Albert-Ludwigs-Universität
Freiburg

Master Thesis

Variations of the Sankoff-Algorithm with a Focus on Heuristics

Done by: Farah Majid Abdul Hameed

Born on: 4 May 1980

Under the chair of: Prof. Dr. Rolf Backofen

Direct Supervisor: Dr. Sebastian Will

15 June 2009

Fakultät für Angewandte Wissenschaften
Institut für Informatik
Lehrstuhl für Bioinformatik

Selbständigkeitserklärung

Hiermit erkläre ich, dass die hier vorliegende Masterarbeit von mir selbständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen erstellt wurde.

Freiburg, den

Unterschrift

Abstract

The combination of the alignment and secondary structure prediction solutions of two RNA sequences can significantly improve the accuracy of the structural predictions. The algorithm which simultaneously solves these problems tends to be computationally expensive like the original form “Sankoff Algorithm” [S85]. Thus, the methods which addressed this problem impose constraints that reduce the computational complexity by restricting the folding and/or alignment and thus make the Sankoff algorithm more practical.

In this thesis, reviewing the different Sankoff-style methods in such a way that compares them corresponding to the Sankoff algorithm, through the parallels and differences. As well as, the focus is on the heuristics (i.e. the imposed constraints on the alignments and/or the structures) and comparing between them.

In practical, the work discusses:

- Sankoff algorithm which is the original form of simultaneous Folding and Alignment of RNA sequences,
- Dynalign method which is the direct implementation of Sankoff algorithm,
- Foldalign method which is the first implementation of simultaneous Folding and Alignment of RNA sequences,
- PMcomp/LocARNA which are a simplification of the Sankoff algorithm.

In this work, practical results are obtained for three dataset examples of RNA families that have different sequence lengths, which indicate to the different influences on the speed of each program depending on the type and strength of each heuristic in these methods. Thus, the conclusion is combining some heuristics of the current methods in such a way that can improve the computation efficiency as well as accuracy as a new method or new versions of the current methods.

Acknowledgements

- This work would not have been possible without the support and encouragement of my supervisor *Dr. Sebastian Will*, under his supervision we chose this topic and began the thesis.

- I would like to thank *Prof. Dr. Rolf Backofen* the head of Bioinformatics team and the teaching staff and also members of the University of Freiburg.

- Special thanks go to DAAD whose gave me the chance to study the Master in Germany and for their support during whole my study time.

- I cannot end without thanking my family, on whose constant encouragement and love I have relied throughout my time at the Academy.

Contents

Chapter One

Introduction

1.1 Motivation

The dynamic programming comparison for the sequences finds widespread applications in the field of molecular biology, through the detection and evaluation of similarities among a number of nucleic acid (DNA, RNA) sequences, as well as protein sequences.

This thesis will concentrate on the RNA sequences, precisely for the RNA secondary structure which is essential for biological function. However, it is still difficult to determine experimentally the RNA structure. The most popular algorithm to predict the structure is the Minimum Free Energy (MFE) which folds a single sequence. This method has been implemented via Mfold [ZS81] and RNAfold [HFB+S94]. Nevertheless, the accuracy of MFE structure prediction is still restricted in practice. In general, the comparative methods [PTW99] are also used for determining RNA structure but at the best accuracy.

Actually, there are three automated approaches for analyzing RNA sequences and structures, which are illustrated below in Figure 1.1 [GG04]. For Plan A, the aligned sequences are obtained by using a standard multiple sequence alignment algorithms, such as ClustalW [THG94], t-coffee [NHH00], prrn [G99]. Then a consensus secondary structure is inferred by attempting to detect the covariation of base paired sites in the alignment. The mutual information measure is frequently used to this [CK91], [GPH+PS92] and [GHB+S97]. The tools which have been developed recently are used a combination of energetic and a covariance terms [HFS02], or evolutionary Stochastic Context-Free Grammar [KH03]. This plan is at the step of multiple sequence alignment, the well preserved sequence is produced. Although this shows a very successful approach, but it is still restricted with the sequence homology that should be a high enough in the alignment step that helps to find structurally the consistent mutations.

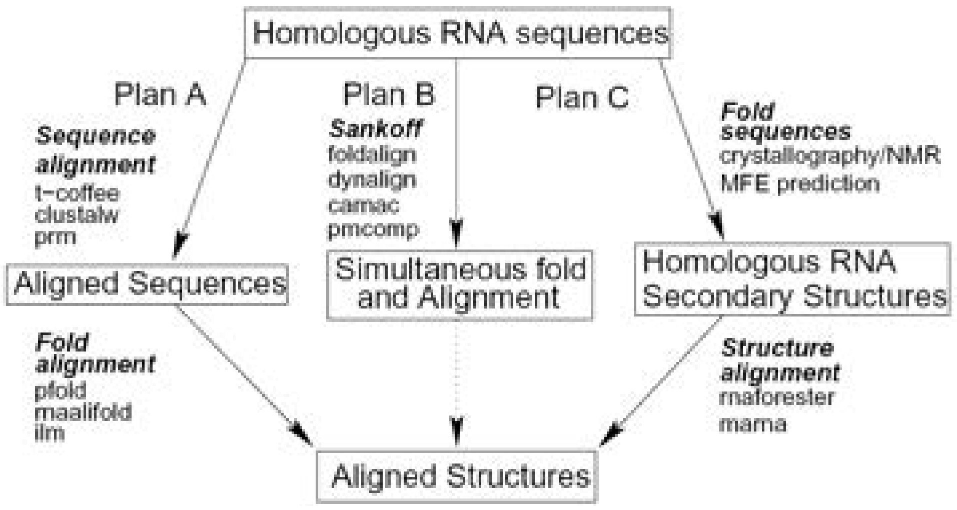


Figure 1.1: Three automated approaches for producing the aligned structure of RNA sequences. [GG04]

For plan B, the Sankoff algorithm was the first strict mathematical treatments of simultaneous alignment and folding for RNA sequences, which was proposed by David Sankoff [S85]. This algorithm is computationally expensive (i.e. it requires a lot of computational resources $\mathcal{O}(N^{3n})$ for time and $\mathcal{O}(N^{2n})$ for memory, where N is the sequence length and n is the number of sequences). Therefore, several restricted versions are implemented of the Sankoff algorithm impose some realistic constraints on the size and/or shape of the substructures to reduce the computational complexities for time and memory. These restricted versions can be divided into two groups according to which scheme are used:

1) The energy-based methods which also can be distinguished into two groups:

a) The methods implementing more or less a complete energy model of the folding part, such as Foldalign [GHS97a, GHS97b, HLS+G05, HTG07] which is the first implementation of simultaneous Folding and Alignment of RNA sequences and Dynalign [MT02, HSM07] which is the method of direct Sankoff algorithm.
b) The methods suppose that a structure model of the input sequences is given in the form of weights for each base pair, such as PMcomp [HBS04] and LocARNA [WRH+SB07] which are a simplification of the Sankoff algorithm.

There are other Sankoff-style methods based on the energy model, but in this thesis we are only interested in the above methods, in addition to showing the heuristics for each one.

2) The probabilistic methods that based on the Stochastic Context Free Grammars (SCFG) parameters which are estimated from multiple sequence alignments. These methods that we did not discussed in this thesis, like Stemloc [H05] which is a pairwise RNA structural alignment prediction program based on SCFG, it uses “fold” and “alignment” envelopes to reduce the computation and memory. There is another method known as also RNA structural alignment “Consan” [DE06].

Finally, plan C is represented by aligning RNA secondary structures rather than sequences. Due to the nature of the nested branching of RNA structures, these are appropriately represented as trees. Since there is no ability to find sequence conservation through alignment step, fold the sequences separately by using the methods of single sequence structure prediction and then directly align the result structures. There are several methods for aligning structures of RNA. In the measuring the similarity by edit operations, the structure comparisons have been generalized to trees ([T79], [S88], [SZ90], [ZSh89], [JLM+Z02], [WZ01], [SB03]), other methods align locally ([HTG+K03], [BW04]). Since the main weakness of this approach is the single sequence structure predictions are inaccurate in many times, that leads to affect on all further analyses. Hence, this approach is too strong to be used when the reliable structures are provided.

1.2 Contribution

As mentioned above, that the original form of simultaneous Folding and Alignment of RNA sequences is the Sankoff algorithm. Due to the expensive computations of this algorithm, some restricted versions of the Sankoff algorithm are implemented.

The main contribution of this thesis is to review the different Sankoff-style methods and concentrate on their heuristics which used to make the Sankoff algorithm applicable in practice. In addition to compare these methods corresponding to the original form “Sankoff Algorithm”, on the basis of the consideration of various aspects, such as the system scoring scheme, the computation time and memory requirements.

Furthermore, this thesis gives also the results which are obtained from running the programs of these methods under some special selected parameters (which represent the heuristics) of these programs to compare the speed of these methods, as well as to compare and analyze the identified heuristics.

1.3 Overview

Now an overview of the chapters that make up our work is given. Chapter 2 introduces the background of the RNA biology and the required preliminaries for the purposes of this thesis. Chapter 3 reviews the Sankoff-style methods and shows the points of the parallels and differences corresponding to the Sankoff algorithm. In Chapter 4, the used dataset examples of RNA families are given. In addition the obtained results from these methods are presented and discussed. Finally, Chapter 5 summarizes the conclusions of this thesis and gives an outlook for the future work in this area.

Chapter Two

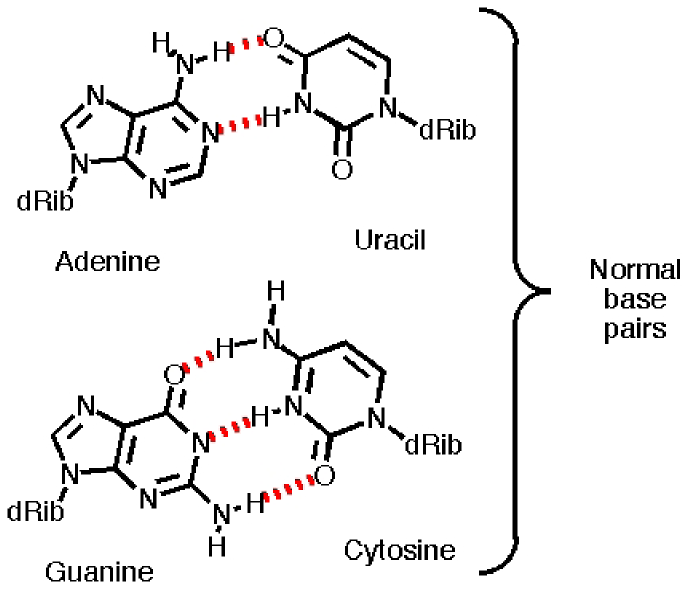
Background and preliminaries

This chapter will give the background and preliminaries needed for the purposes of this thesis, by introducing the methods of RNA sequence comparisons and the secondary structure prediction that represent the important information before going to the methods in the next chapter.

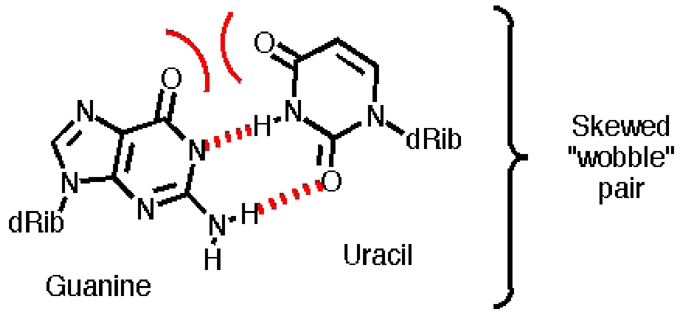
2.1 Biology of RNA

Ribonucleic Acid (RNA) is a single-stranded molecule which is composed of a long series of the linked nucleotides by the phosphodiester linkages. Each of these nucleotides is made up of a ribose sugar, a phosphate group and a nitrogenous base. There are four possible bases which are generally adenine (A), cytosine (C), guanine (G) and uracil (U). Due to the hydrogen bonds between the certain basepairs, a stable structure will be formed. These basepairs are formed between (C-G) and (A-U) of Watson-Crick basepairs and the Wobble basepairs between (G-U). Figure 2.1.a below shows these types of basepairs in RNA.

There are different kinds of RNA: messenger RNA (mRNA) carries information from DNA about a protein sequence to structures called ribosomes, Transfer RNA (tRNA) is a small RNA chain consisting of about 80 nucleotides that serves as adapter between mRNA and amino acids, Ribosomal RNA (rRNA) which is the main component of the ribosomes [WP08].



i) Watson-Crick basepairs.



ii) Wobble basepairs.

Figure 2.1.a: The RNA basepairs i) Watson-Crick basepairs, ii) Wobble basepairs. Image Source: “BC 5254/GCS 719, Computer Applications in Biomedical Research” http://www.finchcms.edu/cms/biochem/Walters/rna_folding.html.

An RNA structure is represented at different levels:

- Primary structure: It represents by a linear sequence of nucleotides over the alphabet $\sum = \{A, C, G, U\}$. These nucleotides (bases) are connected together by covalent phosphodiester bonds. Part (i) in Figure 2.1.b shows the primary structure.

- Secondary structure: As mentioned above due to the hydrogen bonds between the certain pairs of bases, a stable structure will be formed. The basepairs are formed by Watson-Crick base pairing G-C and A-U, and Wobble base pairing G-U. Section 2.3 will talk about the secondary structure in more details. Figure 2.1.b shows in part (ii) the secondary structure.
- Tertiary structure: It is the three-dimensional structure of molecule which is relevant to the hydrogen bond occurrences between bases. Its elements involve an interaction between the distinct elements of secondary structure. Pseudoknots are example of tertiary structure. The tertiary structure shows in part (iii) Figure 2.1.b.

This thesis will treat the secondary structures of RNA.

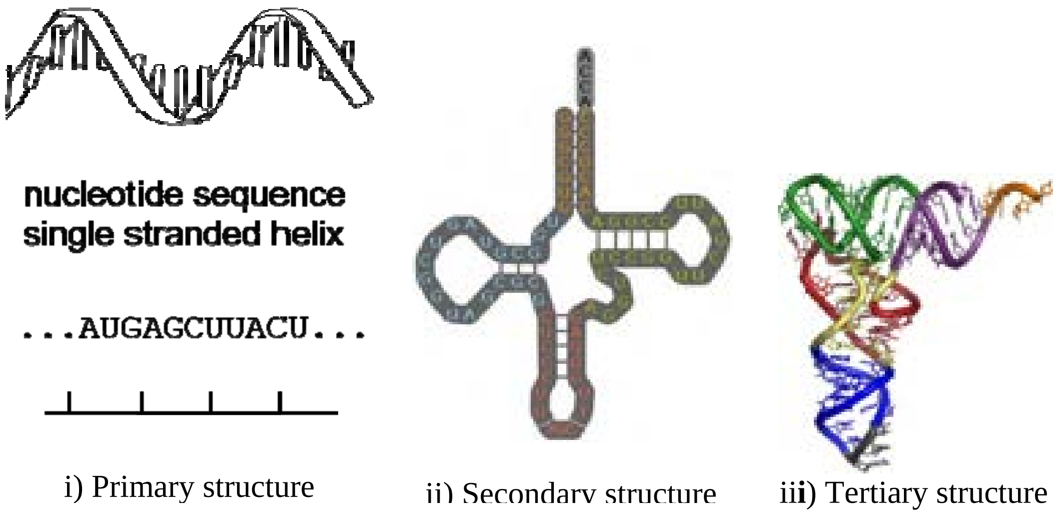


Figure 2.1.b: The RNA structure is represented at different levels. Image Source: Mathias Möhl, Sebastian Will, Rolf Backofen. “Fixed Parameter Tractable Alignment of RNA Structures Including Arbitrary Pseudoknots”. Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008):69-81.

2.2 Sequence Alignment

Sequence Alignment is one of the important terms for sequence comparison that compares the similarities among sequences. Sequence similarity often indicates functional and structural similarity.

Definition 2.2: (Alignment) Let two sequences S_1 and S_2 be over an alphabet $\sum$ with $- \notin \sum$, such that S_1, S_2 $\in \sum^*$. An *Alignment* $\mathcal{A}$ is a pair (S_1°, S_2°) with S_1°, $S_2^\circ \in (\sum \cup \{-\})^*$ such that:

1. The length of the aligned sequences are equal, i.e. $|S_1^\circ| = |S_2^\circ|$

2. There is no position *i* such that: $S_{1_i}^\circ = - = S_{2_i}^\circ$

3. The sequences S_1° and S_2° give the same sequences S_1 and S_2 respectively, if all gaps are removed.

Where the gaps are located in a sequence alignment for a base in one sequence there is no analogous base in the other sequence.

Note: for the RNA sequence alignment, an alphabet over $\sum = \{A, C, G, U\}$.

There are several types of sequence alignment and the Global alignment is one of these types. Global alignment represents the best alignment over the entire length of the two sequences and is suitable when both have similar length with enough degree of similarity throughout. A general global alignment technique is called the Needleman-Wunsch algorithm [NW70] which is based on dynamic programming that we will discuss it later in this chapter. Here, we refer to the pairwise sequence alignment which finds the best matching alignment between two sequences.

Example: Let S_1 = AGACUAGACAU and S_2 = CGAGACGU over $\sum = \{A, C, G, U\}$, the possible global pairwise sequence alignment which satisfies the above conditions, is:

S_1° = AGACUAGACAU

S_2° = CGA – – – GACGU

In addition to this sequence alignment, there is also another form that is comparison-dependent on the shape of the structures, this is called *Structural alignment*.

2.2.1 Edit Distance

Edit Distance is also one of definitions for comparing sequences that is a metric for measuring the amount of differences between two sequences. The edit distance between two sequences is given by the minimum number of edit operations that require transforming one sequence into the other. It is defined according to Clote et al. [CB2000] as follows:

Definition 2.2.1.a: (Edit Distance) Given a cost function $\mathcal{w}$: $(\sum \cup\{-\}) \times (\sum \cup\{-\}) \rightarrow \mathbb{R}$ and two sequences S_1 and S_2 over a finite alphabet $\sum$ where S_1, $S_2 \in \sum$*, the cost of E = e_1, e_2, …, e_r of edit operations is defined as $\mathcal{w}\ (E) = \sum_{i=1}^{r} \mathcal{w}\ (e_i)$. Then the *edit distance* of S_1, S_2 is defined as:

$$d_{\mathcal{w}}(S_1, S_2) = \min \{\mathcal{w}\ (E) \mid S_1 \Rightarrow_E S_2\}$$

There is another type of cost that gives mismatch cost x and gap cost y that are defined according to Sankoff [S85] as follows:

Definition 2.2.1.b: (Another representation for the cost of Edit distance) Let S_1 = $s_{1_1} \ldots s_{1_m}$ and S_2 = $s_{2_1} \ldots s_{2_n}$ be two sequences. An alignment of S_1 and S_2 is defined by two integer sequences $1 \leq i_1 < i_2 < \cdots < i_r \leq m$ and $1 \leq j_1 < j_2 < \cdots < j_r \leq n$. Let $\mathcal{s}$ be the number of pairs (i_k, j_k), such that $s_{1_{i_k}} \neq s_{2_{j_k}}$. The *cost* of the alignment is:

$$(m + n - 2r)y + \mathcal{s}x$$

The case of $\mathcal{s}$ replaces each $s_{1_{i_k}}$ by $s_{2_{j_k}}$, if they are not equal, and r indicate the number of all pairs in the alignment sequences.

