.2/92.5	0.143/0.143	0.143/0.143
RF01833	37 – 55	3	7	5	12	**0.964**/0.957	0.850/**0.900**	0.917/**0.933**	94.1/93.6	0.05/0.15	0.9/0.9
RF01834	23 – 35	3	5	6	11	0.982/0.982	0.909/**0.955**	0.942/**0.967**	93.5/93.5	0.0909/0.273	1.09/1.09
RF01840	42 – 54	3	10	7	17	0.758/**0.797**	0.942/**0.970**	0.834/**0.868**	87.3/86.9	0.0968/0.516	0.581/1.03

Table 4.4: Realigning 60 pseudoknotted RFAM families. BP: base pair. S: normal base pairs. P: pseudoknot base pairs. T: total base pairs.

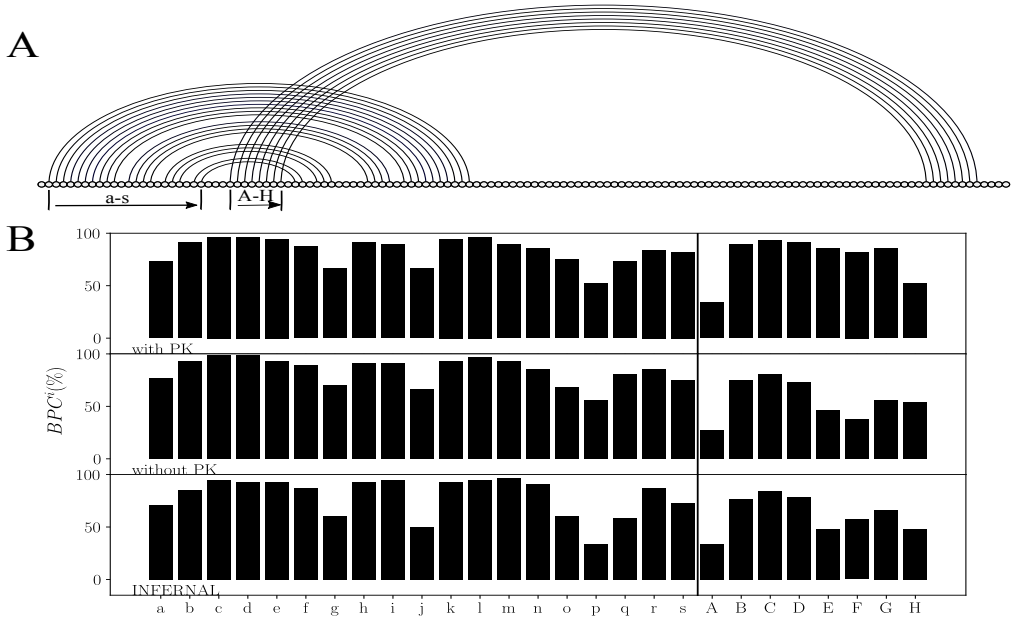


Figure 4.18: RF01089 has 27 base pairs. s1-s19 are secondary base pairs and p1-p8 are pseudoknot base pairs. This figure shows the positional percentage of base pairs through the 56 selected sequences in the family. BPC^i: base pair conservation for base pair with index i. A) is the structure of the reference sequence. The first/second row of B) shows the BCP^i for alignment by `LiCoRNA` with/without pseudoknotted base pairs. The third row of B) shows the BCP^i for those of RFAM.

Figure 4.18 A) shows the reference structure and Figure 4.18 B) shows the BCP^i value of the base pair with index i. Comparing the BCP^i values of base pairs with indices $A-H$ which are shown in the first and third row of Figure 4.18 B), the alignments of `LiCoRNA` support pseudoknot base pairs better than RFAM. However, Comparing the BCP^i values of the base pairs with indices $A-H$ which are shown in the second and third row of Figure 4.18 B), the alignments in `LiCoRNA` have relatively higher BPC^i values of the 19 base pairs in secondary structure region and do not show a good result in the pseudoknot region. Therefore, we reach our conclusion, if we consider the pseudoknot base pairs to realign using `LiCoRNA` and actually this family has pseudoknot, we get more base pairs in the pseudoknot regions without changing the base pair conservation of the 19 base pairs and the sequence identity.

To verify this idea, we further calculated the $BPCI_r^i = \frac{BPC^i_{\texttt{LiCoRNA}} - BPC^i_{RFAM}}{BPC^i_{RFAM}}$ for the predicted alignment by `LiCoRNA` compared with the alignment by `INFERNAL`. Finally, for each family, we have $AvgBPCI_r = \frac{\sum_{i=1}^{v} BPCI_r^i}{v}$ where v is the number of base pairs in the reference structure. We align the reference sequence with other members in one family with/without pseudoknotted base pairs using `LiCoRNA`. From Table 4.5, we reach several conclusions: 1) The base pair conservation in pseudoknotted regions is higher when we align with the pseudoknotted structure. The reason is obvious: when we align with the pseudoknotted region, we consider not only the sequence similarity, but also the conservation of the structure. 2) The increasing number of base pairs in the pseudoknotted region sometimes decreases the number of base pairs in the secondary structure.

4.7 Advantages and disadvantages of `LiCoRNA`

The advantages of `LiCoRNA` can be summaries as: (1) `LiCoRNA` supports any kind of pseudoknotted structures. (2) `LiCoRNA` performs equivalent performance compared with the state-of-the-art algorithms for all the RFMA pseudoknotted families in the benchmark. However, the main disadvantage is that `LiCoRNA` is slower compared with the other state-of-the-art algorithms. Therefore, more heuristic methods can be used. Besides, to further improve the accuracy of `LiCoRNA`, more score scheme can be applied and I will talk about in the next chapter.

families	S(%)		P(%)		T(%)		families	S(%)		P(%)		T(%)	
	with	without	with	without	with	without		with	without	with	without	with	without
RF00041	1.09	1.31	3.40	−0.28	1.50	1.03	RF00094	2.50	2.17	1.86	−5.94	2.31	−0.19
RF00165	0.54	0.54	35.64	−4.68	16.14	1.78	RF00176	0.31	0.32	1.23	−3.62	0.42	−0.15
RF00233	1.52	2.35	32.67	9.41	5.58	3.27	RF00381	0.52	0.52	0.96	0.77	0.67	0.60
RF00499	20.06	26.37	−43.73	−60.02	15.40	20.05	RF00505	0.75	0.75	0.56	0.0	0.64	0.33
RF00507	3.07	2.36	2.82	−0.99	2.96	0.87	RF00622	1.05	1.12	1.70	0.24	1.28	0.81
RF01072	0.0	0.0	13.06	5.62	7.83	3.37	RF01074	1.82	1.82	0.0	0.0	1.14	1.14
RF01079	0.0	0.0	14.17	0.0	7.08	0.0	RF01084	9.99	10.0	10.38	−7.57	10.05	7.55
RF01087	10.82	11.92	28.49	4.03	17.13	9.10	RF01088	0.0	0.0	42.86	9.52	9.32	2.07
RF01089	7.83	9.08	25.19	−9.70	22.05	11.0	RF01090	0.0	0.0	2.15	0.63	0.96	0.28
RF01092	0.0	0.0	5.71	0.0	2.86	0.0	RF01093	15.99	15.99	0.0	0.0	10.99	10.99
RF01096	0.66	1.01	2.86	0.90	1.57	0.97	RF01098	0.0	0.0	15.66	8.16	9.13	4.76
RF01100	0.0	0.0	0.25	0.0	9.09	0.0	RF01704	3.56	4.59	2.89	−2.98	3.37	2.48
RF01726	1.28	1.28	2.10	1.25	1.55	1.27	RF01735	1.81	1.91	0.60	−0.88	1.69	1.63
RF01761	7.76	8.61	5.54	−10.95	7.36	5.05	RF01775	2.76	2.76	0.0	0.0	2.23	2.23
RF01785	0.0	0.0	11.11	0.0	3.70	0.0	RF01833	−0.71	0.0	6.03	0.0	2.10	0.0
RF01834	0.0	0.0	5.0	0.0	2.73	0.0	RF01840	25.80	28.83	2.92	0.0	16.41	16.97