The definitions 2.2.1.a and 2.2.1.b are equivalent, such that:

$$w\left(s_{1_i}, s_{2_j}\right) = \begin{cases} x & if\ s_{1_i} \neq s_{2_j} \\ y & if\ s_{1_i} = -\ or\ s_{2_j} = - \\ 0 & otherwise \end{cases}$$

2.2.2 Needleman-Wunsch Edit Distance Algorithm

In global sequence alignment, an attempt to align the entirety of two different sequences is made up to and includes the ends of sequence as mentioned before. The solving method for an edit distance problem is Needleman-Wunsch algorithm (1970) which was the first application of dynamic programming algorithm to biological sequence comparison [NW70]. The idea is to use dynamic programming to efficiently implement a recursion.

Definition 2.2.2: (Needleman-Wunsch Edit Distance algorithm) Given a metric cost function w, and two input sequences S_1 and S_2 over an alphabet $\sum$. The *Needleman-Wunsch algorithm* defines the matrix $D[i,j]$ with $0 \leq i \leq |S_1|$ and $0 \leq j \leq |S_2|$ by the recursion formula to obtain an optimal alignment as described:

$$\forall\, i,j > 0 : D[i,j] = min \begin{cases} D[i-1, j-1] + w(S_1[i], S_2[j]), \\ D[i-1, j] + w(S_1[i], -), \\ D[i, j-1] + w(-, S_2[j]). \end{cases}$$

With Initialization

$D[0,0] = 0,$

$D[i,0] = \sum_{k=1}^{i} w\ (S_1[k], -),$

$D[0,j] = \sum_{k=1}^{j} w\ (-, S_2[k]).$

The filled matrix is from top left $D[1,1]$ to bottom right $D[|S_1|, |S_2|]$. Suppose we have filled in the three entries $D[i-1, j], D[i-1, j-1]$ and $D[i, j-1]$ to the up and diagonally above and left of $D[i, j]$ respectively, that we have an optimal alignment for each of those three pairs, and then minimizing overall these pairs. We can either align $S_1[i]$ with $S_2[j]$, or align $S_1[i]$ with a new gap, or align $S_2[j]$ with a new gap.

After filling the matrix, the corresponding alignment is obtained from a trace back step through the filled matrix.

2.2.3 Smith-Waterman algorithm

This algorithm is a dynamic programming algorithm for the local sequence alignment [SW81]. Local sequence alignment is suitable for comparing with the sequences have short similar subsequences over two different lengths of sequences. The Smith-Waterman algorithm guarantees that finding the optimal local alignment accordance to the scoring system (substitution matrix and gap penalty) being applied, where the substitution matrix is a similarity between each pair of bases.

Definition 2.2.3: (Smith-Waterman algorithm) Given two input sequences $S_1 = s_{1_1} \dots s_{1_n}$ and $S_2 = s_{2_1} \dots s_{2_m}$ over an alphabet $\sum$ and the scoring function w between the sequence alignments. The *Smith-Waterman algorithm* defines the matrix $H[i, j]$ with $0 \le i \le n$ and $0 \le j \le m$, by the following recursion equations to produce the maximum similarity score:

$$H(i,j) = \max \begin{Bmatrix} 0 \\ H(i-1, j-1) + w\left(s_{1_i}, s_{2_j}\right) \\ H(i-1, j) + w\left(s_{1_i}, -\right) \\ H(i, j-1) + w\left(-, s_{2_j}\right) \end{Bmatrix} \quad for\ 1 \le i \le n, 1 \le j \le m$$

With initialization, $H[i, 0] = H[0, j] = 0$ for $0 \le i \le n, 0 \le j \le m$

This algorithm differs from Needleman-Wunsch algorithm by including a zero value for the negative similarity (i.e. out the range of subsequences).

2.2.4 Multiple Sequence Alignment

Multiple sequence alignment is an extension of pairwise alignment to incorporate more than two sequences at a time. In general, the input set of query sequences are assumed to have an evolutionary relationship by which they share a lineage and are descended from a common ancestor.

The most widely used approach to multiple sequence alignments uses a heuristic search known as progressive technique (also known as the hierarchical or tree method), that builds up a final Multiple Sequence Alignment by combining pairwise alignments beginning with the most similar pair and progressing to the most distantly related.

All progressive alignment methods require two stages: a first stage in which the relationships between the sequences are represented as a tree, called a *guide tree*, and a second stage in which the Multiple Sequence Alignment is built by adding the sequences sequentially to the growing Multiple Sequence Alignment according to the guide tree. The most important heuristic is to align the most similar pairs of sequences first. Typically guide trees are used to efficiently model this principle in progressive alignment algorithm. The most popular example for this alignment is: ClustalW [THG94].

2.3 RNA Secondary Structure

As mentioned before that RNA is usually a single-stranded linear molecule, but this is not the case in a biological system. RNA strand folds back on to itself via the base pair interactions to form secondary and tertiary structures which are essential for correct biological function. These functions include: (mRNA) genetic information copied from DNA to be used as a template for the synthesis of protein, (tRNA) serves as an adaptor which decodes the genetic code and (rRNA) catalyzes the protein synthesis. Therefore, the importance of the RNA secondary structures is found in many biological processes.

Furthermore, the efficiency in the structure prediction can provide the essential directions for experimental investigations.

The folding for an RNA molecule depends on the sequence nucleotides, by the complementary base pairing on it. A formal definition for an RNA secondary structure is given according to [MT78] as follows:

Definition 2.3: (RNA Secondary Structure) Let $S \in \{A, C, G, U\}^*$ be an RNA sequence. An *RNA secondary structure* over the sequence S is defined as a set of the base pairs P as follows:

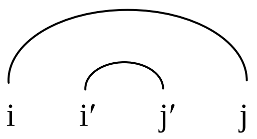
$P = \{(i, j) \mid i < j \wedge S_i$ and S_j form a complementary pair (Watson-Crick) or a non-standard pair (Wobble basepair)}

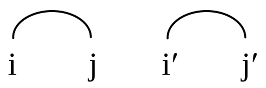
Where $\forall\, (i, j) \in P$ and $\forall\, (i', j') \in P$, P must satisfy the following condition:

- $i = i' \Leftrightarrow j = j'$, each base can have at most one bond with one other base.

A structure P is called nested structure, if it satisfies the following condition:

- $i < i'$ implies $(j' < j) \vee (j < i')$ must be satisfied.





Otherwise P is called crossing.

RNA secondary structure consists of contiguous basepairs which are called helices, and different kinds of loops that are the unpaired bases surrounded by helices. Hence, the secondary structure can be divided into various structural elements.

A base pair $(i', j') \in P$ is called *Accessible* from $(i, j) \in P$, if $i < i' < j' < j$ and if there is no other base pair $(i'', j'') \in P$ such that, $i < i'' < i' < j' < j'' < j$.

- A *hairpin loop* is formed when RNA strand folds back on itself. It is defined as follows: a base pair $(i, j) \in P$ closes a hairpin loop if $\forall\, i < i' \leq j' < j$: $(i', j') \notin P$.

- A *stack loop* is formed in the case two adjacent base pairs. It is defined as follows: a base pair (i, j) $\in P$ closes a stacking if (i +1, j -1)$\in P$. Several numbers of stacking basepairs are called *stem*.

- An *internal loop* is at least one unpaired base on each strand of the loop separating two paired regions. It is defined as follows: two base pairs (i , j) $\in P$ and (i′, j′) $\in P$ form an internal loop (i , j , i′ , j′) if they satisfy the following conditions:

- $i < i' < j' < j$
- $(i' - i) + (j - j') > 2$ (no stack)
- There is no base pair (i″, j″) between (i, j) and (i′, j′).

- A *bulge* has unpaired base on only one strand of the loop. It is called left or right bulge if $j = j' + 1$ or $i' = i + 1$ respectively. The other strand has uninterrupted base pairing.

All these elements of the secondary structure represent the *non-branching structures*.

- A *multi-branched loop* is the double-stranded regions which are coming together with separations of any number of unpaired bases, sometimes called *bifurcated structures*.

- An *external loop* is a number of single-stranded bases (unpaired bases) and basepairs which are not accessible from any basepair. There is no contribution to the total free energy in these regions.

Note that, the closing pair (i, j) of the k-loop is not itself defined to form part of that loop in the decomposition of the structure. Figure 2.3.a. shows the RNA secondary structure elements.

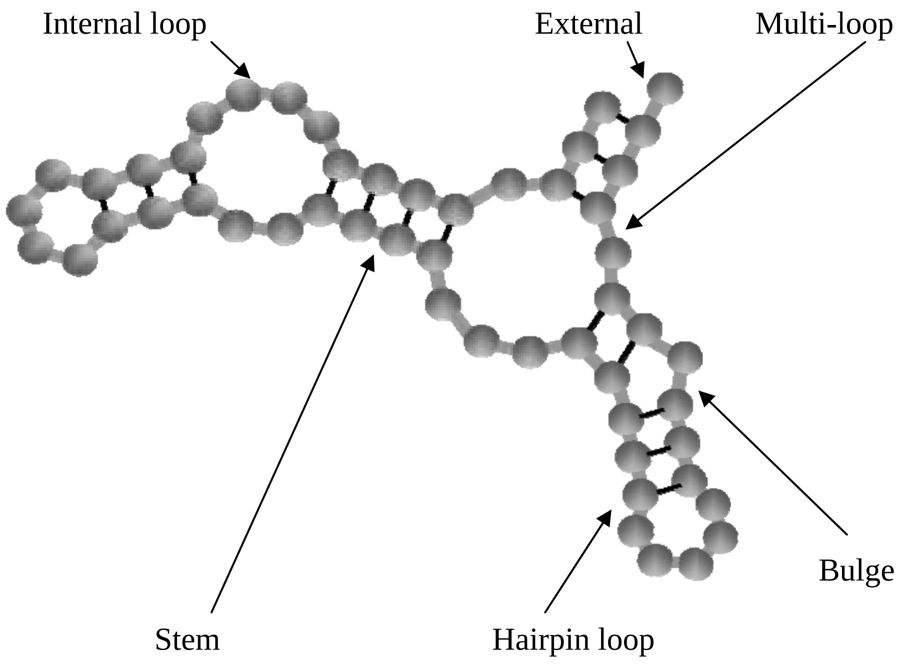


Figure 2.3.a: RNA secondary structures elements. Image Source: Internal loops in RNA secondary structure prediction, Lyngsø, Zuker, and Pedersen (1999).

There are different approaches for RNA secondary structure representations, such as RNA secondary structure graph and in bracket notation. They are showed in Figure 2.3.b.

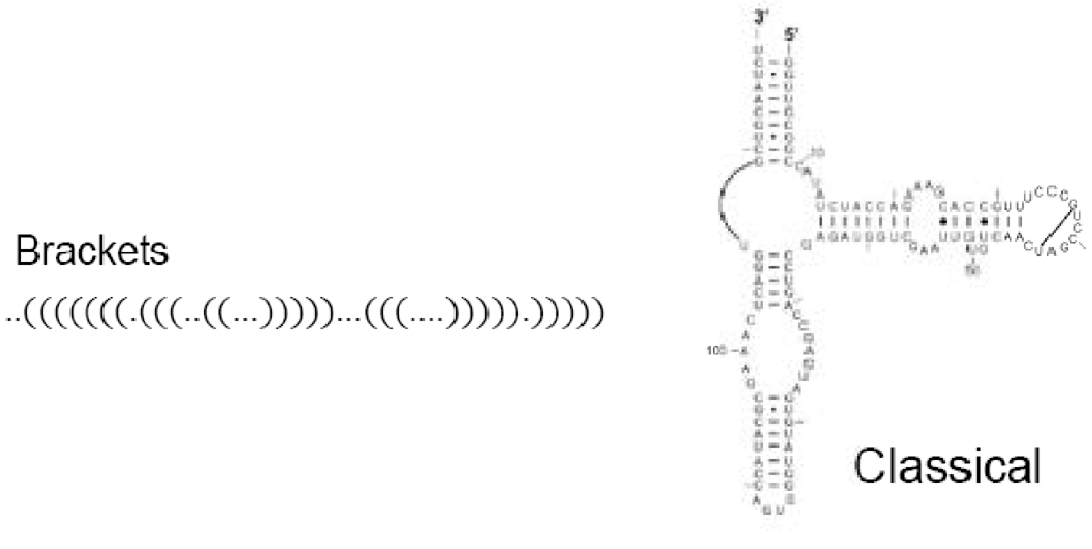


Figure 2.3.b: RNA secondary structure representations. Image Source: Jérôme Waldispühl. “18.417: Introduction to Computational Structural Biology”, Department of Mathematics, MIT

2.4 RNA Secondary Structure Prediction

There are two main approaches used to predict the secondary structure:

1. Comparative Sequence Analysis: It is the gold standard which determines an RNA secondary structure when a crystal structure is absent. It uses multiple sequence alignments of homologues sequences to predict the structure. Hence, it needs aligning many sequences with identical function. [PTW99]

2. Dynamic Programming (DP): This algorithm is used to solve the optimization problems by dividing the problem into independent sub problems. Each sub problem is solved only once, and its solution is stored in a table such that re-computing the solution is avoided. It contains several approaches that are used to predict an RNA secondary structure:

Nussinov algorithm [N80] represents one of the first attempts of RNA secondary structure prediction. It determines the maximum number of basepairs in non crossing structure. It can be defined as follows:

Definition 2.4.a: (Nussinov matrix) Let S be an RNA sequence. The *Nussinov matrix* $N_{i,j}$ is defined as follow:

$N_{i,j}$ = max $\{|P| \mid P$ is a nested structure of the subsequence $s_i \ldots s_j\}$

where $1 \le i \le |S|$ and $i-1 \le j \le |S| \land j > 1$, this implies to the following recursion equations:

Initialization: $N_{i-1,i} = 0$, $N_{i,i} = 0 \quad \forall \; 1 \le i \le |S|$,

Recursion: $\forall$ i $\le$ j

$$N_{i,j} = max \begin{cases} N_{i,j-1}, \\ max_{i \le k < j} N_{i,k-1} + 1 + N_{k+1,j-1}, \\ Where\ s_k\ and\ s_j \text{ complementary} \end{cases}$$

Now after filling this matrix, we will get the best structure by using a trace back procedure. The *Nussinov algorithm* detects mostly just one variant from various possibilities for base pairing, it also does not consider the size of internal loops, the stacking of base pairs and the strength of base pair.

The second approach which is achieved by using DP is Zuker algorithm [ZS81] that we treated in this thesis. It computes the Minimum Free Energy secondary structure and it distinguishes among all possible basepair loops.

Before discussing this algorithm, we need to identify the definition of Free Energy.

Definition 2.4.b: (Free Energy) The Gibbsian *Free Energy* G in a system (e.g. of gas molecules in equilibrium or in a dilution of molecules) holds:

$$G = U - T\,S$$

where U is the enthalpy, T is the absolute temperature (in Kelvin) and S is the entropy.

Two terms determine the *free energy* of molecule:

- Enthalpy: from secondary structure basepairs.
- Entropy: "disorder in unpaired regions".

Usually the difference $\Delta G = \Delta H - T\,\Delta S$, can be experimentally measured ⇒ flexible rules.

where ΔG is approximated as the sum of contributions from loops of base pairs and other secondary structures.

Definition 2.4.c: (Energy contribution for loops) the energy contribution of the secondary structure elements are defined as follows:

- Hairpin loop (i, j),
- Stacking (i, j, i +1, j - 1),
- Internal loop (i, j, i′, j′).

  Assume that $E(\ell)$ describes the energy contribution for each of the above three structure element loops, and the simplified energy contribution for the multi-loops is:

- Multi-loop: $E(\ell) = a + bk + ck'$,

where a, b, c are weights, a is the energy contribution of closing basepair, b and c are constants for k = number of helices and k′ = number of single-stranded positions, respectively.

The complete free energy is measured, by taking the summation of energy loops:

$E(P) = \sum_{(i,j)\in P} E(P_{ij})$, where $E(P_{ij})$ is the energy contribution of the secondary structure element P which is closed by basepair(i, j). The energy of such structure P_x of sequence x, is called "Turner Free Energy", and the total free energy is called "Turner Free Energy Model".

Now for the Zuker algorithm that the widely used computational approach for predicting RNA secondary structures from single sequences, which is based on thermodynamic models that associates the free energy values from each possible secondary structure of a strand. The secondary structure is with the lowest possible free energy value, the minimum free energy (MFE) structure is predicted to be the most stable secondary structure of the strand.

Now, the Zuker matrices are defined according to Sankoff [S85] as follows:

Definition 2.4.d: (Matrix $F(i, j)$, Matrix $C(i, j)$) Let $F(i, j)$ be the minimum energy possible for a secondary structure P on the partial sequence $i, \ldots, j$. Let $C(i, j)$ be the minimum energy given that $(i, j) \in P$, where $C(i, j) = \infty$ if no such structure exists.

$$C(i,j) = min \begin{cases} E(\ell)\,, & \ell \text{ is the hairpin closed by } (i, j), \\ min\{E(\ell) + C(p,q)\}\,, & \ell \text{ a 2-loop closed by } (i, j) \text{ with} \\ & (p, q) \text{ accessible, } u = p - i + j - q - 2 \le U, \\ \underset{i<h<j-1}{min}\{G(i+1,h) + G(h+1,j-1) + a\}\,, & \end{cases}$$

where $E(\ell)$ is represent Turner Free Energy. G is a matrix for multiple loops defined as follows:

$$G(i,j) = min\begin{cases} C(i,j) + c\ , \\ \\ \underset{i \le h < j}{min}\ min\begin{cases} G(i,h) + (j-h)b\ , \\ G(i,h) + G(h+1,j)\ , \\ (h-i+1)b + G(h+1,j)\ , \end{cases} \end{cases}$$

$$F(i,j) = min\begin{cases} C(i,j)\ , \\ \\ \min\limits_{i \le h < j}\{F(i,h) + F(h+1,j)\}\ , \end{cases}$$

As usual, by applying trace back step to the filled matrices, we can get the minimum free energy secondary structure.

Chapter Three

Methods for Simultaneous Alignment and Folding

This chapter will introduce the methods that are used to solve the problem of simultaneous Folding and Alignment for RNA sequences. We will start with the original work “Sankoff Algorithm” and then present the variants to this algorithm, which are restricted implementations to reduce the computational complexity.