Table 4.5: $AvgBPCI_r^i$ values for each family. We align the reference structure with other members in the family twice: with pseudoknot base pairs and without pseudoknot base pairs using `LiCoRNA`. S: normal base pairs. P: pseudoknot base pairs. T: total base pairs.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

Basically, RNAs are classified into two classes: coding RNAs and non-coding RNAs (ncRNAs). ncRNAs play an important role in cellular processes by forming a specific tertiary structure. Finding the structure of ncRNA is often the first step to explain its function. Besides, as mentioned before, RNA folds in a hierarchical and sequential manner which means that RNA secondary structure often determines the tertiary structure. Therefore, it is necessary to have a correct secondary structure given the sequence.

Generally, there are two types of computational methods to predict the secondary structure of ncRNA: *De novo* folding approach when we have one single sequence and *comparative* approach when we have additional homologous sequences at hand. Here we focus on the *comparative* approach. The step structure-sequence alignment directly determinates the accuracy of the *comparative* approach. We used arc-annotated sequence to represent RNA structures and the structure-sequence alignment problem is Max SNP-hard if the structure is CROSSING or UNLIMITED. To solve the problem, Rinaudo *et al.* [78] developed a fully general method for sequence-structure comparison, which is able to take as input any type of pseudoknotted structures.

My work is based on Rinaudo's algorithm. I first introduced some implementation details about `LiCoRNA` which is a new program for structure-sequence alignment considering arbitrary pseudoknots. `LiCoRNA` uses a 4×4 matrix in RIBOSUM85-60 to score the base match/mismatch operation, a 16×16 matrix in RIBOSUM85-60 to score the arc-match and arc-mismatch operations. For arc-altering and arc-deletion

operation, we evaluated both arc ends in the alignment independently. For the implementation of the tree decomposition, we used a package named `LiBTW` as mentioned in 4.2. To accelerate the DP without losing accuracy, we implemented the banded DP with constraint N and the banded DP reduces the time complexity to $\mathcal{O}(|\mathcal{X}| \cdot N^{tw})$ compared with Rinaudo's algorithm $\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$. The number of bags in the tree decomposition is denoted as $|\mathcal{X}|$, tw represents the tree-width and n is the length of target sequence W.

Additionally, we designed three algorithms to generate suboptimal alignments: (1) stochastic backtracking algorithm based on the partition function, (2) K best suboptimal alignment algorithm and (3) Δ-suboptimal alignment algorithm.

Different from calculating one optimal alignment by Rinaudo's algorithm, we offered a method to calculate the Maximum Expected Accuracy (MEA) alignment. The computation of the MEA alignment requires the computation of the posterior probabilities for all possible correspondences of bases and base pairs. In this work, we either empirically estimate these probabilities using stochastic sampling, or compute those exactly through the inside-outside algorithm.

The results of our experiments are mainly divided into two parts. The first part is to evaluate the performance of `LiCoRNA` based on the seed alignments in the pseudoknotted RFAM families by comparison with three other state-of-the-art programs `PAL`, `PSTAG` and `profile-csHMMs`. The evaluation parameters are sensitivity/PPV and average fractional identity(AFI). The main point in the result is that `LiCoRNA` can predict the pairwise alignments for all the families and generally shows equivalent or better results than its competitors for almost all the families. The second part is that we curate RFAM full pseudoknotted alignments using `LiCoRNA`. RFAM full alignments are constructed by `INFERNAL` which is based on the covariance model. However, covariance model does not model RNA CROSSING structure. Therefore, we hope that a systematic realignment will allow to reveal or refute an evolutionary pressure towards the preservation of a functional pseudoknot.The metrics are base pair conservation for each base pairs in the reference structure and the sequence identity. The hypothesis is that an alignment is considered better when its predicted structure has more base pairs, and its sequence identity is comparable to that of the RFAM alignment. The result illustrates that `LiCoRNA` supports pseudoknots without losing the sequence identity.

5.2 Perspectives

5.2.1 Score scheme of LiCoRNA

There are two other score schemes that can be applied to LiCoRNA. The first is profile-based score scheme. The score scheme for LiCoRNA is position-independent which means that the scores for substitutions and gaps are the same in different positions of RNA. Even though we give different gap open penalties and extension penalties for single stranded region and stack region, the gap penalties for the positions of either region are still the same. In other words, from a probabilistic point of view, it assumes that each evolutionary change occurs with the same probability for all the positions. Obviously, the assumption is not right. However, it seems hard to find another way to score pairwise alignment given one query sequence. However, once the homologous sequences of the query sequence are at hand, we can use a profile-based score scheme where position dependent log-odds scores can be calculated from the homologous sequences. Therefore, it is possible to check some positions are conserved which means that mutations and indels occur with less probability and some positions are less important for the structure or function of the ncRNA which means that mutations can happen with higher probability. This idea has been implemented in many programs such as HMMER which is based on hidden Markov Models and INFERNAL which is based on Covariance Models.

The idea of the profile is first introduced by Gribskov [36]. The profile analysis in [36] includes two steps: (1) construct the profile from the multiple alignment of homologs. Gribskov used an $N \times 21$ matrix (for proteins) to store the position-specific scores for substitutions and indels. The length of the multiple aligned query sequences is denoted as N. The first 20 columns store the score for finding each of the 20 amino acid residues and the 21st column are used to store the score of insertion and deletion. (2) Compare the profile with a single sequence which takes the same amount of time as pairwise alignment using DP. In our case, for arc-altering operation and arc-removing operation, we considered both arc ends in an alignment independently. Therefore, the position-specific score scheme for structure-sequence alignment for RNA can follow the similar strategy as Gribskov. For different unpaired positions, the 4×4 base substitution RIBOSUM matrix is multiplied by different weights and for different base pairing positions, the 16×16 base pair substitution RIBOSUM matrix is multiplied

by different weights. Finally, For different positions, indels penalty is multiplied by different weights.

The major advantage of profile is the detection of distantly related homology as proved by Gribskov [36]. However, profile approach also has disadvantage which requires computing more scoring parameters compared with pairwise alignment.

Secondly, even though profile approaches can improve the accuracy, it also assumes that the training sequences are statistically independent, however in fact they have phylogenetic relationship [66]. Therefore, it is more reasonable to use the probabilistic evolutional model to parameterize Rinaudo's algorithm. This is an active research area [80, 44] where important challenges still exist, especially how to model indels in the evolutional process.