3.1 Sankoff Algorithm

The last chapter discusses “Sequence Alignment” by explaining how to calculate the optimal alignment distance between two different sequences; it also talks about “Folding of RNA sequence” and through such approach which achieved to predict the RNA secondary structure by Dynamic Programming, the Minimum Free Energy secondary structure can be computed. As we see, all these problems have optimal dynamic programming solutions.

Sankoff algorithm [S85] solves these problems which have dynamic programming solutions simultaneously for two sequences with length N and at time proportional to $\mathcal{O}(N^6)$ and storage $\mathcal{O}(N^4)$. The following steps are included to describe Sankoff algorithm according to Sankoff [S85]:

First, equivalent structures must be defined for two RNA sequences; the branching configuration represents an invariant part of the identity structure. It is determined by two structure elements, which are the external pairs and multiple loops.

Definition 3.1.a: (Equivalent Structures) Let $i_1 < i_2 < \ldots < i_n$ and $j_1 < j_2 < \ldots < j_m$ be positions in the sequences S_1 and S_2 respectively, of all elements that are either an external pair or an accessible pair in a multiple loop of the nested structures P_1 of sequence S_1 and P_2 of sequence S_2. The *Equivalent Structures* for P_1 and P_2 according to the branching configurations require that n = m and $(i_f, i_g) \in P_1$ if and only if $(j_f, j_g) \in P_2$.

According to the definition there are no restrictions on the number and type of 2-loops that are nested in each external pair and in each multiple loop accessible pair, and also on the number of unpaired bases in any k-loop and unpaired external bases. Therefore, the equivalence between the structures represents an essential part for finding two sequences which have a common folding, but it is still not sufficient, as shown in Figure 3.1.a.

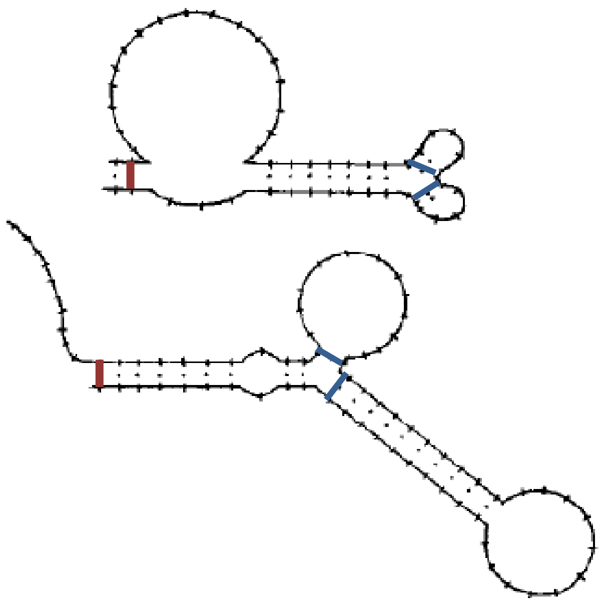


Figure 3.1.a: This figure shows two equivalents, but they have highly dissimilar secondary structures. [S85]

Second, the same secondary structures of two sequences are needed to provide the high similarities and not just the equivalent branching configurations. To assess the similarities between two equivalent structures, the idea of Alignment will be introduced. The equivalence between the structures of two sequences is guaranteed, through the constrained alignments between these structures.

Definition 3.1.b: (Constrained Alignment) Let $i_1 < i_2 < \ldots < i_n$ and $j_1 < j_2 < \ldots < j_m$ be positions in the sequences S_1 and S_2 respectively. The *Constrained Alignment* on the structures P_1 and P_2 of these sequences respectively is: i_1 aligned with j_1 and i_2 with j_2 … i_n with j_m.

Thus, any K-loop in one structure is aligned with a single K-loop of the other structure, or may be deleted or inserted in some cases. Now, one can see the following cases that describe the constrained alignment between two structures according to Sankoff [S85] as follows:

- External pairs and accessible pairs in multiple loops: all external pairs and accessible pairs in multiple loops are aligned and not inserted or deleted; such that each of them will have its correspondence in the other structure.
- 2-Loops: it has no constraint against the insertion or the deletion to their accessible pairs; hence it is free to be different from one structure and the other.
- Hairpins: a hairpin in the structure is aligned with its correspondence in the other structure, such that the equivalent structures will have the same number of hairpins and at the same locations on the structures.

Third, we can now clearly determine our target of finding the “equivalent structures” and “constrained alignment” in such a way that makes the whole configuration of structure and alignment, optimal. However, the expectation to find the equivalent structures which are thermodynamically optimal in each sequence separately is difficult to get. Even if it is found like this case, an appropriate constrained alignment between them might not inevitably be the minimum cost among all possible pairs of the equivalent structures.

Definition 3.1.c$_1$: (Sankoff-score) Let $\mathcal{A}$ be an alignment of the sequences S_1 and S_2, *and* let P_1 be a nested structure of sequence S_1 and let P_2 be a nested structure of sequence S_2. The *Sankoff-score* of $\mathcal{A}$, P_1 and P_2 is given as:

Sankoff-score $(\mathcal{A}, P_1, P_2)$ = Edit-Distance $(\mathcal{A})$ + E_1 (P_1) + E_2 (P_2)

Definition 3.1.c$_2$: (Sankoff problem) Given two sequences S_1 and S_2 as input. The *Sankoff problem* is the problem to find the lowest free energy secondary structure common to a nested structure P_1 of sequence S_1 and a nested structure P_2 of sequence

S_2, *and* an alignment $\mathcal{A}$ of the sequences S_1 and S_2, such that the two structures P_1 and P_2 are equivalent and the alignment between these structures is constrained, and Sankoff-score $(\mathcal{A}, P_1, P_2)$ is minimized.

Therefore, to optimize this problem, we use a new objective function which represents a trade-off between the free energy and alignment cost for the two sequences. The following definitions are used to find the optimizing structure and alignments for two sequences, which are defined according to Sankoff [S85] as follows:

Definition 3.1.c$_3$: (Matrix $D(i_1, j_1; i_2, j_2)$) The extension of the definition $D(i, j)$ (which is defined in the previous chapter) is, the minimal Edit Distance cost of an alignment between partial sequences $s_{i1}, \ldots, s_{j1}$ and $s_{i2}, \ldots, s_{j2}$,
If $i_1 > j_1$, then the cost for inserting the entire sequence is $s_{i2}, \ldots, s_{j2}$
If $i_2 > j_2$, then the cost for deleting the entire sequence is $s_{i1}, \ldots, s_{j1}$

Definition 3.1.c$_4$: (Matrix $F(i_1, j_1; i_2, j_2)$, Matrix $C(i_1, j_1; i_2, j_2)$) Let $F(i_1, j_1; i_2, j_2)$ be the minimum cost possible for a pair of equivalent secondary structures P_1 and P_2 on positions $i_1, \ldots, j_1$ and $i_2, \ldots, j_2$ of sequences S_1 and S_2 respectively, where the cost is the sum of the free energy and the constrained alignment cost. Let C $(i_1, j_1; i_2, j_2)$ be the minimum cost given that $(i_1, j_1) \in P_1$ and $(i_2, j_2) \in P_2$ without considering the costs of aligning s_{i_1}, s_{j_1}, $s_{i_2}$and s_{j_2}, If no such pair of structures exists, set $C = \infty$. Then recursion C is:

$$C(i_1,j_1;i_2,j_2)$$

$$= min \begin{cases} E(\ell_1) + E(\ell_2) + D(i_1+1, j_1-1; i_2+1, j_2-1)\ , \quad \ell_1, \ell_2 \textit{ hairpins closed by } (i_1,j_1), (i_2,j_2) \textit{ respectively,} \\ \\ min\,\{E(\ell_1) + E(\ell_2) + C(p_1,q_1;p_2,q_2) + D(i_1+1,p_1;i_2+1,p_2) \\ \qquad + D(q_1,j_1-1;q_2,j_2-1)\}\ , \\ \qquad \ell_1, \ell_2 \textit{ are 2-loops closed by } (i_1,j_1), \\ \qquad (i_2,j_2) \textit{ with } (p_1,q_1), (p_2,q_2) \textit{ accessible,} \\ \qquad p_1 - i_1 + j_1 - q_1 - 2 \le U, p_2 - i_2 + j_2 - q_2 - 2 \le U, \\ \qquad \textit{or one of } \begin{cases} \ell_1 = \emptyset \quad \textit{and } (p_1,q_1) = (i_1,j_1) \\ \ell_2 = \emptyset \quad \textit{and } (p_2,q_2) = (i_2,j_2), \end{cases} \\ \\ \min\limits_{\substack{i_1<h_1<j_1-1 \\ i_2<h_2<j_2-1}} \{G(i_1+1,h_1;i_2+1,h_2) + G(h_1+1,j_1-1;h_2+1,j_2-1) + 2a\} \end{cases}$$

The first and second options in this matrix refer to the fact that all terms in hairpin or in 2-loop are aligned with their corresponding or, in the case of 2-loop, that the entire loop is deleted or inserted. The third option is, *G*, be matrix used for multiple loops.

Recursion for *G* and *F*:

$$G(i_1,j_1;i_2,j_2) = min \begin{cases} C(i_1,j_1;i_2,j_2) + 2c + D(i_1,i_1;i_2,i_2) + D(j_1,j_1;j_2,j_2)\ , \\ \\ \underset{\substack{i_1<h_1<j_1 \\ i_2<h_2<j_2}}{min}\ min \begin{cases} G(i_1,h_1;i_2,h_2) + (j_1-h_1+j_2-h_2)b \\ \qquad + D(h_1+1,j_1;h_2+1,j_2)\ , \\ G(i_1,h_1;i_2,h_2) + G(h_1+1,j_1;h_2+1,j_2)\ , \\ (h_1-i_1+1+h_2-i_2+1)b \\ \qquad + G(h_1+1,j_1;h_2+1,j_2) \\ \qquad + D(i_1,h_1;i_2,h_2)\ , \end{cases} \end{cases}$$

$$F(i_1,j_1;i_2,j_2) = min \begin{cases} C(i_1,j_1;i_2,j_2) + D(i_1,i_1;i_2,i_2) + D(j_1,j_1;j_2,j_2)\ , \\ \\ \underset{\substack{i_1\le h_1<j_1 \\ i_2\le h_2<j_2}}{min} \{F(i_1,h_1;i_2,h_2) + F(h_1+1,j_1;h_2+1,j_2)\}\ , \\ \\ D(i_1,j_1;i_2,j_2)\ , \end{cases}$$

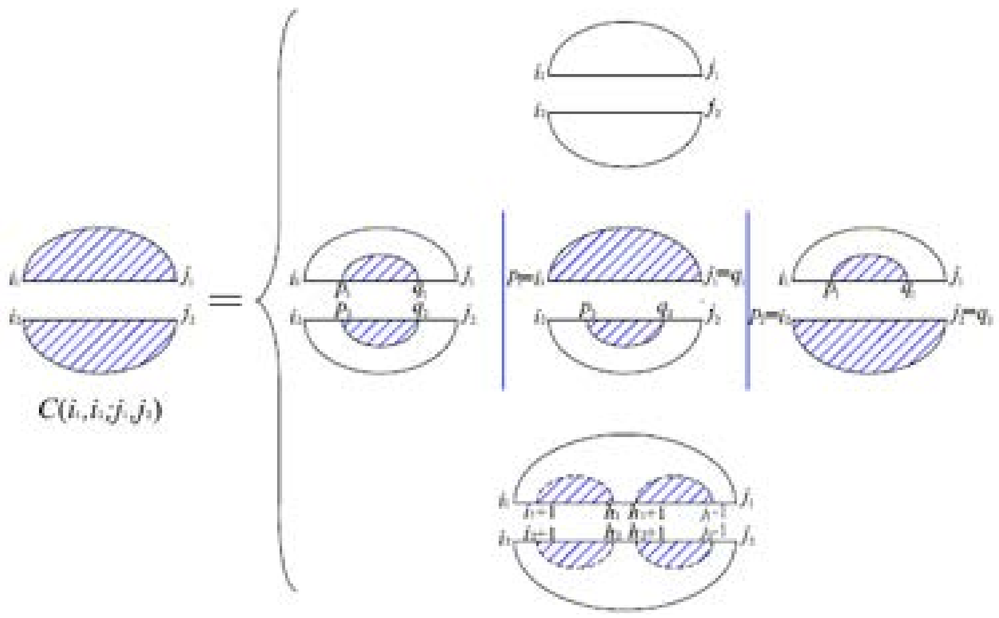
The initial conditions are *C* (i_1, i_1; i_2, i_2) = ∞ and *G* (i_1, i_1; i_2, j_2) = *G* (i_1, j_1; i_2, i_2) = ∞.

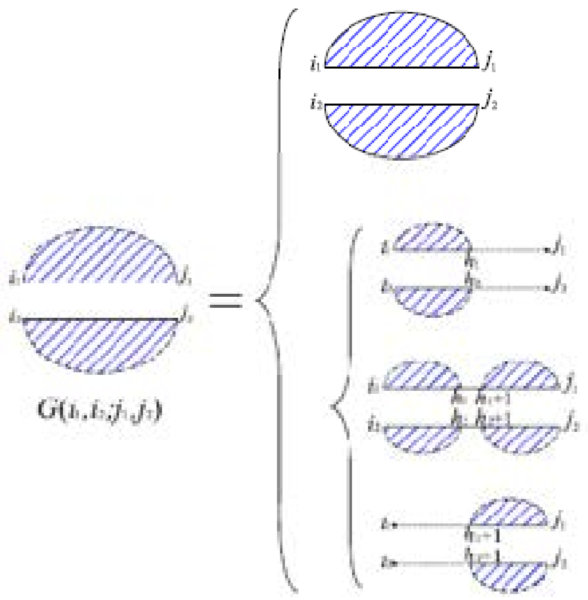
The first case of matrix *G* refers to the fact that corresponding accessible pairs are aligned in multiple loops, and the same for external pairs in the first option of matrix *F*. The second and fourth options in *G* refer to corresponding multiple loops aligned and the same for the last option of *F* for the corresponding external regions. The last and first options in *F* indicate the two cases, zero external pairs and one such pair, which are found in both structures respectively.

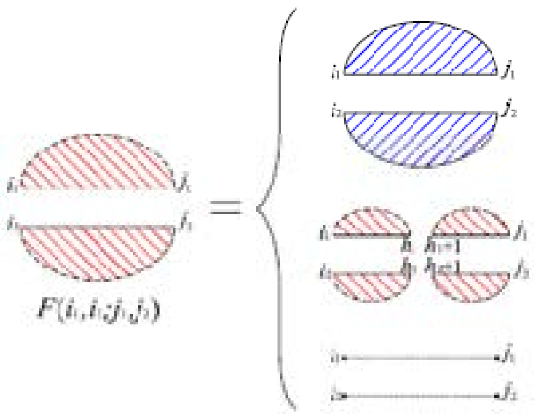
As we showed above, that these matrices include all configurations in a structure such that the corresponding accessible pairs for the multiple loops and the accessible terms for the external regions are aligned; also the corresponding closing pair of hairpins is aligned, but for the loops of index 2 they are either aligned, completely inserted or deleted.

Before illustrating the correctness of these recurrences which identify the optimal structures, we just want to indicate that the recursions of Zuker algorithm (that are defined in the previous chapter) are an important part to construct the above recursions.

Now, the following Figures illustrate the correctness of the Sankoff algorithm recursions:







Finally, it can be seen that Sankoff algorithm is considered an ideal approach, but it has high computational complexity. For this reason, several methods have emerged which implement Sankoff algorithm but with pragmatic restrictions to make it practical to use and these methods are presented in the following parts of this chapter.

3.2 Dynalign

The previous part shows the simultaneous RNA sequences Alignment and Folding by presenting the Sankoff Algorithm which represents the original form to solve these problems simultaneously. Since, it is computationally over expensive (as we have seen), so there are several methods which implement Sankoff Algorithm under various restrictions to make it more practical to use. Dynalign indicates one of these methods that suggested by Mathews et al. [MT02].

The main idea of Dynalign is to find the secondary structure common to two sequences. This method depends on the dynamic programming algorithm suggested by Sankoff, which finds lowest free energy secondary structure common for two RNA sequences and the sequence alignment that supports this structure. The restriction used in this method, is parameter M which restricts the maximum distance between the positions of aligned nucleotides of two sequences. Therefore, the computational complexity will be more tractable with this restriction, $\mathcal{O}(M^3N^3)$ for the time and $\mathcal{O}(M^2N^2)$ for the memory, where M is the maximum separation parameter which restricts the set of sequence alignments which are considered and N is the length of the shorter sequence [MT02].

The general description of Dynalign is given according to Mathews et al. [MT02] as:

"Dynalign is a computer algorithm that improves the accuracy of structure prediction by combining free energy minimization and comparative sequence analysis to find a low free energy structure common to two sequences without requiring any sequence identity".

When they write about "comparative sequence analysis", they refer to finding a structure common to two or more sequences. This is the method, which is not yet automated, by which most RNA secondary structures are solved. Thus, the comparative sequence analysis here shows the comparison between two structures during the alignment.

Definition 3.2: (Dynalign problem) Let two sequences S_1 and S_2 be given as input. The Dynalign problem finds the lowest free energy secondary structure common to S_1 and S_2, *and* the sequence alignment between S_1 and S_2 that supports this structure, such that the basepairs of S_1 and S_2 are preserved in the same aligned positions in the alignment, and it minimizes $\Delta G°_{total}$ which is the total free energy of the system, where:

$$\Delta G^{\circ}_{total} = \Delta G^{\circ}s_1 + \Delta G^{\circ}s_2 + k\ \Delta G^{\circ}_{gap}$$

Where $\Delta G^{\circ}s_1$ and $\Delta G^{\circ}s_2$ are forms of “Turner Free Energy Model” (i.e. the conformational free energies) for the sequences S_1 and S_2 respectively, which are computed by the nearest-neighbor approximation [MSZ+T99] (another typology of free energy that refers to a “special case” of the Sankoff). ΔG°_{gap} is the gap penalty that applies to each gap in the alignment and k is the number of gaps.

This method does not explicitly score the sequence identity, because as shown from the equation above ΔG°_{total} is not based on the matching nucleotides that occurred in the sequence alignment. Generally, this method can be used to predict the structures for homologous sequences that do not have the sequence identity but only the structure conservation.

One should see, the analogy between Dynalign method and Sankoff algorithm in principle: It is shown in Sankoff algorithm minimizes the total cost of combining the free energy minimization of the RNA secondary structure for two sequences (Folding), and the minimum alignment cost by the optimal distance between the structures (Alignment). Therefore, Sankoff algorithm depends on the matching nucleotides in the Alignment. The Dynalign method minimizes the total free energy as shown above, by combining the free energy minimization for two sequences (Folding), and the energy contribution of gap which is multiplied by the number of gaps (Alignment). So it does not depend on the matching nucleotides.