5.2.2 Searching conserved structures in genomes

One of the purposes of structure-sequence alignment is to search for ncRNAs in genomes [24]. That is to say, one program takes any RNA sequence with structure (or any multiple aligned RNA sequences with consensus structure) as the query, and uses a position-independent score scheme (profile-based score scheme) to search a sequence database and get the alignments with high score. `LiCoRNA` is a powerful tool that can search any type of pseudoknots in the sequence database. The computational complexity is the major challenge for `LiCoRNA` when the query structure has complex pseudoknotted structure. Applying Rinaudo's algorithm needs $\mathcal{O}(|\mathcal{X}|\cdot n^{tw+1})$ ($\mathcal{O}(|\mathcal{X}|\cdot n^{tw})$ for smooth tree decomposition) time where $|\mathcal{X}|$ is the number of bags in the tree decomposition, tw is the tree-width and n is the length of target sequence.

The same problem also appears for `INFERNAL`. Nawrocki and Eddy [66] developed an acceleration method called Query-Dependent Banding (QDB) which reduced the complexity of the alignment based on the covariance models from $\mathcal{O}(LN^{2.4})$ to $\mathcal{O}(LN^{1.3})$ where N and L represent the length of query and target RNA. Besides, Weinberg *et al.* also developed several filters for the searches based on covariance models [108, 109, 110]. These filters include profile HMM based filters and covariance model based filters which has proved to accelerate the dynamic programming for covariance models without loss of accuracy.

Inspired by [110] and [63], we can have other methods to determine the bands for banded DP as mentioned in Chapter 3. The method consists of four steps:

- Ignore the structure of query structure and calculate the posterior probability of each match (i, j) between two sequences using the partition function where i, j are the indices of two sequences, separately [63].
- Determine the band for each nucleotide in the query sequence according to the threshold for the negligible match probability.
- Reconsider the structure of the query structure and set the band for each nucleotide in the query structure according to the band for each nucleotide set by sequence-sequence alignment.
- Align the target sequence to query structure using Rinaudo's algorithm.

Another method that I have to mention is Song's algorithm [91] in Chapter 2. The complexity for Song's algorithm is $\mathcal{O}(|\mathcal{X}| \cdot k^{tw})$ where k(usually 7) is the maximum number of base pairing regions for each stem, tw is the tree width of the tree decomposition and $|\mathcal{X}|$ is the number of bags in the tree decomposition. Compared with Song's algorithm, the complexity of Rinaudo's algorithm is $\mathcal{O}(|\mathcal{X}| \cdot n^{tw+1})$ ($\mathcal{O}(|\mathcal{X}| \cdot n^{tw})$ for smooth tree decomposition) where n is the length of the target sequence which is usually more than 100. Song's algorithm seems to reduce the computation time. However, Song's algorithm is a heuristic method and one may miss the optimal alignments. By taking other methods to improve the accuracy, a reimplementation of Song's algorithm in `LiCoRNA` for searching conserved structure may be a good choice.

5.2.3 Identification of RNA 3D motifs

Many studies have shown that RNA tertiary structure is modular and composed of recurrent conserved motifs that correspond to the hairpin loops (HL), internal loops (IL) and multi-helix junction loops (MHJ) one sees in RNA secondary structure [4, 111, 94, 97]. The 3D motifs are stabilized by base-pairing, base-stacking and base-backbone interactions. Among the interactions, base-pairing is the most specific [94]. As defined in [56], there are 12 main families of base pairs, because each base (purine and pyrimidine) has three interactions edges, Watson-Crick edge(WC), Sugar edge(SE), and Hoogsteen edge(H) and the interaction can be in either *cis* or *trans* orientations. Figure 5.1 illustrates two common structural motifs, Kink-turn motif and C-loop motif. Kink-turn motif appears in the internal loop in secondary

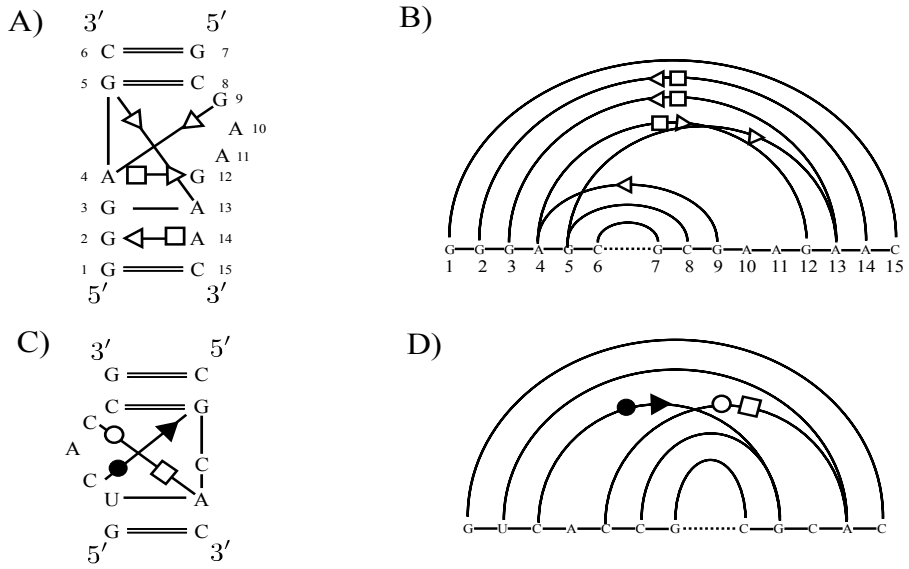
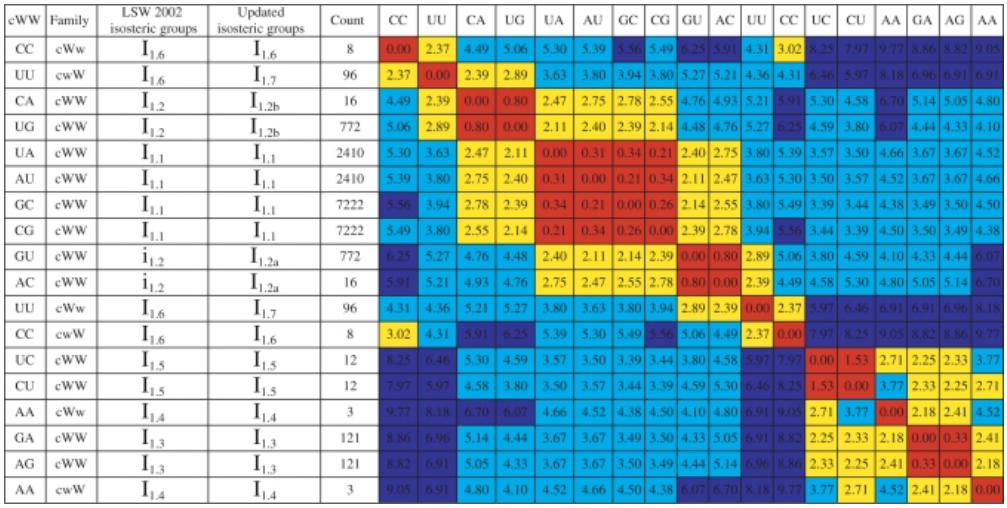


Figure 5.1: (A) 2D diagram for base-pairing patterns for Kink-turn motif. (the detail of the notation in [56]) (B) arc representation of Kink-turn motif. Dash line means that left strand and right strand are separated in 3D structure. (C) 2D diagram for base-pairing patterns for C-loop motif. (D) arc representation of C-loop motif.

structure and produces a sharp bend in tertiary structure. C-loop motif also appears in the internal loop and is involved in tertiary interaction in the stem-loop structure.

Theoretically, tree decomposition-based DP works in identification of RNA 3D motifs for several reasons. The first is that some motifs are defined by their own non-canonical base-pairing pattern. Let us take Kink-turn motif as an example shown in Figure 5.1 (A). There are five base pairs characterizing the motif [57]. (1) base pair between positions 5 and 8 is usually canonical $cis-$WC/WC base pair. (2) base pair between positions 4 and 12, base pair between positions 3 and 13 are tandem $trans-$H/SE. (3) base pair between positions 5 and 13 is $trans-$SE/SE. (4) base pair between positions 4 and 9 is also $trans-$SE/SE. Therefore, modeling motifs by their base-pairing pattern is a good idea as shown in Figure 5.1 (B) and (D) which are the arc representations [123] of the 3D motifs . The arc representation is similar to the arc-annotated representation for RNA secondary structure. However, we need to distinguish different kinds of base pairs in the 12 geometric families.

The second reason is that tree decomposition-based DP can model the relationship of base pairs very well, especially multi-pairing and crossing base pairs. Multi-pairing means that one nucleotide is involved in more than one base pair, like the positions



cWW	Family	LSW 2002 isosteric groups	Updated isosteric groups	Count	CC	UU	CA	UG	UA	AU	GC	CG	GU	AC	UU	CC	UC	CU	AA	GA	AG	AA
CC	cWw	$I_{1.6}$	$I_{1.6}$	8	0.00	2.37	4.49	5.06	5.30	5.39	5.56	5.49	6.25	5.91	4.31	3.02	8.25	7.97	9.77	8.86	8.82	9.05
UU	cwW	$I_{1.6}$	$I_{1.7}$	96	2.37	0.00	2.39	2.89	3.63	3.80	3.94	3.80	5.27	5.21	4.36	4.31	6.46	5.97	8.18	6.96	6.91	6.91
CA	cWW	$I_{1.2}$	$I_{1.2b}$	16	4.49	2.39	0.00	0.80	2.47	2.75	2.78	2.55	4.76	4.93	5.21	5.91	5.30	4.58	6.70	5.14	5.05	4.80
UG	cWW	$I_{1.2}$	$I_{1.2b}$	772	5.06	2.89	0.80	0.00	2.11	2.40	2.39	2.14	4.48	4.76	5.27	6.25	4.59	3.80	6.07	4.44	4.33	4.10
UA	cWW	$I_{1.1}$	$I_{1.1}$	2410	5.30	3.63	2.47	2.11	0.00	0.31	0.34	0.21	2.40	2.75	3.80	5.39	3.57	3.50	4.66	3.67	3.67	4.52
AU	cWW	$I_{1.1}$	$I_{1.1}$	2410	5.39	3.80	2.75	2.40	0.31	0.00	0.21	0.34	2.11	2.47	3.63	5.30	3.50	3.57	4.52	3.67	3.67	4.66
GC	cWW	$I_{1.1}$	$I_{1.1}$	7222	5.56	3.94	2.78	2.39	0.34	0.21	0.00	0.26	2.14	2.55	3.80	5.49	3.39	3.44	4.38	3.49	3.50	4.50
CG	cWW	$I_{1.1}$	$I_{1.1}$	7222	5.49	3.80	2.55	2.14	0.21	0.34	0.26	0.00	2.39	2.78	3.94	5.56	3.44	3.39	4.50	3.50	3.49	4.38
GU	cWW	$i_{1.2}$	$I_{1.2a}$	772	6.25	5.27	4.76	4.48	2.40	2.11	2.14	2.39	0.00	0.80	2.89	5.06	3.80	4.59	4.10	4.33	4.44	6.07
AC	cWW	$i_{1.2}$	$I_{1.2a}$	16	5.91	5.21	4.93	4.76	2.75	2.47	2.55	2.78	0.80	0.00	2.39	4.49	4.58	5.30	4.80	5.05	5.14	6.70
UU	cWw	$I_{1.6}$	$I_{1.7}$	96	4.31	4.36	5.21	5.27	3.80	3.63	3.80	3.94	2.89	2.39	0.00	2.37	5.97	6.46	6.91	6.91	6.96	8.18
CC	cwW	$I_{1.6}$	$I_{1.6}$	8	3.02	4.31	5.91	6.25	5.39	5.30	5.49	5.56	5.06	4.49	2.37	0.00	7.97	8.25	9.05	8.82	8.86	9.77
UC	cWW	$I_{1.5}$	$I_{1.5}$	12	8.25	6.46	5.30	4.59	3.57	3.50	3.39	3.44	3.80	4.58	5.97	7.97	0.00	1.53	2.71	2.25	2.33	3.77
CU	cWW	$I_{1.5}$	$I_{1.5}$	12	7.97	5.97	4.58	3.80	3.50	3.57	3.44	3.39	4.59	5.30	6.46	8.25	1.53	0.00	3.77	2.33	2.25	2.71
AA	cWw	$I_{1.4}$	$I_{1.4}$	3	9.77	8.18	6.70	6.07	4.66	4.52	4.38	4.50	4.10	4.80	6.91	9.05	2.71	3.77	0.00	2.18	2.41	4.52
GA	cWW	$I_{1.3}$	$I_{1.3}$	121	8.86	6.96	5.14	4.44	3.67	3.67	3.49	3.50	4.33	5.05	6.91	8.82	2.25	2.33	2.18	0.00	0.33	2.41
AG	cWW	$I_{1.3}$	$I_{1.3}$	121	8.82	6.91	5.05	4.33	3.67	3.67	3.50	3.49	4.44	5.14	6.96	8.86	2.33	2.25	2.41	0.33	0.00	2.18
AA	cwW	$I_{1.4}$	$I_{1.4}$	3	9.05	6.91	4.80	4.10	4.52	4.66	4.50	4.38	6.07	6.70	8.18	9.77	3.77	2.71	4.52	2.41	2.18	0.00

Figure 5.2: IDI matrix for the *cis*-WC/WC family, as featured in [93]. Cells are colored in this matrix. (1) red: isosteric base pair (IDI $\leq$ 2.0); (2) yellow: near isosteric base pairs (2.0 $<$ IDI $\leq$ 3.3); (3) cyan: non-isosteric base pairs (3.3 $<$ IDI $\leq$ 5.0); (4) blue: very different base pairs (5.0 $<$ IDI).

4 and 5 in Figure 5.1 (B).

The third reason is that the definition of isostericity [56] and the IsoDiscrepancy Index (IDI) [93] provides the information to determine which base pair substitution may occur for a given 3D motif. In the evolutional process, the 3D structure of RNA homologous sequences changes more slowly than the sequence. When one base pair is replaced by another kind of base pair without drastically disturbing the geometry of the backbone, we denote such base pairs as 'isosteric'. Isostericity can be quantified using IDI with the fact that two base pairs with sufficiently low IDI can be considered isosteric. The dataset they used to calculate the IDI values is from RNA 3D structures in the Protein Data Bank (PDB). Figure 5.2 illustrates IDI matrix for *cis*-WC/WC base pair. The Watson-Crick base pairs are in red cells which represent that they are isosteric base pairs and the wobble base pairs are in yellow cells which represent that they are near isosteric base pairs.