Dynalign is one of the practical implementations of the Sankoff algorithm, which is a dynamic programming algorithm solution for both of sequence alignment and RNA secondary structure prediction for two sequences. Therefore, it guarantees an optimal solution which is one important point in the dynamic programming algorithm for supporting optimal solution.

The restriction of this method is a parameter M which restricts the depth of a search for the alignments between two sequences. This restriction modifies the definition 3.2 and the restriction can be shown as follows:

For all $i \in S_1$, $k \in S_2$ and (i, k) is a pair in alignment: $|i - k| \leq M$.

The computational complexity will be tractable with this restriction, such that $\mathcal{O}(M^3N^3)$ for time and $\mathcal{O}(M^2N^2)$ for storage, where a parameter M decreases the set of alignments which are considered and N is the length of shorter sequence, compared with Sankoff algorithm, $\mathcal{O}(N^6)$ for time and $\mathcal{O}(N^4)$ for storage, where N is also the length of the smaller of the two sequences.

In the Dynalign implementation, another scheme for a parameter M is used to recast the parameter M implementation in such a way that scales with the difference in sequence length for the two sequences, where the nucleotide i from the first sequence and the nucleotide k from the second sequence:

$$|i \times (N_2/N_1) - k| \leq M$$

where N_1 is the length of the first sequence and N_2 is the length of the second sequence. This restriction allows aligning the ends of the sequences ($i = N_1$and $k = N_2$) at any M and any difference of the sequence length. This restriction can be chosen for significantly smaller M sizes than the shorter sequence length N, hence, it reduces the computation complexity. The Results chapter will show that this scheme for parameter M was used in the Dynalign program.

Dynalign does not depend on the scoring of base matches as we have showed; therefore, it has no problem for compensating base changes. Unlike to the Sankoff algorithm that includes a scoring function for the matching nucleotides in the alignment.

In practice, both of Sankoff algorithm and Dynalign method have limitation on the sequence length because of computational complexity. No prediction of pseudoknots occurs in each of Sankoff algorithm and Dynalign method.

Dynalign has three matrices like Sankoff, but with different names. In this method, the maximum distance restriction between aligned nucleotides by parameter M, computationally leads to simplify the structure calculation. These matrices that including multi-branch loops are defined according to Mathews et al. [MT02] as follows:

- **Definition 3.2.a**: (Matrix *V(i, j, k, l)*) Matrix *V(i, j, k, l)* is the minimal sum of the free energies for the two sequences that cover nucleotides *i, …, j* in the first sequence and *k, …, l* in the second sequence, such that *(i, j)*and *(k, l)* are basepairs and *i* aligned with *k* and *j* aligned with *l*, plus any gap penalties for interior nucleotides in the sequence alignment. It is recursively computed as minimum of the three cases:

- V_1 for hairpin loops which is closed by the basepairs *(i, j)* and *(k, l)*,
- V_2 is the lowest sum of the free energies for one of these loops: a helix extension, bulge loop, or internal loop in the common structure.
- V_3 is also the lowest sum of free energies for a multi-branch loop that closed by the basepairs *(i, j)* and *(k, l)*. It has 16 cases that are computed to all possible combinations of whether or not that *i+1* and *j-1* are dangling ends on the basepair *(i, j)*, and whether *k+1* or *l-1* are dangling ends on the basepair *(k, l)*.

This matrix corresponds to the *C*-matrix in Sankoff, but in the case of a multi-branch loop, in contrast to Sankoff, it includes all possible cases for determining the dangling ends on both the closing basepairs.

- **Definition 3.2.b**: (Matrix *W(i, j, k, l)*) Matrix *W(i, j, k, l)* is the minimal sum of the free energies for the nucleotides *i, …, j* of the first sequence and *k, …, l* of the second sequence and *i* aligned with *k* and *j* aligned with *l*, plus any gap penalties for interior nucleotides in the sequence alignment. It is the minimum of three cases:

- W_1 for adding unpaired nucleotides to a multi-branch loop, and similarly to V_3, it has also 16 possible cases.
- W_2 for helix termini.
- W_3 for bifurcation in the structure. It's necessary for considering multi-branch loops with more than three branching helices.

This matrix corresponds to the *G*-matrix in Sankoff, but in the case of adding unpaired nucleotides, in contrast to Sankoff, it includes all possible cases for determining the unpaired nucleotides of the dangling ends.

- **Definition 3.2.c**: (Matrix *W5(i, k)*) Matrix *W5(i, k)* is the minimal sum of free energies for the nucleotides *1, …,i* of the first sequence and *1, …, k* of the second sequence, plus any gap penalties for nucleotides in sequence alignment. It is the minimum of the four cases, which assuming that several consecutive helices in both structures are not closed by basepairs.

This matrix corresponds to the *F*-matrix in Sankoff, but in contrast to Sankoff, one of the cases also has 16 possible cases for allowing the dangling ends on the helices closed by such basepair.

In recent years, there have been some developments for this method that reducing the computational complexity for time and memory requirements, and in addition to improve the accuracy in the structure prediction.

In Dynalign method, the restriction is the parameter M, as discussed before that defines as a measure of maximum insertion length. This parameter controls the trade-off between the computation and accuracy. Whereas a small value of M is desired to decrease the computation time, the accuracy of the secondary structure prediction will also be decreased. Therefore, the determination value of M is essential for the structure prediction accuracy. A large value of M is desired for the longer sequences, since they need longer insertions, while more computation time will be required. Due to this limitation in the Dynalign method, for selecting the values of parameter M, a new methodology is suggested in Dynalign by Harmanci et al. [HSM07].

This new methodology imposes constraints on the alignment in Dynalign, and these constraints are defined by a probabilistic analysis. A posteriori probability that is used for the nucleotide alignments, estimates the confidence in local accuracy of the sequence alignment, and it is efficiently computed by Hidden Markov Model [DEK+M99]. These estimations restrict the choices of the dynamic programming step by the constraint windows. In high confidence regions, strong constraints are imposed on the possibilities in dynamic programming steps by cutting off the computation which is not required. Otherwise, the low confidence regions, allow many possibilities in the dynamic programming steps.

The used formulation of Hidden Markov Model (HMM) computes the *posteriori* symbol-to-symbol alignment probabilities for the homologous sequences which are represented by $\Pr(i \leftrightarrow k \mid S_1, S_2)$, i.e. the probability of co-incidence between one nucleotide position i of sequence S_1 with other nucleotide position k of sequence S_2 [DEK+M99].

There are three conditions satisfying the co-incidence between two nucleotide positions (one of each sequence) according to Harmanci et al. [HSM07], these conditions are as follows:

- Nucleotide positions i and k are aligned,
- Nucleotide position i occurs in an "insertion" in sequence S_1 and nucleotide position k in sequence S_2 aligns with nucleotide position i_ from sequence S_1, where i_ denotes the largest position index less than i in sequence S_1 that aligns with a nucleotide position from sequence S_2.
- Nucleotide position k occurs in an "insertion" in the sequence S_2 and nucleotide position i in sequence S_1 aligns with nucleotide position k_ from sequence S_2, where k denotes the largest position index less than k in sequence S_2 that aligns with a nucleotide position from sequence S_1.

The effective computation of a posteriori probability of the co-incidence between nucleotide positions is done by using the HMM forward-backward algorithm, which is described according to Harmanci et al. [HSM07] as follows:

$$Pr(i \leftrightarrow k \mid S_1, S_2) = \frac{\sum_m \alpha_m(i,k)\beta_m(i,k)}{Pr(S_1,S_2)}$$

Where the sum is over m = {ALN, INS1, INS2}, which represents the set of three possible states for the nucleotides co-incidence. Each state defines the alignment according to the nucleotide positions between two sequences. So the aligned nucleotide positions and an insertion of the sequence S_1 and an insertion of the sequence S_2, are representing the states ALN, INS1, INS2 respectively. The forward variable α_m(i, k) keeps track of events before alignment position (i, k) and the backward variable β_m(i, k) keeps track of events after alignment position (i, k). $Pr(S_1, S_2)$ is the probability of emission of the observed sequences.

A low value of the posterior co-incidence probability $Pr(i \leftrightarrow k \mid S_1, S_2)$ is that a nucleotide position i in sequence S_1 not probable to co-incident with nucleotide position k in sequence S_2. Therefore, the suggestion is to impose constraint on the alignments in Dynalign by excluding all alignments that are not probable through having very small value of the posterior co-incidence probability. So this occurs by defining an alignment constraint by comparing the posterior co-incidence probability with an appropriate low threshold $P_{threshold}$, and according to Harmanci et al. [HSM07] an alignment constraint set is defined as follows:

$$C = \{(i, k) \mid (Pr(i \leftrightarrow k \mid S_1, S_2) > P_{threshold}\}$$

Where *C* represents an alignment constraint set and its elements describe the pairs of nucleotide positions that may co-incident between the sequences. Otherwise, they are rejected.

The threshold value $P_{threshold}$ has analogous to a parameter M. It controls a trade-off between computation and accuracy. A low value of $P_{threshold}$, a strong confidence is determined and hence the constraint sets will include all actual alignments, but more computation is required for the choices that are increased. For a high value of $P_{threshold}$, the computation requirements are decreased because an alignment constraint set will be restricted. A higher $P_{threshold}$, may be very restrictive such that it prevents the optimal alignment to be included in the set of alignments that is considered due to the alignment constraint set. Hence, the prediction accuracy will be reduced in Dynalign.

The alignment constraints are effectively determined by providing an appropriate HMM parameter and threshold values. An appropriate threshold probability is chosen according to the several experiments performed on the sequences in the implementation, which is depended on the similarity of sequences.

This new methodology for Dynalign produces a significant improvement in accuracy and speed, compared to the previous heuristic of Dynalign.

3.3 Foldalign

This part will discuss another method which represents the first practical implementation of Sankoff algorithm for simultaneous folding and alignment of RNA sequences, this is called Foldalign method. This method has three versions, according to the heuristics and improvements that have been added.

3.3.1 Pairwise Foldalign 1.0

This method utilizes a simplified version of Sankoff Algorithm but neglects the branching structures; this version is called pairwise Foldalign 1.0 [GHS97a], [GHS97b]. Therefore, it lowers the calculation time from $\mathcal{O}(N^6)$ to $\mathcal{O}(N^4)$ for two sequences, where N is the sequence length. This method combines both of sequence similarity and structure, such that, it is based on the local sequence alignment by using the definition of (Smith-Waterman algorithm) for the aligning part, and optimizes the number of basepairs in the structures by using the definition of (Nussinov algorithm), rather than free energies for the folding part.

Now one must to show the analogies of the Sankoff algorithm between the Foldalign method which simplifies and extends the basic Sankoff algorithm. As is known the Sankoff algorithm minimizes the total cost of combination of both the minimum Alignment cost and the minimum free energy structures for two sequences; the Foldalign method is focused on the local sequence alignment (that is mentioned before, in the preliminaries chapter), therefore, it applies the definition of (Smith-Waterman algorithm) which finds the maximum score local alignment for two sequences. So, it maximizes a score that combines sequence similarity and structure. For this case, it employs the Nussinov algorithm that maximizes the number of basepairs to score the structure. Now, this gives the ability to exploit pairwise Foldalign 1.0 for determining the maximum scoring local alignment RNA sequences.

Now, the following definitions of the pairwise Foldalign 1.0 according to Gorodkin et al. [GHS97a], [GHS97b] are presented as follows:

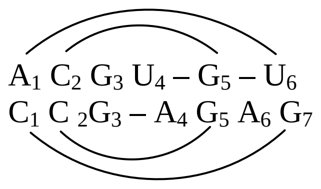
Definition 3.3.1.a: (FA1.0 scoring matrix) The FA1.0 scoring matrix ($S_{ij,kl}$) is 25 × 25 matrix for the four bases that also include the gaps, where the indices i, j, k, l ∈ {A, C, G, U, −} and its values in ℝ. It combines the two independent contributions that will be discussed later in the FA1.0 scoring matrix construction.

Definition 3.3.1.b: (FA1.0-score) Let $\mathcal{A}$ be an alignment of the sequences A and B, let P_A be a nested structure of sequence A and let P_B be a nested structure of sequence B. We assume a fixed FA1.0 scoring matrix S. The *FA1.0-score* of $\mathcal{A}$, P_A and P_B is given as:

$$\text{FA1.0} - \text{score}(\mathcal{A}, P_A, P_B) = \sum_{\substack{(i,j)\in P_A \\ (k,l)\in P_B \\ and\ (i,k)\in\mathcal{A},(j,l)\in\mathcal{A}}} \zeta_{a_i b_k, a_j b_l} + \sum_{p\in[1\ldots|\mathcal{A}|]} \sigma(a_p^\diamond, b_p^\diamond)$$

where σ is the similarity score for all subalignments, and $\zeta_{a_i b_k, a_j b_l}$ is a cost for aligning i with k and j with l when (i, j) and (k, l) are basepairs.

For example:

$\mathcal{A} =$ 

$$\sigma(\text{A},\text{C}) + \sigma(\text{C},\text{C}) + \sigma(\text{G},\text{G}) + \sigma(\text{U},-) + \sigma(-,\text{A}) + \sigma(\text{G},\text{G}) + \sigma(-,\text{A}) + \sigma(\text{U},\text{G}) + \zeta_{\text{ACUG}} + \zeta_{\text{CCGG}}$$

$$\mathcal{A} = \{(1,1), (2,2), (3,3), (5,5), (6,7)\}$$

Definition 3.3.1.c: (FA1.0 problem) Given two sequences A and B as input, the FA1.0 problem is the problem to find an alignment $\mathcal{A}$ of the sequences A and B, *and* a nested structure P_A of sequence A and a nested structure P_B of sequence B, such that the basepairs of the structures P_A and P_B are preserved in the aligned positions, and with a constraint of non-branching structures and FA1.0-score $(\mathcal{A}, P_A, P_B)$ is maximized.

As is shown the recursion of Nussinov algorithm (in the preliminaries chapter) allows for branching structures. Here, in order to reduce the time complexity for the Foldalign method, the case related for the branching structures is dropped.

Formally, this method finds the best subsequence alignment between two sequences by using the 4-D dynamic programming algorithm according to Gorodkin et al. [GHS97b] as follows:

Definition 3.3.1.d: (FA1.0 recursion) Let two subsequences $a_i \dots a_j$ and $b_k \dots b_l$, and a fixed FA1.0 scoring matrix ($S_{a_i a_j, b_k b_l}$), be given with a non-branching structures constraint, then the output is the highest scoring subsequence alignment of matrix *D*, that is produced by the following recursion:

$$D_{ij,kl} = max \begin{cases} D_{(i+1)(j-1),(k+1)(l-1)} + S_{a_i a_j, b_k b_l}, & (a) \\[2ex] D_{i(j-1),(k+1)(l-1)} + S_{-a_j, b_k b_l}, & \\ D_{(i+1)j,(k+1)(l-1)} + S_{a_i -, b_k b_l}, & (b) \\ D_{(i+1)(j-1),k(l-1)} + S_{a_i a_j, -b_l}, & \\ D_{(i+1)(j-1),(k+1)l} + S_{a_i a_j, b_k -}, & \\[2ex] D_{(i+1)(j-1),kl} + S_{a_i a_j, --}, & (c) \\ D_{ij,(k+1)(l-1)} + S_{--, b_k b_l}, & \\[2ex] D_{(i+1)j,(k+1)l} + S_{a_i -, b_k -}, & (d) \\ D_{i(j-1),k(l-1)} + S_{-a_j, -b_l}, & \\[2ex] D_{(i+1)j,k(l-1)} + S_{a_i -, -b_l}, & (e) \\ D_{i(j-1),(k+1)l} + S_{-a_j, b_k -}, & \\[2ex] D_{(i+1)j,kl} + S_{a_i -, --}, & \\ D_{i(j-1),kl} + S_{-a_j, --}, & (f) \\ D_{ij,(k+1)l} + S_{--, b_k -}, & \\ D_{ij,k(l-1)} + S_{--, -b_l} & \end{cases}$$

where a maximum value of $D_{ij,kl}$ gives the maximal similarity between the subsequences $a_i \dots a_j$ and $b_k \dots b_l$.

The different classes are represented by the letters from (a) to (f) on the right side of the above recursion, according to the number of gaps and its distribution within the alignment. As is shown, in this recursion the branching structures are not allowed, in order to reduce the time complexity.

Note that zero value is found in the recursion of local sequence alignment, but it is not included in this recursion because the matrix $D_{ij,kl}$ contains alignment scores over all the subsequences, $a_i \dots a_j$ and $b_k \dots b_l$, also because the negative values are allowed to be included within the complete alignment.

Definition 3.3.1.e: (FA1.0 scoring matrix construction) $S_{ij,kl}$ is the FA1.0 scoring matrix that is constructed for the subsequences i, …, j and k, …, l, from two terms of the independent contributions, so *S* describes the sum of the two matrices $\mathcal{A}$ and *B*, as follows: $S = \mathcal{A} + B$,

Where *S* is a *scoring matrix 1.0* that substitutes any pair of bases with the other including gaps, $\mathcal{A}$ is a matrix for sequence alignment and *B* is a matrix for basepairs alignment.

First, a matrix $\mathcal{A}$ will be constructed from two independent matrices, such that, it contains all pairs that are possible in one sequence of positions (i, j) and in other sequence of positions (k, l), by combining the cost of $(\mathcal{A}_0)_{ik}$ that aligns i with k and the cost $(\mathcal{A}_0)_{jl}$ that aligns with l, as shown in:

$$\mathcal{A}_{ij,kl} = (\mathcal{A}_0)_{ik} + (\mathcal{A}_0)_{jl}$$

where $\mathcal{A}_{ij,kl}$ represents the score matrix for aligning any two bases in one sequence with any two bases in the other sequence with gaps including. $(\mathcal{A}_0)_{ik}$ and $(\mathcal{A}_0)_{jl}$ are similarity substitution matrices.

Now, to build the score matrix *B* for base pairing, a simple description is introduced as follows [(*)]:

$$B_{ij,kl} = \begin{cases} \zeta_{ikjl} \text{ if (i, j) and (k, l) can basepairs} \\ 0 \text{ otherwise} \end{cases}$$

[(*)] There is another presentation for the score matrix *B*, and one example for the matrix *S*, [GHS97b].

where ζ is a base pairing alignment matrix that gives a score for substituting a basepair of one sequence to a basepair of other sequence, its values gives reason to occurring compensating mutations in the final matrix *S*.

The 4-D dynamic programming of pairwise Foldalign 1.0 will extend to contain alignment for two entities (individual sequences and/or aligned sequences) in a set of RNA sequences, without overlapping of sequences between them; therefore, the Greedy algorithm will be used for this extension to construct the multiple alignments from pairwise comparisons that are optimized by the pairwise Foldalign 1.0 for the preservation of both sequence and structure.