The fourth reason is that tree decomposition-based DP algorithm is efficient in this case. Though the complexity of the algorithm does not change, the length of the query structure and target sequence are small.

After showing the reasons, the general idea to identify the 3D motifs is shown as follows:

- For the **query structural motif**, we transfer it into the arc representation as shown in Figure 5.1.
- Given a PDB structure, we use RNAVIEW [119] to generate a 2-dimensional display of RNA/DNA structure with tertiary interactions. Then we cut the targets structure into many local structural segments [123]. We treat them as **target structural segments**.
- We transfer the arc representation of the query structural motif as RNA query graph which can be decomposed by tree decomposition.
- A tree decomposition based dynamic programming method can be used to search for the **target structural motifs** in the target structural segments with high scores.
- The corresponding target structural motifs can be treated as the same kind of motif as the query structural motif.

Appendix A

Table A.1: 86 Pseudoknotted RFAM families and their corresponding RNA names.

RFAM	RNA name
RF00009	Nuclear RNase P
RF00010	Ribonuclease P class A
RF00011	Ribonuclease P class B
RF00023	Transfer-messenger RNA
RF00024	Telomerase RNA component
RF00028	Group I catalytic intron
RF00030	RNase MRP
RF00041	Enteroviral 3' UTR element
RF00094	HDV ribozyme
RF00140	Alpha operon ribosome binding site
RF00165	Coronavirus 3' UTR pseudoknot
RF00176	Tombusvirus 3' UTR region IV
RF00216	c-myc internal ribosome entry site (IRES)
RF00233	tymoviruses/pomovirusesfamily tRNA-like 3' UTR element
RF00259	Interferon gamma 5' UTR regulatory element
RF00261	L-myc internal ribosome entry site (IRES)
RF00373	Archaeal RNase P
RF00381	Antizyme RNA frameshifting stimulation element
RF00390	UPSK RNA
RF00458	Cripavirus internal ribosome entry site (IRES)
RF00499	Human parechovirus 1 (HPeV1) cis regulatory element (CRE)
RF00505	RydC RNA
RF00507	Coronavirus frameshifting stimulation element
RF00622	Mammalian CPEB3 ribozyme
RF01050	Saccharomyces telomerase

Continued on next page

Table A.1 – continued from previous page

RFAM	RNA name
RF01072	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01073	Gag/pol translational readthrough site
RF01074	Putative RNA-dependent RNA polymerase ribosomal frameshift site
RF01075	Pseudoknot of tRNA-like structure
RF01076	Polymerase ribosomal frameshift site
RF01077	Pseudoknot of tRNA-like structure
RF01078	3'-terminal pseudoknot in PYVV
RF01079	Putative RNA-dependent RNA polymerase ribosomal frameshift site
RF01080	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01081	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01082	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01083	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01084	Pseudoknot of tRNA-like structure
RF01085	Pseudoknot of tRNA-like structure
RF01087	Pseudoknot of the regulatory region of the repZ gene
RF01088	Pseudoknot of tRNA-like structure
RF01089	Pseudoknot of the regulatory region of the repBA gene
RF01090	Edr gene ribosomal frameshift signal
RF01091	3'-terminal pseudoknot in SPCSV
RF01092	Gag/pol translational readthrough site
RF01093	Ma3 gene ribosomal frameshift signal
RF01094	Polymerase ribosomal frameshift site
RF01095	3'-terminal pseudoknot of CuYV/BPYV
RF01096	HepA virus 3'-terminal pseudoknot
RF01097	Gag/pro ribosomal frameshift site
RF01098	Gag/pro ribosomal frameshift site
RF01099	Pseudoknot of influenza A virus gene
RF01100	3'-terminal pseudoknot in BYV
RF01101	Pseudoknot of tRNA-like structure
RF01102	5'-leader pseudoknot of TEV/CVMV
RF01103	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01104	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01105	UPSK RNA
RF01106	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01107	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01108	UPSK RNA

Continued on next page

Table A.1 – continued from previous page

RFAM	RNA name
RF01109	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01111	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01113	UPSK RNA
RF01114	Pseudoknot of upstream pseudoknot domain (UPD) of the 3'UTR
RF01577	Plasmodium RNase_P
RF01689	AdoCbl variant RNA
RF01704	Downstream peptide RNA
RF01715	Pedo-repair RNA
RF01725	SAM-I/IV variant riboswitch
RF01726	SAM-II long loop
RF01735	epsC RNA
RF01745	manA RNA
RF01761	wcaG RNA
RF01768	ribosomal frameshift site
RF01775	RNA S.aureus Orsay G
RF01785	ribosomal frameshift site
RF01786	Cyclic di-GMP-II riboswitch
RF01788	drz-agam-2-2 ribozyme
RF01807	GIR1 branching ribozyme
RF01831	THF riboswitch
RF01833	ribosomal frameshift site
RF01834	ribosomal frameshift site
RF01840	ribosomal frameshift element
RF01849	Alphaproteobacteria transfer-messenger RNA
RF01850	Betaproteobacteria transfer-messenger RNA

Bibliography

[1] Peter L Adams, Mary R Stahley, Michelle L Gill, Anne B Kosek, Jimin Wang, and Scott A Strobel. Crystal structure of a group I intron splicing intermediate. *RNA*, 10(12):1867–1887, 2004.

[2] Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1):45–62, 2000.

[3] Vineet Bafna, S Muthukrishnan, and R Ravi. Computing similarity between RNA strings. In *Combinatorial Pattern Matching*, pages 1–16. Springer, 1995.

[4] Robert T Batey, Robert P Rambo, Jennifer A Doudna, et al. Tertiary motifs in RNA structure and folding. *Angewandte Chemie International Edition*, 38(16):2326–2343, 1999.

[5] Emmanuelle Becker, Aurélie Cotillard, Vincent Meyer, Hocine Madaoui, and Raphaël Guérois. HMM-Kalign: a tool for generating sub-optimal HMM alignments. *Bioinformatics*, 23(22):3095–3097, 2007.

[6] Stephan H Bernhart, Ivo L Hofacker, Sebastian Will, Andreas R Gruber, and Peter F Stadler. RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics*, 9(1):474, 2008.

[7] Anne Berry, Pinar Heggernes, and Genevieve Simonet. The minimum degree heuristic and the minimal triangulation process. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 58–70. Springer, 2003.

[8] Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, and Heleene Touzet. Alignments of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 7(2):309–322, 2010.

[9] Guillaume Blin and Hélene Touzet. How to compare arc-annotated sequences: The alignment hierarchy. In *International Symposium on String Processing and Information Retrieval*, pages 291–303. Springer, 2006.

[10] Hans L. Bodlaender. *Classes of graphs with bounded tree-width*, volume 86. Department of Information and Computing Sciences, Utrecht University, 1986.

[11] Hans L Bodlaender and Arie MCA Koster. Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):259–275, 2010.