However, in a set of n sequences, there might be some of these sequences not-related to the rest or might be functionally related but denote the two or more of structural classes that do not provide a single common motif over all sequences. The overall method (i.e. pairwise Foldalign 1.0 and Greedy algorithm) distinguishes that there are $m \leq n$ sequences including the most significant common motif in the alignment, while the rest of sequences might refer to other structural class. Hence, it considers that there are 2^n subsets for n sequences, and it identifies the subset which is the most significant common motif, while neglecting the other non-useful subsets.

The Greedy algorithm is described in two steps according to Gorodkin et al. [GHS97b] as follows:

- Comparing all individual sequences with each other, and then comparing all pairwise alignments with all individual sequences, as long as in each comparison a sequence does not appear more than once.
- All triplet alignments align with individual sequences, and all pairwise alignments compare with each other, again as long as in each comparison a sequence does not appear more than once.

By continuing with this algorithm, all sequences will be compared at the end of the alignment. It requires time $\mathcal{O}(N^4n^n)$ where N is the sequence length for n sequences (i.e. exponential time), and as mentioned above some of sequences (subsets) are improbable to be involved in the final aligned subset, therefore, such procedure requires discarding non-useful alignments (aligned subsets).

There are two limitations on the comparisons that are used to optimize this algorithm; therefore, they reduce the time complexity to $\mathcal{O}(N^4n^2)$. These are: (1) a single sequence which always one of the two entities, and (2) there is “threshold” number of the highest scoring alignments at each round that is stored.

Explanation: Considering "threshold" 30, and comparing each single sequence with each pairwise alignment create triplet alignments and only the 30 of the best scoring alignments are stored to comparing again with single sequences to create four sequences alignments. With these limitations on the comparisons the complexity becomes $\mathcal{O}(N^4n^2)$.

In addition to tractability over the Greedy algorithm, Foldalign method has another advantage that it can find the subsets (aligned sequences) with most significant alignments. The disadvantage of the Foldalign method is that there is no guarantee to find the optimal solution, as in Sankoff algorithm.

During the implementation of the pairwise Foldalign 1.0, two limitations are showed, these are defined as follows:

- **Definition 3.3.1.f**: (δ-restriction) The maximum scoring alignments $D_{ij,kl}$ for the subsequences i, …, j and k, …, l are calculated, if restricting the maximum length of difference between these two subsequences being aligned by *δ nucleotides*.

Only $D_{ij,kl}$, where $|(j - i + 1) - (l - k + 1)| \leq \delta$ is satisfied, are calculated.

Otherwise $D_{ij,kl} = -\infty$, if $|(j - i + 1) - (l - k + 1)| > \delta$.

- **Definition 3.3.1.g**: (λ-restriction) The maximum scoring alignments $D_{ij,kl}$ for the subsequences i, …, j and k, …, l are calculated, if restricting the maximum RNA-motif length by *λ nucleotides*.

$(j - i + 1) \leq \lambda$

$(l - k + 1) \leq \lambda$

where i, j, k, l, are indices of $D_{ij,kl}$.

Since these two heuristics have effect on the alignment length of the problem that is solved, they will reduce the time and memory complexities.

3.3.2 Pairwise Foldalign 2.0

The main limitation of the pairwise Foldalign 1.0 implementation includes only the stem-loop structures because of the computational complexity. Therefore, several improvements are applied to extend and improve the Foldalign method.

Now a new Foldalign implementation suggested by Havgaard et al. [HLS+G05] will be discussed. This pairwise Foldalign 2.0 implementation extends from the previous implementation to include: the bifurcated structures, structural parameters provided in the scoring scheme that employs for free energy minimization (similar to energy terms in Dynalign) [MSZ+T99] [XSB+KSJ98], and also contains computation of the substitution matrices that is similar to RIBOSUM [KE03].

Now, we present the following definitions for the pairwise Foldalign 2.0 according to Havgaard et al. [HLS+G05] as follow:

Definition 3.3.2.a: (FA2.0 score) Let $\mathcal{A}$ be an alignment of the sequences A and B, let P_A be a nested structure of sequence A and let P_B be a nested structure of sequence B. Then the *FA2.0-score* of $\mathcal{A}$, P_A and P_B is given as:

$$\text{FA2.0-score }(\mathcal{A},\ P_A,\ P_B) = \sum_{\substack{(i,j)\in P_A,(k,l)\in P_B \\ where\ (i,j) aligned\ (k,l) by\ \mathcal{A}}} [\tau(a_i, a_j;\ b_k, b_l)] + E_A(P_A) + E_B(P_B) + \sum_{(i,k)\in\mathcal{A}} \sigma(a_i, b_k)$$

where τ and σ are the similarity parameters for substituting base-pairs and unpaired bases respectively, which are similar to RIBOSUM matrices. E(P_A) and E(P_B) are subset of Turner energies used as the energy parameters that compute the free energy minimization [MSZ+T99], [ZMT99].

Definition 3.3.2.b: (FA2.0 problem) Given two sequences A and B as input. The FA2.0 problem is the problem to find an alignment $\mathcal{A}$ of the sequences A and B, *and* a nested structure P_A of sequence A and a nested structure P_B of sequence B, such that the basepairs are conserved in the aligned positions and FA2.0-score ($\mathcal{A}$, P_A, P_B) is maximized.

As mentioned above, that the branching structures are included in this version of Foldalign.

The FA2.0 cost ($S_{a_i a_j, b_k b_l}$) is a cost for the substitution of (a_i, a_j) from sequence A with (b_k, b_l) from sequence B, and the two subsequences are folding simultaneously. It has a dynamical computation relying on the five structural contexts (structure elements).

Definition 3.3.2.c: (FA2.0 recursion) Let two subsequences $a_i \dots a_j$ and $b_k \dots b_l$ be given as input. FA2.0 cost ($S_{a_i a_j, b_k b_l}$) is calculated according to the structural context, and with including branching structures. Then the maximum scoring subsequence alignments of matrix *D* is produced by the following recursion according to Havgaard et al. [HLS+G05] as follow:

$$D_{ij,kl} = max \begin{cases} D_{(i+1)(j-1),(k+1)(l-1)} + S_{a_i a_j, b_k b_l}\,, & (a) \\[1ex] D_{i(j-1),(k+1)(l-1)} + S_{-a_j, b_k b_l}\,, & \\ D_{(i+1)j,(k+1)(l-1)} + S_{a_i -, b_k b_l}\,, & (b) \\ D_{(i+1)(j-1),k(l-1)} + S_{a_i a_j, -b_l}\,, & \\ D_{(i+1)(j-1),(k+1)l} + S_{a_i a_j, b_k -}\,, & \\[1ex] D_{(i+1)(j-1),kl} + S_{a_i a_j, --}\,, & (c) \\ D_{ij,(k+1)(l-1)} + S_{--, b_k b_l}\,, & \\[1ex] D_{(i+1)j,(k+1)l} + S_{a_i -, b_k -}\,, & (d) \\ D_{i(j-1),k(l-1)} + S_{-a_j, -b_l}\,, & \\[1ex] D_{(i+1)j,k(l-1)} + S_{a_i -, -b_l}\,, & (e) \\ D_{i(j-1),(k+1)l} + S_{-a_j, b_k -}\,, & \\[1ex] D_{(i+1)j,kl} + S_{a_i -, --}\,, & \\ D_{i(j-1),kl} + S_{-a_j, --}\,, & (f) \\ D_{ij,(k+1)l} + S_{--, b_k -}\,, & \\ D_{ij,k(l-1)} + S_{--, -b_l}\,, & \\[1ex] \max\limits_{\substack{i<n<j-1 \\ k<m<l-1}} \{D_{in,km} + D_{(n+1)j,(m+1)l}\} & (g) \end{cases}$$

where maximal $D_{ij,kl}$ refers to the most similar subsequences $a_i \dots a_j$ and $b_k \dots b_l$, and each of a_i, a_j, b_k, b_l are nucleotides at positions i, j, k, l, respectively.

The letters above from (a) to (f) on the right-side of recursion are the same as those in the previous implementation, only (g) is added to allow for the bifurcation structures. For each D-entry, the associated context is stored in a separate matrix.

Generally, $S_{a_i a_j, b_k b_l}$ has a dynamical computation by relying on the five structural contexts (structure elements). The parameters used in this computation are static such as substitution costs of basepair (aligning part) and the energy parameters (folding part). Now, the calculations of cost $S_{a_i a_j, b_k b_l}$ for each structural context are presented according to Havgaard et al. [HLS+G05] as follows:

- Hairpin-loop: the calculation is always initialized by aligning two nucleotides in the hairpin-loop context. The cost of the alignment between two hairpin-loops is:

  $$S_{\text{hp}} = S_{\text{substitution}} + S_{\text{length}} + S_{\text{stack}}$$

  where $S_{\text{substitution}} = \sum S_{\text{SS}}(a_i, b_k)$ is the cost of combining the substitution for each pair of nucleotides and gap cost for each gap that are included in the loop. $S_{\text{SS}}(a_i, b_k)$ is the single-strand substitution cost, which correspondences to σ. $S_{\text{length}} = S_{hp\text{-length}}(j - i + 1) + S_{hp\text{-length}}(l - k + 1)$ is the cost that is dependent on the loop size, which is computed from the energy parameters. For the hairpin-loops that have more than three nucleotides long, the energy cost $S_{\text{stack}} = S_{hp\text{-stack}}(a_i, a_j, a_{i-1}, a_{j+1}) + S_{hp\text{-stack}}(b_k, b_l, b_{k-1}, b_{l+1})$ is combining of two independent sums for stacking in the two hairpin-loops.

- Stem: A stem is the number of stacked basepairs, with long at least two basepairs. A single basepair is not allowed and is recalculated as part of the surrounding loop. The cost of the alignment between two stems is:

  $$S_{\text{bp}} = S_{\text{substitution}} + S_{\text{stack}}$$

  where $S_{\text{substitution}} = \sum S_{\text{SS}}(a_i, a_j, b_k, b_l)$ is the cost of combining the substitutions for the basepairs in one subsequence with the basepairs in other subsequence, which correspondences to τ. S_{stack} is the stack energy cost which has the same computation above but for two stems.

The other structural contexts are computed in the same manner with some addition costs of energy parameter that are added according to the requirement of structural context.

- Internal-loop: An internal-loop is the single-stranded nucleotides on both sides of RNA structure that are surrounded by stems. The cost of aligning two internal-loops is:

$$S_{\text{il}} = S_{\text{substitution}} + S_{\text{length}} + S_{\text{asymmetry}} + S_{\text{stack}}$$

- Bulge-loop: A bulge-loop is also single-stranded region but only on one side of RNA structure that is surrounded by stems. The cost of aligning two bulge-loops is:

$$S_{\text{bl}} = S_{\text{substitution}} + S_{\text{length}} + S_{\text{stack}}$$

- Multibranched-loop: A multibranched-loop is the region where more than two stems meet. The cost of a multibranched-loop is:

$$S_{\text{hp}} = S_{\text{substitution}} + S_{\text{mbl-closing}} + (n_{\text{stem}} - 2)\, S_{\text{stem}} + n_{\text{singlenucleotides}} \times S_{\text{nucleotide}} + S_{\text{stack}}$$

All these costs are stored in either specific matrices or tables in the score matrix according to structural context and its parameters. For more details, the reader is referred to the paper Havgaard et al. [HLS+G05].

The two constraints that are used in the previous implementation of FA1.0, they are also used with this implementation of FA2.0: λ and δ. These reduce the complexity for each of time $N_A\ N_B\ \lambda^2\ \delta^2$ and memory $N_A\ N_B\ \lambda\ \delta$, where N_A and N_B are the lengths of A, B sequences respectively.

By dropping the cases (b) and (e) from the recursion above, the speed of this implementation will be increased because the number of cases at each entry in recursion will be reduced. This has no influence on the memory and time complexities because there is no modification of the structure types which can be aligned, so that we can obtain the alignments for the cases (b) and (e), by integrating some of the other cases of the recursion. For example, the case (b.1) can be obtained from combining the cases (f.3) and (d.2), or from (e.2) and (f.4), or replacing of other cases.

The restriction has been placed on the case (g) when it is calculated, where the case (g) is composed of two substructures, another optimization conforms to speed of this implementation, if (i, n) and (k, m) are basepairs of the left substructure $D_{in,km}$, and if j is base-paired and l is base-paired of the right substructure $D_{(n+1)j,(m+1)l}$.

This implementation handles with the problem that finds the common local structural motifs of two RNA sequences with sequence similarity less than 40%, where these sequences are not distinguishable of the folding energy to their surrounding sequence context. It also represents an efficient way for executing simultaneous mutual scan for two sequences to find the common local structural motifs.

3.3.3 Pairwise Foldalign 2.1

Now, we will introduce the last implementation of the Foldalign method that was described by Havgaard et al. [HTG07], as a new heuristic in the previous implementation (i.e. Foldalign 2.0). This heuristic is represented by the dynamical pruning of the dynamic programming matrix, through excluding the subalignments that have scores lower than length-dependent threshold (pruning threshold). This heuristic increases the speed without reducing in the predictive performance. It represents a new implementation in the Foldalign method that is used for pairwise local or global structural alignments of the RNA sequences. In addition the memory requirement is reduced by a constraint of branch points which uses the divide and conquers method. This thesis is not interested in the latter.

Now the following definitions are presented for the pairwise Foldalign 2.1 according to Havgaard et al. [HTG07] as follows:

Definition 3.3.3.a: (FA2.1-score) FA2.1 score has the same definition as the previous Foldalign implementation (FA2.0 score).

Definition 3.3.3.b: (FA2.1 problem) FA2.1 problem has also the same definition as the previous Foldalign implementation (FA2.0 problem).

The cost of FA2.1 distinguishes also into several costs that are used to add a set of nucleotides to the alignment. This cost is the same as FA2.0 cost which has a dynamical computation depending on the five structural contexts (structural elements).

Definition 3.3.3.c: (FA2.1 recursion) Let two subsequences be given as input $a_i \ldots a_j$ and $b_k \ldots b_l$ of sequences A and B respectively. This recursion seems the same recursion as the previous version with only a few improvements or simplifications to the energy model. Then maximum scoring subsequence alignments of *D* is produced according to Havgaard et al. [HTG07] as follows:

$$D_{i,j,k,l} = max \begin{cases} D_{i+1,j-1,k+1,l-1} + S_{bp}(a_i, a_j, b_k, b_l, \sigma_{i+1,j-1,k+1,l-1}) & \text{(a)} \\ D_{i+1,j-1,k,l} + S_{bpiI}(a_i, a_j, -, -, \sigma_{i+1,j-1,k,l}) & \text{(b)} \\ D_{i,j,k+1,l-1} + S_{bpiK}(-, -, b_k, b_l, \sigma_{i,j,k+1,l-1}) & \text{(c)} \\ D_{i+1,j,k+1,l} + S_{al}(a_i, b_k, \sigma_{i+1,j,k+1,l}) & \text{(d)} \\ D_{i,j-1,k,l-1} + S_{ar}(a_j, b_l, \sigma_{i,j-1,k,l-1}) & \text{(e)} \\ D_{i+1,j,k,l} + S_{glI}(a_i, -, \sigma_{i+1,j,k,l}) & \text{(f)} \\ D_{i,j-1,k,l} + S_{grI}(a_j, -, \sigma_{i,j-1,k,l}) & \text{(g)} \\ D_{i,j,k+1,l} + S_{glK}(-, b_k, \sigma_{i,j,k+1,l}) & \text{(h)} \\ D_{i,j,k,l-1} + S_{grK}(-, b_l, \sigma_{i,j,k,l-1}) & \text{(i)} \\ \max\limits_{\substack{i<m<j \\ k<n<l}} \{D'_{i,m,k,n} + D'_{m+1,j,n+1,l} + C_{mblhelix}\} & \text{(j)} \end{cases}$$

where $D_{i,j,k,l}$ is the alignment score, $\sigma_{i,j,k,l}$ is the alignment state. In addition to these matrices, there are four length matrices used in the implementation: $\mu_{1(i,j,k,l)}$, $\mu_{2(i,j,k,l)}$, $\mu_{3(i,j,k,l)}$, $\mu_{4(i,j,k,l)}$ which are the lengths of the single stranded regions external to the last basepairs. Therefore, this version contains six of 4-D matrices that are required for computing the recursion. $C_{mblhelix}$ which is the cost for adding extra stems.

In case (j), the unpaired nucleotides of branched loops score the same as unpaired nucleotides in the external loops. Hence, *D'* is the alignment score that is corrected for external single stranded nucleotides.

As mentioned that cost of FA2.1 represents the costs of S_{bp} to S_{grK} which are computed depending on the alignment state (σ) which includes five structural contexts: hairpin-loop, stem, bulge-loop, internal-loop, and external/bifurcated-loop.

The case (a) adds a basepair in both structures. The cases (b) and (c) add basepair inserts in either of the structures. The cases (d) and (e) add aligned unpaired nucleotides in either end of the alignment. The cases (f) and (i) add an unpaired nucleotide aligned to a gap to the alignment. The case (j) is the bifurcation case which joins two substructures into one in each of the structures.

The alignment score $D_{i,j,k,l}$ is the maximum alignment score over all the *D*-entries and the alignment state $\sigma_{i,j,k,l}$ becomes the state for the best structure alignment which is computed according to the associated context of *D*-entry, where the context of each *D*-entry is stored in a separate matrix. Analogously, this is done for the length matrices: $\mu_{1(i,j,k,l)}$, $\mu_{2(i,j,k,l)}$, $\mu_{3(i,j,k,l)}$, $\mu_{4(i,j,k,l)}$ which are the lengths of single stranded regions for the best alignment of (i, j, k, l). These lengths are updated according to the associated context. For more details about the procedure of recursion, the reader is referred to the supplementary material [PS1] for Havgaard et al. [HTG07].

The construction of Foldalign compares with the Sankoff algorithm, that Sankoff has three matrices which distinguish the different states in the structure. In the Foldalign, only one matrix uses several states which are distinguished in the structure. Therefore, the maximum *D*-entry from the Foldalign recursion should give a best state which is stored in a separate matrix.

Due to the general case of structure which is represented by matrix F in Sankoff algorithm which is not carried in Foldalign as obvious state, therefore, this state is corrected by the *D'*-entry, where $D'_{(i,j,k,l)} = D_{(i,j,k,l)} + S'(\sigma_{(i,j,k,l)})$.

We applied some examples on this recursion where, without recalculation in the stem state, the recursion does not work or may not get the optimal solutions according to the observed states in this recursion. However, Foldalign 2.1 has been able practically (i.e. when it was run) to solve these examples which would not work in an optimal way without recalculation. This demonstrates the effect of the “potential basepair” state that was added in the implementation for realizing the recalculation. Although these special states are handled in this method still the method dose not guarantees an optimal solution.

The following two examples were run in Foldalign 2.1 and we got their solutions.