[12] Hans L Bodlaender and Arie MCA Koster. Treewidth computations II. Lower bounds. *Information and Computation*, 209(7):1103–1119, 2011.

[13] Sydney Brenner, François Jacob, and Matthew Meselson. An unstable intermediate carrying information from genes to ribosomes for protein synthesis. *Nature*, 190(4776):576–581, 1961.

[14] David A Brow and Harry F Noller. Protection of ribosomal RNA from kethoxal in polyribosomes: Implication of specific sites in ribosome function. *Journal of Molecular Biology*, 163(1):27–46, 1983.

[15] Jamie J Cannone, Sankar Subramanian, Murray N Schnare, James R Collett, Lisa M D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V Madabusi, Kirsten M Müller, et al. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3(1):2, 2002.

[16] Song Cao and Shijie Chen. Predicting structures and stabilities for H-type pseudoknots with interhelix loops. *RNA*, 15(4):696–706, 2009.

[17] Cedric Chauve, Julien Courtiel, and Yann Ponty. An unambiguous and complete dynamic programming algorithm for tree alignment. *arXiv preprint arXiv:1505.05983*, 2015.

[18] Anne Condon, Beth Davy, Baharak Rastegari, Shelly Zhao, and Finbarr Tarrant. Classifying rna pseudoknotted structures. *Theoretical Computer Science*, 320(1):35–50, 2004.

[19] Francis HC Crick. On protein synthesis. In *Symposia of the Society for Experimental Biology*, volume 12, page 8, 1958.

[20] Ralf Dahm. Friedrich miescher and the discovery of DNA. *Developmental Biology*, 278(2):274–288, 2005.

[21] Yves Van de Peer, Ilse Van den Broeck, Peter De Rijk, and Rupert De Wachter. Database on the structure of small ribosomal subunit RNA. *Nucleic Acids Research*, 22(17):3488–3494, 1994.

[22] Rina Dechter and Judea Pearl. Tree-clustering schemes for constraint-processing. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, pages 150–154. AAAI Press, 1988.

[23] Katherine E Deigan, Tian W Li, David H Mathews, and Kevin M Weeks. Accurate shape-directed RNA structure determination. *Proceedings of the National Academy of Sciences*, 106(1):97–102, 2009.

[24] Alain Denise and Philippe Rinaudo. Optimisation problems for pairwise RNA sequence and structure comparison: a brief survey. *Transactions on Computational Collective Intelligence*, 13:70–82, 2014.

[25] Liang Ding, Xingran Xue, Sal LaMarca, Mohammad Mohebbi, Abdul Samad, Russell L Malmberg, and Liming Cai. Ab initio prediction of RNA nucleotide interactions with backbone k-tree model. *arXiv preprint arXiv:1407.7080*, 2014.

[26] Ye Ding and Charles E Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Research*, 31(24):7280–7301, 2003.

[27] Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of Computational Chemistry*, 24(13):1664–1677, 2003.

[28] Chuong B Do, Mahathi SP Mahabhashyam, Michael Brudno, and Serafim Batzoglou. Probcons: probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340, 2005.

[29] Martin Egli, George Minasov, Li Su, and Alexander Rich. Metal ions and flexibility in a viral RNA pseudoknot at atomic resolution. *Proceedings of the National Academy of Sciences*, 99(7):4302–4307, 2002.

[30] Patricia Anne Evans. *Algorithms and complexity for annotated sequence analysis*. PhD thesis, University of Victoria, 1999.

[31] Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics*, 5(1):140, 2004.

[32] Robert Giegerich. Explaining and controlling ambiguity in dynamic programming. In *CPM*, volume 1848, pages 46–59. Springer, 2000.

[33] Walter Gilbert. Origin of life: The RNA world. *Nature*, 319(6055), 1986.

[34] Jan Gorodkin, Laurie J Heyer, and Gary D Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Research*, 25(18):3724–3732, 1997.

[35] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982.

[36] Michael Gribskov, Andrew D McLachlan, and David Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84(13):4355–4358, 1987.

[37] Francois Gros, Howard Hiatt, Walter Gilbert, Chuck G Kurland, RW Risebrough, and James D Watson. Unstable ribonucleic acid revealed by pulse labelling of Escherichia coli. *Nature*, 190(4776):581–585, 1961.

[38] Cecilia Guerrier-Takada, Katheleen Gardiner, Terry Marsh, Norman Pace, and Sidney Altman. The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, 35(3):849–857, 1983.

[39] Buhm Han, Banu Dost, Vineet Bafna, and Shaojie Zhang. Structural alignment of pseudoknotted RNA. *Journal of Computational Biology*, 15(5):489–504, 2008.

[40] Arif Ozgun Harmanci, Gaurav Sharma, and David H Mathews. Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics*, 8(1):130, 2007.

[41] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[42] Mahlon B Hoagland, Mary Louise Stephenson, Jesse F Scott, Liselotte I Hecht, and Paul C Zamecnik. A soluble ribonucleic acid intermediate in protein synthesis. *Journal of Biological Chemistry*, 231(1):241–257, 1958.

[43] Matthias Hochsmann, Thomas Toller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structures. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 159–168. IEEE, 2003.

[44] Ian Holmes. A probabilistic model for the evolution of RNA structure. *BMC Bioinformatics*, 5(1):166, 2004.

[45] Ian Holmes and Richard Durbin. Dynamic programming alignment accuracy. *Journal of Computational Biology*, 5(3):493–504, 1998.

[46] Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics, 2005.

[47] Irnov Irnov and Wade C Winkler. A regulatory RNA required for antitermination of biofilm and capsular polysaccharide operons in bacillales. *Molecular Microbiology*, 76(3):559–575, 2010.

[48] Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–388, 2002.

[49] Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees—an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[50] Horace Freeland Judson and Walter Gratzer. *The eighth day of creation*. Cold Spring Harbor Laboratory Press,U.S., 1996.

[51] Uffe Kjærulff. Triangulation of graphs–algorithms giving small total state space. *Technical Report*, 1990.

[52] Robert J Klein and Sean R Eddy. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4(1):44, 2003.

[53] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[54] David J Lane, Bernadette Pace, Gary J Olsen, David A Stahl, Mitchell L Sogin, and Norman R Pace. Rapid determination of 16s ribosomal RNA sequences for phylogenetic analyses. *Proceedings of the National Academy of Sciences*, 82(20):6955–6959, 1985.

[55] Neocles B Leontis, Jesse Stombaugh, and Eric Westhof. The non-Watson-Crick base pairs and their associated isostericity matrices. *Nucleic Acids Research*, 30(16):3497–3531, 2002.

[56] Neocles B Leontis and Eric Westhof. Geometric nomenclature and classification of RNA base pairs. *Nucleic Acids Research*, 7(4):499–512, 2001.

[57] Aurelie Lescoute, Neocles B Leontis, Christian Massire, and Eric Westhof. Recurrent structural RNA motifs, isostericity matrices and sequence alignments. *Nucleic Acids Research*, 33(8):2395–2409, 2005.

[58] Rune B Lyngsø and Christian NS Pedersen. Pseudoknots in RNA secondary structures. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 201–209. ACM, 2000.

[59] Rune B Lyngsø and Christian NS Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3-4):409–427, 2000.