Example 1:

```
>Seq_1
CCAAAAAUGG
>Seq_2
CCAAAAAUGG

; ALIGN          Seq_1        CCAAAAAUGG
; ALIGN          Structure    ( ( (  .  .  .  .   ) ) )
; ALIGN          Seq_2        CCAAAAAUGG

; ALIGNING           Seq_1 against Seq_2
; STEM END 1 10 1 10 ; START 0 0 0 0 SCORE -20000
; BACKTRACK          115 115 (1 10, 1 10) 21 0 0 0 0 Basepair ik jl
; BACKTRACK          23 23 (2 9, 2 9) 21 0 0 0 0 Basepair ik jl
; BACKTRACK          -45 -45 (3 8, 3 8) 20 0 0 0 0 Hairpin -> stem ik jl
; BACKTRACK          32 -72 (4 7, 4 7) 2 4 0 4 0 Hairpin ik
; BACKTRACK          24 -84 (5 7, 5 7) 2 3 0 3 0 Hairpin ik
; BACKTRACK          16 -94 (6 7, 6 7) 2 2 0 2 0 Hairpin ik
; BACKTRACK          8 -104 (7 7, 7 7) 1 1 0 1 0 Initial Hairpin ik
; BACKTRACK          Branch end
```

Example 2:

```
>Seq_1
GCGAAAAUGC
>Seq_2
GCGAAAAUGC

; ALIGN          Seq_1       GCGAAAAUGC
; ALIGN          Structure   ( ( . . . . . . ) )
; ALIGN          Seq_2       GCGAAAAUGC

; STEM END 1 10 1 10 ; START 0 0 0 0 SCORE -20000
; BACKTRACK         87 87 (1 10, 1 10) 21 0 0 0 0 Basepair ik jl
; BACKTRACK         -9 -9 (2 9, 2 9) 20 0 0 0 0 Hairpin -> stem ik jl
; BACKTRACK         39 -57 (3 8, 3 8) 2 6 0 6 0 Hairpin ik
; BACKTRACK         36 -66 (4 8, 4 8) 2 5 0 5 0 Hairpin ik
; BACKTRACK         28 -76 (5 8, 5 8) 2 4 0 4 0 Hairpin ik
; BACKTRACK         20 -88 (6 8, 6 8) 2 3 0 3 0 Hairpin ik
; BACKTRACK         12 -98 (7 8, 7 8) 2 2 0 2 0 Hairpin ik
; BACKTRACK         4 -108 (8 8, 8 8) 1 1 0 1 0 Initial Hairpin ik
; BACKTRACK         Branch end
```

As mentioned above, the dynamical pruning works to eliminate all subalignments that are at the poorly levels, and that occurs by comparing the score $D_{ij,kl}$ of a subalignment with a threshold of local alignment described as follows:

$$D_{ij,kl} \text{ is pruned if } D_{ij,kl} < \Theta_{local}\,(l_A) \text{ or } D_{ij,kl} < \Theta_{local}\,(l_B)$$

which is equivalent to $D_{ij,kl} < \min \Theta_{local}\,((l_A), (l_B))$

where Θ_{local} is based on the length of the subsequences $l_A = (j - i + 1)$ and $l_B = (l - k + 1)$, therefore, a linear form that is found for the proper length dependency is, $\Theta_{local} = a * \min \{l_A, l_B\} + b$, where a and b are constants.

This speeds up the Foldalign 2.1 method, moreover the memory is also improved because it does not need to store the discarding subalignments which are also not used to calculate the longer alignments.

Due to global alignment considering the whole length of sequence, a minimum number of gaps must be added equal to the length difference between the two sequences. The pruning for local alignment will eliminate all subalignments, when the difference lengths are large. Therefore, the special pruning for global alignment is employed as follows:

$$D_{ij,kl} < \Theta_{global} = \Theta_{local}\,(l_A, l_B) + G_E \times \min\,\{abs(l_A - l_B), abs(N_A - N_B)\}$$

Where Θ_{global} threshold for global alignment, G_E is the cost of gap-elongation, Θ_{local} is the threshold of local alignment and N_A and N_B are the sequence lengths of A and B respectively. Here, other values for parameters a and b are used.

The dynamical Pruning represents as a general heuristic and it should be possible to be employed with the other methods that implement the fold and alignment of RNA sequences; it is considered a property in dynamic programming method that is applied with algorithms exploiting dynamic programming. However, it does not ensure that it provides an optimal solution, or in some cases no solution is found, therefore the Foldalign in this case will realign without pruning.

This implementation still applies the old constraints λ and δ, which decrease the complexity to $\mathcal{O}\,(N_A\,N_B\,\lambda^2\,\delta^2)$ for time and $\mathcal{O}\,(\lambda^3\,\delta)$ for memory, where N_A and N_B are the sequence lengths A and B respectively. Furthermore, the bifurcation constraint is also employed in this implementation, which restricts the substructure types that are combined in case (j), such that the first nucleotides (i and k) in the substructure $D_{in,km}$ are base paired, and in substructure $D_{n+1j,m+1l}$, the pairs of bases (n + 1, j) and (m+1, l) must be basepairs (i.e. they should form a stem context). Hence, this constraint restricts all alignments that must be kept for positions (i + 2, …, i + λ) to those which have stem context. This represents an optimization during the local alignment for saving memory.

When using the global alignment in this implementation, the complexity of time and memory are reduced compared with the previous implementation. The global alignment is aligning over the entire length of two sequences, such that the δ is becoming exactly as the parameter M in Dynalign method by restricting the starting of a sub-alignment in the second sequence (i.e. $|i - k| \leq \delta$). The idea is that since the δ-heuristic limits the length difference of sub-alignments, then the position k in the second sequence is limited in relation to the position i in the first sequence. The complexity becomes $\mathcal{O}$ ($N_{min}^3\delta^3$) for time and $\mathcal{O}$ ($N_{min}^2\ \delta^2$) for memory, where $N_{min} = \min\{N_A, N_B\}$. This compared with the complexity of the previous implementation that applies the local alignment, where λ refers to the length of the sequence.

The energy model in this Foldalign implementation changes as compared with the previous implementation (2.0) in three points according to Havgaard et al. [HTG07]:

- The single-stranded nucleotides in external-loops are scored like the single-stranded nucleotides in the multibranch-loops.
- Allowing insert the basepairs at all positions of a stem excepting the first basepair.
- The single-stranded nucleotides in the multibranch-loops that are next to base-paired nucleotides are no longer stacked, i.e. the dangling ends are no longer used.

3.4 PMcomp / LocARNA

This part will present another family of variants of the Sankoff algorithm for simultaneous folding and aligning of two RNA sequences, which exploit the probabilities of basepairs for RNA sequences as structural input. Therefore, they take into account the information about both sequence and structure, where the secondary structures are non-pseudoknoted. These methods are PMcomp [HBS04] and LocARNA [WRH+SB07].

Before we start to show each of these methods separately, some principles are introduced that are related to these methods.

McCaskill algorithm calculates the base pairing probabilities from the partition functions of RNA sequences. It uses a statistical mechanics model to predict the probabilities of individual basepairs in the secondary structure. [M90]

Definition 3.4: (Boltzmann weight, Partition function, Base pairing probability) Given an RNA sequence S, the *Boltzmann weight* of a structure P of S is defined as: $w_B^{(S)}(P) = e^{-E_S(P)/k_B T}$, where $E_S(P)$ is the energy of a structure P, k_B is Boltzmann constant and T is the temperature. The *partition function* of S is defined as: $Z_{P,S} = \sum_{P \text{ of } S} w_B^{(S)}(P)$. The *probability of a structure P of S* is defined as: $Pr\,[P \text{ of } S] = w_B^{(S)}(P)/Z_{P,S}$.

This kind of probability can be computed efficiently by using McCaskill algorithm.

Each of these methods utilizes the basepair weights which are derived from the matrices of basepair probability for each individual sequences, such as the weight ψ_{ij} for the basepair (i, j) of sequence S is described as:

$$\psi_{ij} = \log \frac{Pr_{ij}}{p_0} \Big/ \log \frac{1}{p_0}$$

Where Pr_{ij} is the probability of a basepair (i, j) as calculated by McCaskill algorithm, p_0 is the expected probability for base pairing that is randomly occurring.

Both of these methods calculate the pairwise alignment from the base pairing probability matrices of the RNA sequences, where the McCaskill algorithm computes these matrices. Therefore, these matrices include the energy information for each sequence (as shown above).

When these methods are compared with the Sankoff algorithm, they are based on the input of base pairing probability matrices which contain the energy information about RNA sequences and can be calculated independently.

These methods do not start directly from only the sequences but they require their base pairing probability matrices which are important for the folding part in these methods. Furthermore, these methods do not distinguish among all structure elements. Since, the implementation of these methods depends on the simple scoring system, such that they avoid implementing and computing the complete energy model of RNA folding during alignment.

3.4.1 PMcomp

As we mentioned above, this method calculates the pairwise alignment from the matrices of base pairing probability of the RNA sequences. These matrices are computed by using the McCaskill's algorithm. Thus, they include the energy information about each sequence. Then PMcomp finds the "maximal weight" common secondary structure together with the alignment between the sequences. This method was proposed by Hofacker et al. [HBS04].

Before defining this method formally, one must show how to find the "maximum weight" secondary structure that is common to two base pairing probability matrices:

Definition 3.4.1.a: (Consensus secondary structure "C. S. S.", Maximum weight C. S. S.) Let two sequences S_1 and S_2 be given as input with their base pairing probability matrices P^1 and P^2 respectively. The *consensus secondary structure* $\mathcal{S}$ is a set of pairs of basepairs (i, j) and (k, l) of sequences S_1 and S_2 respectively. The *maximum weight C. S. S.* is the consensus secondary structure $\mathcal{S}$ that maximizes:

$$\sum_{((i,j),(k,l))\in\mathcal{S}} \left(\psi_{ij}^{1} + \psi_{kl}^{2}\right)$$

where ${\psi_{ij}}^1$ and ${\psi_{kl}}^2$ are the weights of the basepairs (i, j) and (k, l) of sequences S_1 and S_2 respectively, as described above in the weight's equation. This definition does not consider solving the problem of simultaneous folding and aligning. Moreover, it produced structures that are different from the structures that are formed in the PMcomp method.

The PMcomp problem is defined with respect to its score according to Hofacker et al. [HBS04]:

Definition 3.4.1.b: (PMcomp-score) Let two sequences $S_1 = s_{1_i}, \ldots, s_{1_j}$and $S_2 = s_{2_k}, \ldots, s_{2_l}$ be given as input with their base pairing probability matrices Pr^1 and Pr^2 respectively. Let $\mathcal{A}$ be an alignment of sequences S_1 and S_2, and let the consensus secondary structure $\mathcal{S}$ of sequences S_1 and S_2. The *PMcomp-score* of $\mathcal{A}$ and $\mathcal{S}$ is given as:

$$\text{PMcomp-score } (\mathcal{A}, \mathcal{S}) = \sum_{(ij,kl)\in\mathcal{S}} \left[\psi_{ij}^1 + \psi_{kl}^2 + \tau\left(s_{1_i}, s_{1_j};\ s_{2_k}, s_{2_l}\right)\right] + \gamma N_{gap} + \sum_{\substack{i\in S_1, k\in S_2 \\ and\ i,k\ aligned\ by\ \mathcal{A}}} \sigma\left(s_{1_i}, s_{2_k}\right)$$

where τ and σ are scores of alignment contributions for substituting the basepairs and unpaired bases respectively, γ is the gap penalty and N_{gap} is the number of gaps during insertion and deletion of the alignment.

Definition 3.4.1.c: (PMcomp problem) Let two sequences $S_1 = s_{1_i}, \ldots, s_{1_j}$and $S_2 = s_{2_k}, \ldots, s_{2_l}$be given as input with their base pairing probability matrices Pr^1 and Pr^2 respectively. The *PMcomp problem* finds the consensus secondary structure $\mathcal{S}$ of S_1 and S_2, *and* an alignment $\mathcal{A}$ of S_1 and S_2 with the number of gaps during insertion and deletion of the alignment, such that PMcomp-score ($\mathcal{A}$, $\mathcal{S}$) is maximized.

Now this method defines the best subsequence matching alignments by using dynamic programming algorithm according to Hofacker et al. [HBS04]:

Definition 3.4.1.d: (PMcomp recursion) let two subsequences $S_1 = s_{1_i}, \ldots, s_{1_j}$and $S_2 = s_{2_k}, \ldots, s_{2_l}$ be given as input with their base pairing probability matrices Pr^1 and Pr^2 respectively, in addition to the τ and σ scores and the gap penalty γ. Then the output is the maximum scoring subsequence matching of matrix $S_{i,\,j;\,k,\,l}$ that is obtained by the following recursions:

$$S_{i,j,k,l} = max \begin{cases} S_{i+1,j;k,l} + \gamma \ , \\ S_{i,j;k+1,l} + \gamma \ , \\ S_{i+1,j,k+1,l} + \sigma\left(s_{1_i}, s_{2_k}\right), \\ \max\limits_{h \le j, q \le l} \left(S^M_{i,h;k,q} + S_{h+1,j;q+1,l}\right) \end{cases}$$

$$S^M_{i,j;k,l} = S_{i+1,j-1,k+1,l-1} + \psi^1_{ij} + \psi^2_{kl} + \tau\ \left(s_{1_i}, s_{1_j};\ s_{2_k}, s_{2_l}\right),$$

Initialization $S_{i,j;k,l} = |(j - i) - (l - k)|\,\gamma$ for $j - i \le M + 1$ or $l - k \le M + 1$, where M is the minimum size of hairpin loop (usually $M = 3$).

As presented at the initialization case in the original presentation according to Hofacker et al. [HBS04] is wrong, because $S_{i,j;k,l}$ must be the best score for s_{1_i}, …, $s_{1_j}$and s_{2_k}, …, s_{2_l} . In the alignment one can match bases, insert, or delete, but cannot match basepairs due to short one of the two sequence lengths.

A maximal value of $S_{i,j;k,l}$ gives the most matching for the subsequences s_{1_i}, …, $s_{1_j}$and s_{2_k}, …, s_{2_l}. In addition, the score $S^M_{i,j;k,l}$ be the best match subject with a constraint that the basepairs (i, j) and (k, l) are matched.

The first two cases in the recursion account for gaps in one of the subsequences, the third case refers to match the unpaired bases in both subsequences and the fourth case (max-case) refers to the basepairs (i, h) and (k, q) in the subsequences S_1 and S_2 respectively, which are matched. In addition, the restricted term of $S^M_{i,j;k,l}$ is straightforward.

This recursion needs $\mathcal{O}(N^4)$ for memory and $\mathcal{O}(N^6)$ for time, where N is the sequence length. PMcomp is equivalent to a special version of the Sankoff algorithm (Nussinov-style), where:

$$Pr^1_{ij} = \begin{cases} 1 & if\ \ s_{1_i} \text{ and } s_{1_j}\ can\ form\ basepairs \\ 0 & \text{otherwise} \end{cases}$$

Analogously for Pr^2_{kl}.

There are two restrictions that reduce the complexity: the first restriction is that matching of the basepairs (i, j) ∈ S_1 and (k, l) ∈ S_2 must be within the difference $\Delta = |(j - i) - (l - k)|$, hence, the time complexity will decrease to $\mathcal{O}(N^5)$. The second restriction is that all partial alignments are limited within this difference, the complexity will decrease to $\mathcal{O}(N^4)$ for time and $\mathcal{O}(N^3)$ for memory. If Δ is high, there is no big decrease of the computation effort. Whereas a lower value of Δ, many significant alignment structures will be missing.

After filling the matrix $S_{i,j;k,l}$, backtracking is used to compute the matched positions of the sequences.

When 'average' basepair probability matrix that is described below according to Hofacker et al. [HBS04] is found, the PMcomp method will extend to construct the progressive multiple alignments by using the comparison of the base pairing probability matrices, this is called PMmulti method. The average basepair probability matrix is defined by:

$$Pr^{1\circ 2}_{p,q} = \begin{cases} \sqrt{Pr^1_{i_p,j_q} Pr^2_{k_p,l_q}} & \text{for matches} \\ 0 & \text{otherwise} \end{cases}$$

Where i_p and j_q are the positions in sequence S_1 corresponding to the positions p and q in the alignment. Analogously, k_p, l_q are defined for S_2.

PMmulti method is represented by repeatedly calling for the PMcomp for calculating all pairwise alignments, and then generates a guide tree from assembling the similarity scores by applying the weighted pair group clustering method. Finally, it aligns all alignments along guide tree.

3.4.2 LocARNA

This method is PMcomp-based that calculates the pairwise alignment of RNAs (optionally local), but is more efficient for the time and memory complexities, such that it reduces to $\mathcal{O}(N^2)$ for memory and $\mathcal{O}(N^4)$ for time. This is due to introducing the idea of significant basepairs which are defined by using cutoff-probability, compared to PMcomp method.

Formally, this method can be defined with respect to its score according to Will et al. [WRH+SB07]:

Definition 3.4.2.a: (LocARNA-score) Let $\mathcal{A}$ be an alignment of sequences $S_1 = s_{1_i}$, …, s_{1_j} and $S_2 = s_{2_k}$, …, s_{2_l}, and let the consensus secondary structure $\mathcal{S}$ on $\mathcal{A}$. Then, the *LocARNA score* of $\mathcal{A}$ and $\mathcal{S}$ is given as:

LocARNA-score $(\mathcal{A}, \mathcal{S}) = \sum_{(ij;kl)\in\mathcal{S}} (\psi^1_{ij} + \psi^2_{kl}) + \sum_{(i,k)\in A_s} \sigma(s_{1_i}, s_{2_k}) - \gamma N_{gap}$

where A_s represents the single-stranded part of the alignment (i.e. the unpaired bases) and the parameters (γ, N_{gap}, σ) are defined as in the PMcomp method. Note that the LocARNA-score is essentially the same as the PMcomp-score. τ is omitted only for presentation.

In this method, the weights $\psi_{ij}{}^{1}$, $\psi_{kl}{}^{2}$ are modified by introducing cutoff-probability. This represents the first modification in LocARNA as compared with PMcomp, such that:

$$\psi_{ij} = \begin{cases} log\frac{Pr_{ij}}{p_0} \;/\; log\frac{1}{p_0} & \text{if } Pr_{ij} \geq p^* \\ -\infty & \text{otherwise} \end{cases}$$

where p^* is the cutoff probability, such that the weight be ($-\infty$) for the probability lower than p^*. This modification reduces the time complexity to $\mathcal{O}(\mathrm{N}^4)$, by making p^* constant for different lengths N. Then each base can take part in at most $1/p^*$, so only $\mathcal{O}(1)$ basepairs.

Definition 3.4.2.b: (LocARNA problem) Let two sequences $S_1 = s_{1_i}$, …, s_{1_j} and $S_2 = s_{2_k}$, …, s_{2_l}, be given as input with their base pairing probability matrices Pr^1 and Pr^2 respectively. The *LocARNA problem* calculates an alignment $\mathcal{A}$ of S_1 and S_2 with the number of gaps during insertion and deletion of the alignment *and* the consensus secondary structure $\mathcal{S}$ on $\mathcal{A}$, such that, $\mathcal{A}$ contains a set of match/mismatch pairs,

$\mathcal{S}$contains a set of the conserved basepairs and the LocARNA score ($\mathcal{A}$, $\mathcal{S}$) is maximized.

The second modification in the LocARNA, which improves the PMcomp method, is modified the dynamic programming algorithm being used, in such a way that allows considering only significant basepairs which are produced by applying cut-off probability. Thus, the space complexity reduces to $\mathcal{O}(N^2)$.

Now, the dynamic programming recursion is defined according to Will et al. [WRH+SB07], as follows:

Definition 3.4.2.c: (LocARNA recursion) Let two sequences $S_1 = s_{1_i}, \ldots, s_{1_j}$and $S_2 = s_{2_k}, \ldots, s_{2_l}$,with their base pairing probability matrices Pr^1 and Pr^2 respectively, be given with σ and γ. Then output is the maximum scoring subsequences similarity of matrix *D*, that obtained by the following recursions for *M* and *D*:

$$M_{ij;kl} = max \begin{cases} M_{ij-1;kl-1} + \sigma(s_{1_i}, s_{2_k}) \\ M_{ij-1;kl} + \gamma \\ M_{ij;kl-1} + \gamma \\ max_{j'l'} M_{ij'-1;kl'-1} + D_{j'j;l'l} \end{cases}$$

$$D_{ij;kl} = M_{ij-1;kl-1} + \psi^1_{ij} + \psi^2_{kl}$$

where maximal $D_{ij;kl}$ provides most similarity between the subsequences [$s_{1_i}, \ldots, s_{1_j}$] and [$s_{2_k}, \ldots, s_{2_l}$], with condition that the basepairs (i, j) and (k, l) are parts to form the consensus secondary structure. $D_{ij;kl}$ are calculated and stored only for the considered significant basepairs. Hence, the $D_{ij;kl}$ entries are computed with fix left ends i and k and varying j and l. For computing all entries $D_{i\bullet;k\bullet}$, one needs only entries of $M_{ij;kl}$ for alignments that have left ends (i+1, k+1).

These matrices can be used for both of global and local alignments by calculating the recursion of $M_{0j;0l}$ for the global alignment, where the optimal score of the global alignment is over the entire sequence length (i.e. $M_{0|S_1|;\,0|S_2|}$). This is the same as in the PMcomp method. In the following, local alignment will be considered.

The best local alignment score is obtained by finding the maximal value of subsequence alignments. Therefore, $M_{0j;0l}$ recursion is extended to include the zero entry that cutting off the prefix alignments that are not related to the local alignment, i.e. the negative values which are dissimilar prefix alignments.

$$M_{ij;kl} = max \begin{cases} M_{ij-1;kl-1} + \sigma(s_{1_i}, s_{2_k}) \\ M_{ij-1;kl} + \gamma \\ M_{ij;kl-1} + \gamma \\ max_{j'l'} M_{ij'-1;kl'-1} + D_{j'j;l'l} \end{cases} \qquad \text{for } i > 0 \text{ or } j > 0$$

$$M_{0j;0l} = max \begin{cases} 0 \\ M_{0j-1;0l-1} + \sigma(s_{1_i}, s_{2_k}) \\ M_{0j-1;0l} + \gamma \\ M_{0j;0l-1} + \gamma \\ max_{j'l'} M_{0j'-1;0l'-1} + D_{j'j;l'l} \end{cases}$$

$$D_{ij;kl} = M_{ij-1;kl-1} + \psi^1_{ij} + \psi^2_{kl}.$$

LocARNA method is also like the PMcomp method in extending to construct the progressive multiple alignments from the pairwise alignments, and this is called mLocARNA method. This method has a different algorithm for calculating the "average" basepair probability matrix $Pr^{1\circ 2}$ as found in the PMmulti for the alignment of S_1 and S_2. As a result of PMmulti, most basepairs are eliminated during the alignment for many sequences due to the second case in its definition that is related to the gaps. Therefore, mLocARNA introduces the new definition that prevents undesirable effect for PMmulti

:

$$Pr^{1^o 2}_{pq} = \sqrt{\overline{Pr}^1_{pq} \times \overline{Pr}^2_{pq}}\ ,$$

where

$$\overline{Pr}^{1}_{pq} = \begin{cases} max\left(p_0, Pr^{1}_{i_p i_q}\right) & \text{for a match } p, q \\ p_0 & \text{otherwise} \end{cases}$$

Analogously, for the definition $\overline{Pr}^{2}_{pq}$.

Chapter Four

Results

4.1 Results

The previous chapter demonstrates the theoretical model of the methods that were implemented with different restrictions or heuristics to make the original algorithm “Sankoff Algorithm” more practical. In this chapter, examples will be given of testing the programs of these methods on a collection of RNA families and the computational results that have been obtained will be displayed.

The following table gives some examples of different length datasets of RNA families that were used in our tests, such as tRNA which represents a small RNA sequence and therefore should be an easy example of all programs, 5S_rRNA is slightly harder and Cobalamin is quite challenging, because it’s much longer.

All calculations performed on the programs, were performed in the same environment (with memory 3.7 GB, CPU 2.33 GHz and under Linux operating system).

We tested the programs on their some special parameters by setting them to some interested values. These interested values were compared with the user time. In this thesis, the focus is on the time behavior against some different parameters of the programs because we are interested in comparing the speed of these programs corresponding to the different values of their parameters. All programs were applied to three examples of different length datasets of RNA families which are shown in the following table.

Table 4.1: This table shows three examples of different length datasets of RNA families with their sequence similarity

RFAM	Sequences	Number of nucleotides	APSI (Average Pairwise Sequence Identity)
tRNA	>Z28209.1_4569-4498		56
	GCCCUUUUGGCCAAGUGGUAAGGCAUCGCACUCGUAAUGCGGGGAUCGUGGGUUCAAUUCCCACAGAGGGCA	72	
	>M68929.1_166929-166856		
	GGGCUUAUAGUUUAAUUGGUUCAAACGCACCGCUCAUAACGGUGAUAUUGUAGGUUCGAGUCCUACUAAGCCUA	74	
5S_rRNA	>X52300.1_5-122		37
	CCCCGUGCCUUUAGCGCCUCGGAACCACCCCACUCCAUGCCGAACUGGGUCGUGAAACGUGGCAGCGCCUAUGAUACUUGGACCGCAGGGUCCUGGAAAAGUCGGUGCAGUGCGGGGG	118	
	>M19950.1_1-120		
	GGUUGCGGCCAUAUCUAGCAGAAAGCACCGUUUCCCGUCCGAUCAACUGUAGUUAAGCUGCUAAGAGCCUGACCGAGUAGUGUAGAGGGCGACCAUACGCGAAACUCAGGUGCUGCAAUC	120	
Cobalamin	>AP001508.1_5769-5939		56
	ACUUUAAUAGGCUUCUUAGGUGCCUCAUUUGUAGGAGAAUAGGGAAGUUCUGAAACGACGCGGAGCCCGCCACUGUAGUCGAGGAGCUGCUACAAUACCACUGGGAAACUGGGAAGGUGUAGCAUGCGAUGAAUCGGAGCCAGGAGACCUGCCUAAGAAGAUGCGCUGUCA	171	
	>AE017037.1_59439-59627		
	CCUUUCAAAAGGAAAAUAGGUACACGAACAUUUCGUUUCGUGUUUAAAAGGGAAGCUUGGUGAAACUCCAACACGGUCCCGCCACUGUAAAUGCUGAGAUUUCUUUUUGAUACCACUGUGAAAACGGGAAGGUAAAAGAAAUUAUAUGAAGCAUAAGUCAGGAGACCUGCCUGUUUUAACAACACUGAU	189	

4.1.1 The results of Foldalign version 2.0.3

Since the first version of the Foldalign method (i.e. Foldalign version 1.0) is really outdated now, we start with the second version of the Foldalign method (i.e. Foldalign version 2.0.3). The program of Foldalign version 2.0.3 is available online at [http://foldalign.kvl.dk]. The selected parameters from this version to our test are defined according to the manual of this program as follows:

-max_diff <number> this parameter sets the maximum length difference (i.e. delta-heuristic) to <number>. It is essential for memory and time consumption. In this thesis, the focus is on the time behavior against different delta values in testing the speed of the program. A default value of this parameter is infinity.

- global this parameter turns on the global alignments.

- nobranch this parameter turns off the branching-structure.

For this program, these parameters were tested and are displayed in two tables according to the parameter (-nobranch), in addition to the global parameter and different values of delta (i.e. parameter -max_diff) in both tables. Therefore, the observation on the time behavior against these parameters differs. In general, the user time in the table of branch case (table 4.1.1.b) is much higher than non-branch case (table 4.1.1.a) over all datasets of RNA families but in different ratios depending on the sequence lengths. For example, the tRNA dataset is much faster than the other datasets in both tables (i.e. in branch case and non-branch case), where the speed differs by a factor about 2 between -max_diff 15 and 45 for Table/Figure 4.1.1.a, and this is nearly the same factor for Table/Figure 4.1.1.b but only at slower speed. For 5S-rRNA, a factor is different between these two tables, where this factor is about 2.5 between -max_diff 15 and 45 for Table/Figure 4.1.1.a, and this factor is less than 4 for Table/Figure 4.1.1.b. This means the effect is stronger for longer sequences in branching structure. In other cases, Cobalamin has no possibility to aligning at smaller delta values (i.e. parameter -max_diff), such as -max_diff = 15 which was run in our test. Since the global alignment must work on the entire length of sequences, the length difference between two sequences must be less than or equal to delta value. Whereas the length difference between the Cobalamin sequences is 18 nucleotides which is more than the maximum length difference (-max_diff 15), then it was not worked at this delta value. In contrast with tRNA and 5S_rRNA datasets that have length differences less than or equal to delta value (i.e. -max_diff = 15). Now in the following tables and figures (Tables/Figures 4.1.1.a and b), the user time is increased by increasing the delta values over all datasets of RNA families.

Tables 4.1.1.a: This table shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) without branching structures (i.e. turns on the parameter -nobranch).

RNA Families	**Options**			**Run Time**
tRNA	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	TRUE	5.60
	25	TRUE	TRUE	8.27
	35	TRUE	TRUE	10.77
	45	TRUE	TRUE	11.28
5S_rRNA	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	TRUE	29.71
	25	TRUE	TRUE	47.75
	35	TRUE	TRUE	62.70
	45	TRUE	TRUE	75.67
Cobalamin	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	TRUE	not possible
	25	TRUE	TRUE	203.74
	35	TRUE	TRUE	279.44
	45	TRUE	TRUE	349.29

Tables 4.1.1.b: This table shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) with branching structures (i.e. turns off the parameter -nobranch).

RNA Families	Options			Run Time
tRNA	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	FALSE	8.03
	25	TRUE	FALSE	13.20
	35	TRUE	FALSE	16.68
	45	TRUE	FALSE	18.58
5S_rRNA	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	FALSE	60.55
	25	TRUE	FALSE	122.10
	35	TRUE	FALSE	179.88
	45	TRUE	FALSE	233.49
Cobalamin	-max_diff	-global	-nobranch	User Time (sec.)
	15	TRUE	FALSE	not possible
	25	TRUE	FALSE	775.52
	35	TRUE	FALSE	1268.85
	45	TRUE	FALSE	1801.33

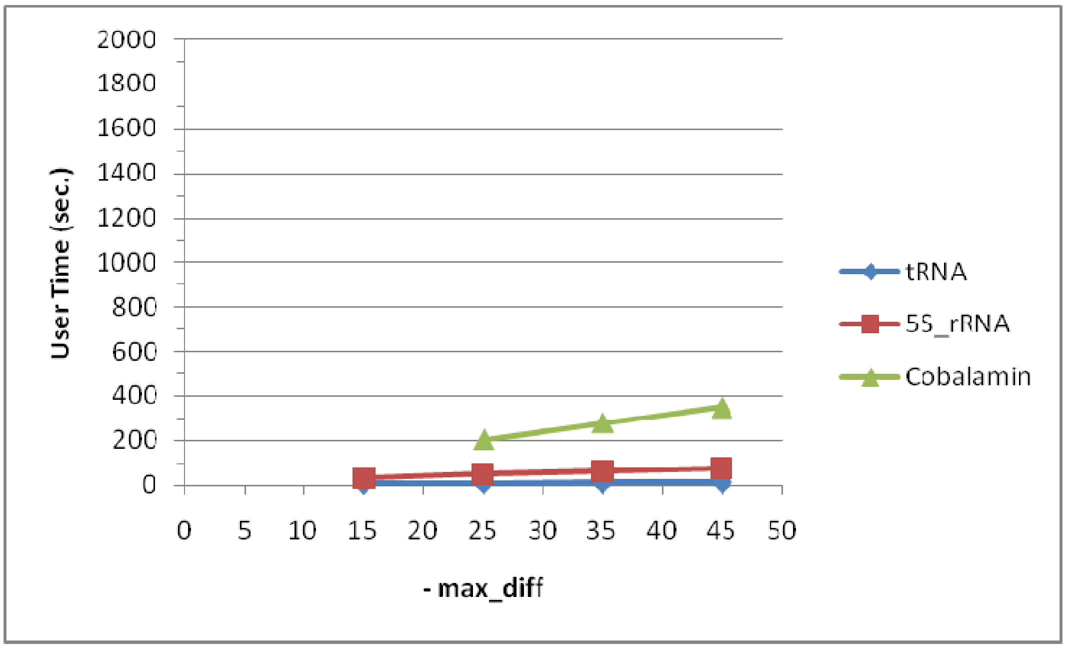


Figure 4.1.1.a: This figure shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) without branching structures (i.e. turns on the parameter -nobranch).

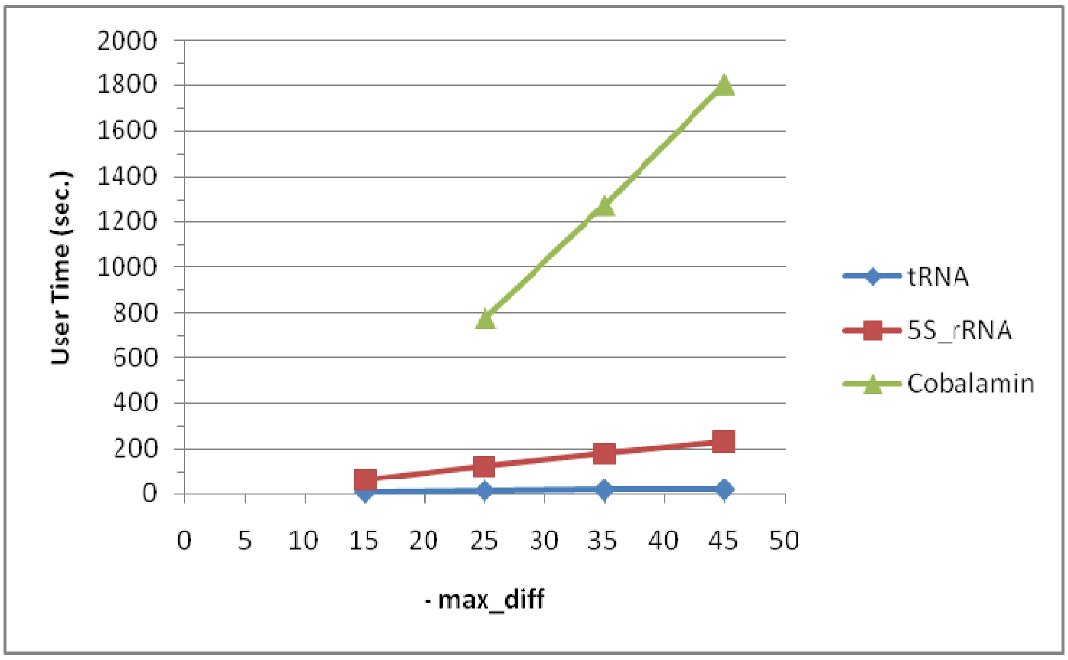


Figure 4.1.1.b: This Figure shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) with branching structures (i.e. turns off the parameter -nobranch).

4.1.2 The results of Foldalign version 2.1.0

This version refers to the last version of the Foldalign method, which is known much faster. The program of Foldalign version 2.1.0 is available online at [http://foldalign.kvl.dk]. The following parameters were selected from this version for our test, the first two parameters are the same as in the previous version of Foldalign (i.e. Foldalign 2.0.3), plus the new parameter which represents the heuristic for this version (Foldalign 2.1.0). These parameters are defined according to the manual of this program as follows:

-max_diff <number> this parameter sets the maximum length difference (i.e. delta-heuristic) to <number>. A default value here for this parameter is 25. In the global alignments where the length difference between the input sequences is greater than 25 nucleotides, -max_diff is set to 1.1 times the length difference.

-global this parameter turns on the global alignments.

-no_pruning this parameter turns off the pruning.

In general, this version is more efficient as compared with the previous version of Foldalign. The observed time behavior was much faster in the pruning case against the different delta values (i.e. parameter -max_diff). In this program, these parameters are displayed into two tables according to the parameter (-no_pruning), in addition to the global parameter and the different delta values (i.e. parameter -max_diff) in both tables. For the pruning case in table 4.1.2.b, the user time was much lower than no-pruning case in table 4.1.2.a over all datasets of RNA families with different ratios. For example, in the Table/Figure 4.1.2.a the parameter -max_diff seems to have no effect on tRNA in absolute time, however the speed differs by a factor of about 2.5 between -max_diff 15 and 45. For Cobalamin this factor is about 4. This means the effect is definitely there already for tRNA, however is stronger for longer sequences. However, in the Table/Figure 4.1.2.b, pruning makes the parameter -max_diff less important, where the speed of tRNA has a very slight difference nearly by a factor about 1.4 between -max_diff 15 and 45. For Cobalamin this factor is more than 2, again the effect is stronger for longer sequences. In this program, Cobalamin dataset has possibility to align over all different delta values which were selected in our test as compared with the previous version. As mentioned above, the delta value (i.e. parameter -max_diff) is set to 1.1 times the length difference during the global alignments where the length difference between the input sequences is more than delta value. (This is due to the improvement performed on the delta-parameter, mentioned before in the theoretical part of this version about the delta heuristic during the global alignment the delta-parameter can be utilized also for restricting the start coordinates of a sub-alignment in the second sequence [HTG07]).

Therefore, this helps to speed up the implementation of this version comparing with the implementation of the previous version.

Tables 4.1.2.a: This table shows the time behavior (in second) at the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) without pruning (i.e. turns on the parameter -no_pruning).