[60] Hiroshi Matsui, Kengo Sato, and Yasubumi Sakakibara. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics*, 21(11):2611–2617, 2005.

[61] John S McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–1119, 1990.

[62] Sanzo Miyazawa. A reliable sequence alignment method based on probabilities of residue correspondences. *Protein Engineering, Design and Selection*, 8(10):999–1009, 1995.

[63] Ulrike Mückstein, Ivo L Hofacker, and Peter F Stadler. Stochastic pairwise alignments. *Bioinformatics*, 18(suppl_2):S153–S160, 2002.

[64] Dalit Naor and Douglas Brutlag. On suboptimal alignments of biological sequences. In *Combinatorial Pattern Matching*, pages 179–196. Springer, 1993.

[65] Eric P Nawrocki, Sarah W Burge, Alex Bateman, Jennifer Daub, Ruth Y Eberhardt, Sean R Eddy, Evan W Floden, Paul P Gardner, Thomas A Jones, John Tate, et al. Rfam 12.0: updates to the RNA families database. *Nucleic Acids Research*, page gku1063, 2014.

[66] Eric P Nawrocki and Sean R Eddy. Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Computational Biology*, 3(3):e56, 2007.

[67] Eric P Nawrocki and Sean R Eddy. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935, 2013.

[68] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[69] Marc Neveu, Hyo-Joong Kim, and Steven A Benner. The strong RNA world hypothesis: fifty years old. *Astrobiology*, 13(4):391–403, 2013.

[70] Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.

[71] Mathilde Paris, Tommy Kaplan, Xiao Yong Li, Jacqueline E Villalta, Susan E Lott, and Michael B Eisen. Extensive divergence of transcription factor binding in Drosophila embryos with highly conserved gene expression. *PLoS Genet*, 9(9):e1003748, 2013.

[72] Anton I Petrov. *RNA 3D motifs: identification, clustering, and analysis*. PhD thesis, Bowling Green State University, 2012.

[73] Comelis WA Pleij, Krijn Rietveld, and Leendert Bosch. A new principle of RNA folding based on pseudoknotting. *Nucleic Acids Research*, 13(5):1717–1731, 1985.

[74] Chris P Ponting, Peter L Oliver, and Wolf Reik. Evolution and functions of long noncoding RNAs. *Cell*, 136(4):629–641, 2009.

[75] Yann Ponty and Cédric Saule. A combinatorial framework for designing (pseudoknotted) RNA algorithms. In *WABI*, volume 6833, pages 250–269. Springer, 2011.

[76] Uttam L RajBhandary and Dieter Söll. *tRNA: Structure, biosynthesis, and function*. American Society for Microbiology, 1994.

[77] Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5(1):104, 2004.

[78] Philippe Rinaudo, Yann Ponty, Dominique Barth, and Alain Denise. Tree decomposition and parameterized algorithms for RNA structure-sequence alignment including tertiary interactions and pseudoknots. In *WABI*, pages 149–164. Springer, 2012.

[79] Elena Rivas and Sean R Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, 285(5):2053–2068, 1999.

[80] Elena Rivas and Sean R Eddy. Parameterizing sequence alignment with an explicit evolutionary model. *BMC Bioinformatics*, 16(1):406, 2015.

[81] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.

[82] Peter W Rose, Andreas Prlić, Ali Altunkaya, Chunxiao Bi, Anthony R Bradley, Cole H Christie, Luigi Di Costanzo, Jose M Duarte, Shuchismita Dutta, Zukang Feng, et al. The RCSB protein data bank: integrative view of protein, gene and 3D structural information. *Nucleic Acids Research*, page gkw1000, 2016.

[83] Yasubumi Sakakibara. Pair Hidden Markov models on tree structures. *Bioinformatics*, 19(suppl_1):i232–i240, 2003.

[84] Jiri Sana, Petra Faltejskova, Marek Svoboda, and Ondrej Slaby. Novel classes of non-coding RNAs and cancer. *Journal of Translational Medicine*, 10(1):103, 2012.

[85] David Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985.

[86] Michael Sarver, Craig L Zirbel, Jesse Stombaugh, Ali Mokdad, and Neocles B Leontis. FR3D: finding local and composite recurrent structural motifs in RNA 3D structures. *Journal of Mathematical Biology*, 56(1):215–252, 2008.

[87] Cédric Saule, Mireille Régnier, Jean-Marc Steyaert, and Alain Denise. Counting RNA pseudoknotted structures. *Journal of Computational Biology*, 18(10):1339–1351, 2011.

[88] Stefan E Seemann, Peter Menzel, Rolf Backofen, and Jan Gorodkin. The PETfold and PETcofold web servers for intra-and intermolecular structures of multiple RNA sequences. *Nucleic Acids Research*, 39(suppl_2):W107–W111, 2011.

[89] Sven Siebert and Rolf Backofen. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21(16):3352–3359, 2005.

[90] Sandra Smit, Jeremy Widmann, and Rob Knight. Evolutionary rates vary among rRNA structural elements. *Nucleic Acids Research*, 35(10):3339–3354, 2007.

[91] Yinglei Song, Chunmei Liu, Russell Malmberg, Fangfang Pan, and Liming Cai. Tree decomposition based fast search of RNA structures including pseudoknots in genomes. In *Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE*, pages 223–234. IEEE, 2005.

[92] Peter Steffen and Robert Giegerich. Versatile and declarative dynamic programming using pair algebras. *BMC Bioinformatics*, 6(1):224, 2005.

[93] Jesse Stombaugh, Craig L Zirbel, Eric Westhof, and Neocles B Leontis. Frequency and isostericity of RNA base pairs. *Nucleic Acids Research*, 37(7):2294–2312, 2009.

[94] Blake A Sweeney, Poorna Roy, and Neocles B Leontis. An introduction to recurrent nucleotide interactions in RNA. *Wiley Interdisciplinary Reviews: RNA*, 6(1):17–45, 2015.

[95] Robert E Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

[96] Michela Taufer, Abel Licon, Roberto Araiza, David Mireles, FHD Van Batenburg, Alexander P Gultyaev, and Ming-Ying Leung. Pseudobase++: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Research*, 37(suppl_1):D127–D135, 2008.

[97] Corinna Theis, Craig L Zirbel, Christian Höner Zu Siederdissen, Christian Anthon, Ivo L Hofacker, Henrik Nielsen, and Jan Gorodkin. RNA 3D modules in genome-wide predictions of RNA 2D structure. *PloS One*, 10(10):e0139900, 2015.

[98] Pilar Tijerina, Sabine Mohr, and Rick Russell. DMS footprinting of structured RNAs and RNA–protein complexes. *Nature Protocols*, 2(10):2608–2623, 2007.

[99] Ignacio Tinoco and Carlos Bustamante. How RNA folds. *Journal of Molecular Biology*, 293(2):271–281, 1999.

[100] Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210(2):277–303, 1999.

[101] FHD Van Batenburg, Alexander P Gultyaev, CWA Pleij, J Ng, and J Oliehoek. PseudoBase: a database with RNA pseudoknots. *Nucleic Acids Research*, 28(1):201–204, 2000.

[102] Thomas van Dijk, Jan-Pieter van den Heuvel, and Wouter Slob. Computing treewidth with libTW. *Citeseer. http://citeseerx. ist. psu. edu/viewdoc/download*, 2006.

[103] Martin Vingron. Near-optimal sequence alignment. *Current Opinion in Structural Biology*, 6(3):346–352, 1996.

[104] Kevin C Wang and Howard Y Chang. Molecular mechanisms of long noncoding RNAs. *Molecular Cell*, 43(6):904–914, 2011.

[105] Stefan Washietl, Ivo L Hofacker, and Peter F Stadler. Fast and reliable prediction of noncoding RNAs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(7):2454–2459, 2005.

[106] Michael S Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proceedings of the National Academy of Sciences*, 80(10):3123–3124, 1983.

[107] Michael S Waterman and Thomas H Byers. A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. *Mathematical Biosciences*, 77(1-2):179–188, 1985.

[108] Zasha Weinberg and Walter L Ruzzo. Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20(suppl_1):i334–i341, 2004.

[109] Zasha Weinberg and Walter L Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 243–251. ACM, 2004.

[110] Zasha Weinberg and Walter L Ruzzo. Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics*, 22(1):35–39, 2005.

[111] Eric Westhof and Pascal Auffinger. RNA tertiary structure. *Encyclopedia of Nnalytical Chemistry*, 2000.

[112] Kevin A Wilkinson, Edward J Merino, and Kevin M Weeks. Selective 2-hydroxyl acylation analyzed by primer extension (SHAPE): quantitative RNA structure analysis at single nucleotide resolution. *Nature Protocols*, 1(3):1610–1616, 2006.

[113] Sebastian Will, Tejal Joshi, Ivo L Hofacker, Peter F Stadler, and Rolf Backofen. LocARNA-P: accurate boundary prediction and improved detection of structural RNAs. *Nucleic Acids Research*, 18(5):900–914, 2012.

[114] Andreas Wilm, Desmond G Higgins, and Cédric Notredame. R-coffee: a method for multiple alignment of non-coding RNA. *Nucleic Acids Research*, 36(9):e52–e52, 2008.

[115] Andreas Wilm, Indra Mainz, and Gerhard Steger. An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms for Molecular Biology*, 1(1):19, 2006.

[116] Carl R Woese and Norman R Pace. Probing RNA structure, function, and history by comparative analysis. *Cold Spring Harbor Monograph Series*, 24:91–91, 1993.

[117] Thomas KF Wong, Tak Wah Lam, Wing-Kin Sung, Brenda WY Cheung, and Siu-Ming Yiu. Structural alignment of RNA with complex pseudoknot structure. *Journal of Computational Biology*, 18(1):97–108, 2011.

[118] Stefan Wuchty, Walter Fontana, Ivo L Hofacker, Peter Schuster, et al. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–165, 1999.

[119] Huanwang Yang, Fabrice Jossinet, Neocles Leontis, Li Chen, John Westbrook, Helen Berman, and Eric Westhof. Tools for the automatic identification and classification of RNA base pairs. *Nucleic Acids Research*, 31(13):3450–3460, 2003.

[120] Byung-Jun Yoon and Palghat P Vaidyanathan. Structural alignment of RNAs using profile-csHMMs and its application to RNA homology search: overview and new results. *IEEE Transactions on Automatic Control*, 53(Special Issue):10–25, 2008.

[121] Yi-Tao Yu, Elizabeth C Scharl, Christine M Smith, and Joan A Steitz. The growing world of small nuclear ribonucleoproteins. *Cold Spring Harbor Monograph Series*, 37:487–524, 1999.

[122] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[123] Cuncong Zhong, Haixu Tang, and Shaojie Zhang. RNAMotifScan: automatic identification of RNA structural motifs using secondary structural alignment. *Nucleic Acids Research*, 38(18):e176–e176, 2010.

[124] William A Ziehler and David R Engelke. Probing RNA structure with chemical reagents and enzymes. *Current Protocols in Nucleic Acid Chemistry*, pages 6–1, 2001.

[125] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

[126] Christian Zwieb, Iwona Wower, and Jacek Wower. Comparative sequence analysis of tmRNA. *Nucleic Acids Research*, 27(10):2063–2071, 1999.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Alignement pratique de structure-séquence d'ARN avec pseudonœuds

Mots clés : ARN non-codant, pseudonœuds, alignement sous-optimal, alignement avec la précision maximale attendue

Résumé: Aligner des macromolécules telles que des protéines, des ADN et des ARN afin de révéler ou exploiter leur homologie fonctionnelle est un défi classique en bioinformatique, qui offre de nombreuses applications.
Récemment, Rinaudo et al. ont donné un algorithme paramétré général pour la comparaison structure-séquence d'ARN. Nous avons développé plusieurs variantes et extensions de cet algorithme. Afin de l'accélérer sans perte sensible de précision, nous avons introduit une approche de programmation dynamique par bande. De plus, trois algorithmes ont été développés pour obtenir des alignements sous-optimaux. En outre, nous introduisons dans ce contexte la notion de MEA (Maximum-expected Structure-Alignment) pour calculer un alignement avec la précision maximale attendue sur un ensemble d'alignements. Tous ces algorithmes ont été implémentés dans un logiciel nommé LiCoRNA. Les performances de LiCoRNA ont été évaluées d'abord sur l'alignement des graines des familles de la base de données RFAM qui comportent des pseudo-noeuds. Comparé aux autres algorithmes de l'état de l'art, LiCoRNA obtient généralement des résultats équivalents ou meilleurs que ses concurrents. Grâce à la grande précision démontrée par LiCoRNA, nous montrons que cet outil peut être utilisé pour améliorer les alignements de certaines familles de RFAM qui comportent des pseudo-noeuds. Nos analyses des alignements complets des familles RFAM avec pseudonoeuds confirment cet écart de qualité, qui se trouve résorbé par un réalignement tenant compte des pseudonoeuds. Elles suggèrent donc l'utilisation systématique de méthodes prenant en charge les pseudonoeuds lors des futures itérations de RFAM.

Title : Practical structure-sequence alignment of pseudoknotted RNAs

Keywords : non-coding RNA, pseudoknots, suboptimal structure-sequence alignment, maximum expected accuracy alignment.

Abstract : Aligning macromolecules such as proteins, DNAs and RNAs in order to reveal their functional homology is a classic challenge in bioinformatics.
Recently, Rinaudo et al. gave a fully general parameterized algorithm for structure sequence comparison. The parameterized algorithm is a tree decomposition based dynamic programming (DP) algorithm. To accelerate the alignment process, we introduced a banded DP. Then three algorithms are introduced to explore either in a deterministic or in a stochastic way, the space of suboptimal or near-optimal alignments. Furthermore, we introduce maximum expected structure sequence alignment to compute the representative of a set of alignments. The algorithms are implemented in a software named LiCoRNA which is able to take as input any type of pseudoknotted structures. We first evaluate the performance of LiCoRNA on the seed alignments in the pseudoknotted RFAM families. Compared to the state-of-the-art algorithms, LiCoRNA shows generally equivalent or better results than its competitors. With the high accuracy showed by LiCoRNA, we further curate RFAM full pseudoknotted alignments. Our analyses of full alignments confirm this expected discrepancy, and its disappearance upon realignment with LiCoRNA, suggesting the use of pseudoknot-aware methods for further analysis of RFAM families.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France