RNA Families	**Options**			**Run Time**
tRNA	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	TRUE	2.14
	25	TRUE	TRUE	3.81
	35	TRUE	TRUE	4.81
	45	TRUE	TRUE	5.39
5S_rRNA	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	TRUE	10.09
	25	TRUE	TRUE	22.14
	35	TRUE	TRUE	32.92
	45	TRUE	TRUE	41.41
Cobalamin	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	TRUE	50.57
	25	TRUE	TRUE	84.55
	35	TRUE	TRUE	148.26
	45	TRUE	TRUE	209.96

Tables 4.1.2.b: This table shows the time behavior (in second) at the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) with pruning (i.e. turns off the parameter -no_pruning).

RNA Families	**Options**			**Run Time**
tRNA	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	FALSE	0.56
	25	TRUE	FALSE	0.71
	35	TRUE	FALSE	0.77
	45	TRUE	FALSE	0.79
5S_rRNA	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	FALSE	1.43
	25	TRUE	FALSE	2.01
	35	TRUE	FALSE	2.45
	45	TRUE	FALSE	2.70
Cobalamin	-max_diff	-global	-no_pruning	User Time (sec.)
	15	TRUE	FALSE	27.77
	25	TRUE	FALSE	39.44
	35	TRUE	FALSE	51.75
	45	TRUE	FALSE	59.63

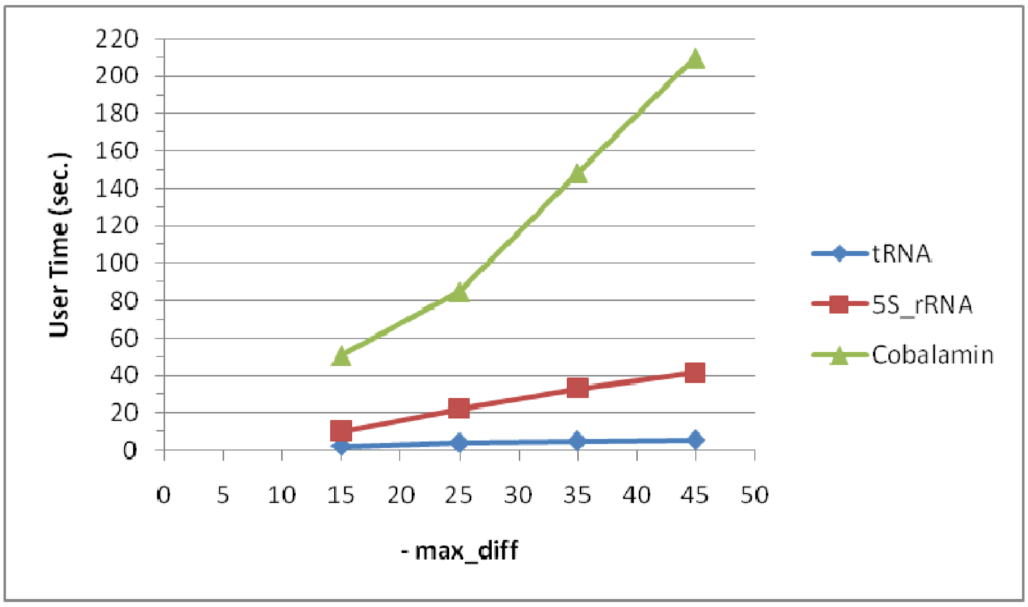


Figure 4.1.2.a: This figure shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) without pruning (i.e. turns on the parameter -no_pruning).

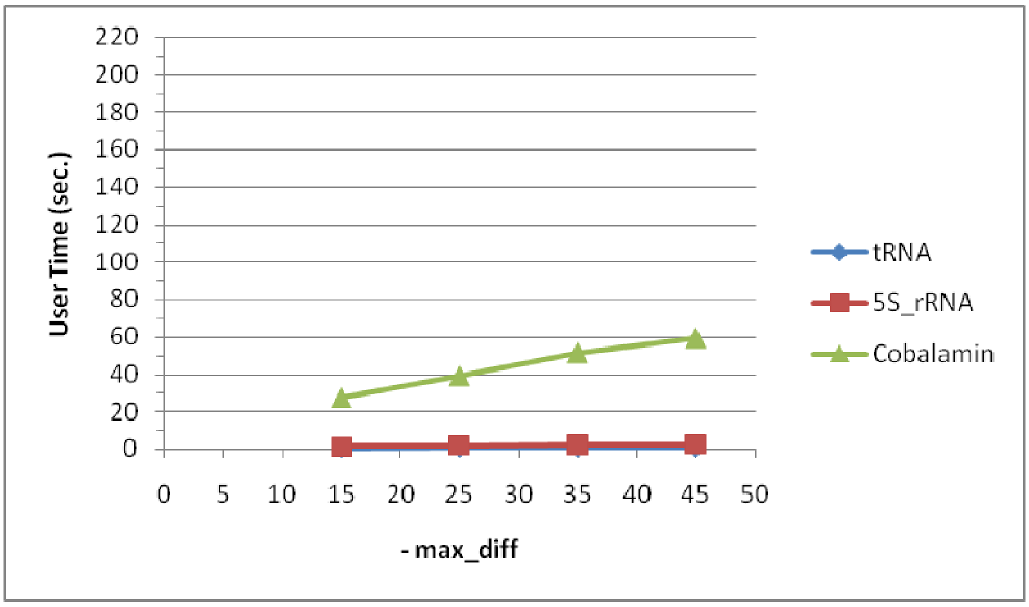


Figure 4.1.2.b: This figure shows the time behavior (in second) of the global alignments (i.e. turns on the parameter -global) against the different delta values (i.e. parameter -max_diff) with pruning (i.e. turns off parameter -no_pruning).

4.1.3 The results of Dynalign

This program includes all restrictions or heuristics which occurred with Dynalign method. The program is available online at http://rna.urmc.rochester.edu/dynalign.html. The selected parameter of this program to run our test is defined according to the manual of this program as follows:

imaxseparation this parameter is a user-specified parameter, M, which was defined as the measure of maximum permissible insertion parameter in the Dynalign method (i.e. correction or recast implementation of parameter M). As mentioned before in the theoretical part of Dynalign method, there is no analytic guidance to select the values of this parameter. Therefore, the imaxseparation parameter is also used in this program to turn on a probabilistic alignment constraint by entering -99 described by Harmanci et al. [HSM07].

Furthermore, there are other parameters which have effect on the speed of the implementation that are not of interested to this thesis. One of these parameters is “singlefold_subopt_percent” which controls a pre-filter step. Dynalign first calls a single sequence secondary structure prediction algorithm. Base pairs for single sequences that result only in relatively high free energy structures are forbidden in the subsequent Dynalign calculation. So this saves calculation time. This is described by Uzilov et al. [UKM06]. Therefore, singlefold_subopt_percent parameter sets the threshold for what constitutes a "high" free energy. By default, it is 30% or greater above the lowest folding free energy change. In our test, we fix this parameter to default value of 30.

In the following table (table 4.1.3), the program is run at different values of parameter M (i.e. parameter imaxseparation) which was used as a maximum insertion parameter as well as to turn on a probabilistic alignment constraint by entering -99. The strength of the latter depends on the similarity of the sequences. Therefore, in our dataset examples of RNA families, we selected sequences which have approximately the same similarity and this is shown in the last column in table 4.1.1. The time behavior was increased over all datasets of RNA families in different ratios by increasing the values of parameter M (i.e. parameter imaxseparation), where M is used as a maximum insertion parameter in Dynalign. The effect of increasing time is definitely clear for tRNA dataset, however is stronger for the longer sequences like Cobalamin. For a parameter imaxseparation which was used to turn on a probabilistic alignment constraint by entering -99, the effect differed according to the different datasets which were used. For tRNA, the user time is less than or nearly has no significant effect on increasing time. In contrast, 5S-rRNA and Cobalamin which have significant increasing in the user time.

In the following figure (Figure 4.1.3), we can see the time behavior (in second) against different values of parameter M (i.e. parameter imaxseparation), where M is used as a maximum insertion parameter in Dynalign. The user time is increased by increasing the values of parameter imaxseparation over all datasets of RNA families in different ratios.

Table 4.1.3: This table shows the time behavior (in second) against the different values of parameter M (i.e. parameter imaxseparation) which was used as a maximum separation parameter for the first heuristic and was also used by entering -99 to turning on a probabilistic alignment constraint for last heuristic in Dynalign method.

RNA Families	Options	Run Time
tRNA	imaxseparation	User Time (sec.)
	-99	1.02
	4	1.23
	6	3.14
	10	10.11
	14	20.18
5S_rRNA	imaxseparation	User Time (sec.)
	-99	88.13
	4	7.76
	6	23.31
	10	85.44
	14	184.43
Cobalamin	imaxseparation	User Time (sec.)
	-99	89.22
	4	26.85
	6	81.43
	10	331.82
	14	779.89

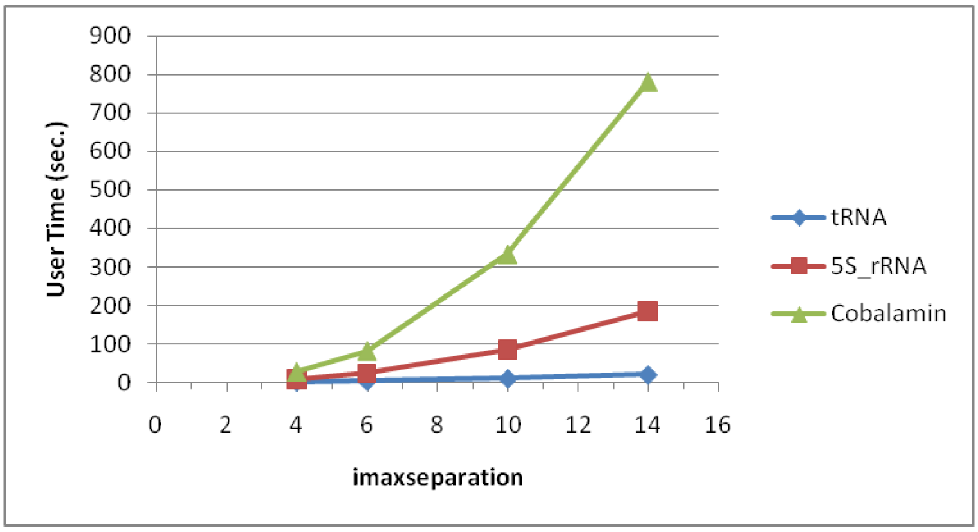


Figure 4.1.3: This figure shows the time behavior (in second) against the different values of parameter M (i.e. parameter imaxseparation) which was used only as a maximum separation parameter for the first heuristic in Dynalign method.

4.1.4 The results of LocARNA

Since LocARNA is PMcomp-based method it is more efficient. Therefore, we performed our test on the LocARNA program which is available online at [http://www.bioinf.uni-freiburg.de/Software/LocARNA/]. The parameters selected to run our test for illustrating the time behavior in this program, are defined according to the manual of this program as follows:

--min-prob this parameter sets the minimal probability (i.e. cutoff-probability). A default value for this parameter is 0.0005.

--max-diff-am this parameter sets the maximal difference for sizes of matched arcs. A default value is -1 which represents “turn-off” option.

--max-diff-match this parameter sets the maximal difference for sizes of matched structural positions. A default value is also -1 which represents “turn-off” parameter.

In the following Table 4.1.4.a, shows the time behavior against different cut-off probability values (i.e. parameter --min-prob) and turn off of the two parameters (--max-diff-am, --max-diff-match) by setting them -1, is increased by decreasing the cut-off probability values which controls the per-filtering the number of base pairs as mentioned before in theoretical part of LocARNA. However, this increasing of the user time differs depending on the dataset lengths. For example, the parameter --min-prob seems to have almost no effect on tRNA in absolute time; however the speed differs by a factor of about 2 between --min-prob 0.05 and 0.00005. For Cobalamin this factor is more than 9.5, where the parameter --min-prob at smaller values has a significant effect, such as at --min-prob 0.0005 and 0.00005. This means the effect is stronger for longer sequences, especially for the smaller values of --min-prob.
Therefore, this can be expected from the time behavior corresponding to the strength of the cut-off probability heuristic (i.e. parameter --min-prob) that has a significant influence on the longer sequences like Cobalamin or on the infinite sequence lengths. Furthermore, this cut-off probability heuristic (i.e. parameter --min-prob) has much larger effect than Δ-parameters (i.e. the parameters --max-diff-am and --max-diff-match) on the time behavior over all our datasets, especially for the longer sequences like Cobalamin.

Figure 4.1.4.a shows the time behavior (in second) against the different values of parameter --min-prob and with turn off the other parameters (--max-diff-am, --max-diff-match), which is increased by decreasing the parameter --min-prob over all datasets but in different ratios.

Table 4.1.4.a: This table shows the time behavior (in second) against the different values of cut-off probability (i.e. parameter --min-prob) and turn off each of the parameters (--max-diff-am, --max-diff-match) by setting them to -1.

RNA Families	Options			Run Time
tRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.05	-1	-1	0.17
	0.005	-1	-1	0.20
	0.0005	-1	-1	0.25
	0.00005	-1	-1	0.34
5S_rRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.05	-1	-1	0.35
	0.005	-1	-1	0.43
	0.0005	-1	-1	0.55
	0.00005	-1	-1	0.85
Cobalamin	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.05	-1	-1	1.10
	0.005	-1	-1	1.78
	0.0005	-1	-1	5.47
	0.00005	-1	-1	10.54

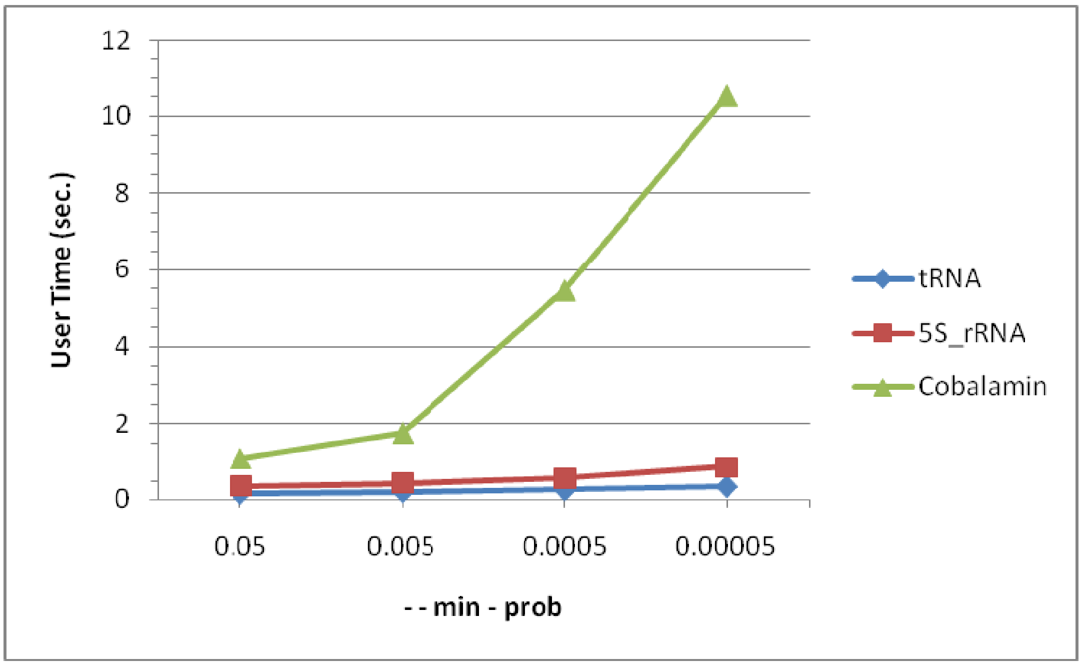


Figure 4.1.4.a: This figure shows the time behavior (in second) against the different values of cut-off probability (i.e. parameter --min-prob) and turn of each of the parameters (--max-diff-am, --max-diff-match) by setting them to -1.

In Table 4.1.4.b.1),shows the time behavior against different values of Δ_{match}-parameter (i.e. parameter --max-diff-match) and turn off Δ_{am}-parameter (i.e. parameter --max-diff-am) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005), which is increased by increasing the values of parameter --max-diff-match. However, this increasing of the user time differs depending on the dataset lengths. For example, the parameter --max-diff-match seems to have almost no effect on tRNA in absolute time; however the speed differs by a factor of about 1.3 between --max-diff-match 15 and 60. For Cobalamin this factor is more than 3.2. This means the effect is also stronger for longer sequences. Figure 4.1.4.b.1) shows this case.

Analogously, table 4.1.4.b.2) shows the behavior time against different values of Δ_{am}-parameter (i.e. parameter --max-diff-am) and with turn off Δ_{match}-parameter (i.e. parameter --max-diff-match) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005), which is increased by increasing the values of parameter --max-diff-am. However, this increasing of the user time also differs depending on the dataset lengths. For example, the parameter --max-diff-am seems to have almost no effect on tRNA in absolute time; however the speed differs by a factor of less than 1.2 between --max-diff-am 15 and 60. For Cobalamin this factor is more than 2. Again, the effect is stronger for longer sequences. Figure 4.1.4.b.2) shows this case.

Table 4.1.4.c, shows the test results of the time behavior against different values of parameter --max-diff-am and by varying the values of parameter --max-diff-match to two times the values of parameter --max-diff-am in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005). The time behavior is again increased by increasing the values of parameter --max-diff-am but such increasing differs on increasing in table 4.1.4.b.2) in different ratios. For example, the parameters --max-diff-am and --max-diff-match seem to have almost no effect on tRNA in absolute time; however the speed differs by a factor of more than 1.2 between --max-diff-am 15 and 60, and --max-diff-match 30 and 120. For Cobalamin this factor is about 3, again the effect is stronger for longer sequences. Figure 4.1.4.c shows this case.

It can be seen that the strength of the Δ_{am}-parameter (i.e. parameter --max-diff-am) has larger effect than Δ_{match}-parameter (i.e. parameter --max-diff-match) on the time behavior.

Table 4.1.4.b.(1): This table shows the time behavior (in second) against the different values of Δ_{match}-parameter (i.e. parameter --max-diff-match) and turn off Δ_{am}-parameter (i.e. parameter --max-diff-am) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005).

RNA Families	Options			Run Time
tRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	-1	15	0.19
	0.0005	-1	30	0.22
	0.0005	-1	60	0.25
5S_rRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	-1	15	0.33
	0.0005	-1	30	0.38
	0.0005	-1	60	0.48
Cobalamin	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	-1	15	1.17
	0.0005	-1	30	2.03
	0.0005	-1	60	3.77

Table 4.1.4.b.(2): This table shows the time behavior (in second) against the different values of Δ_{am}-parameter (i.e. parameter --max-diff-am) and turn off Δ_{match}-parameter (i.e. parameter --max-diff-match) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005).

RNA Families	**Options**			**Run Time**
tRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	-1	0.21
	0.0005	30	-1	0.22
	0.0005	60	-1	0.24
5S_rRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	-1	0.36
	0.0005	30	-1	0.40
	0.0005	60	-1	0.46
Cobalamin	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	-1	2.17
	0.0005	30	-1	3.14
	0.0005	60	-1	4.45

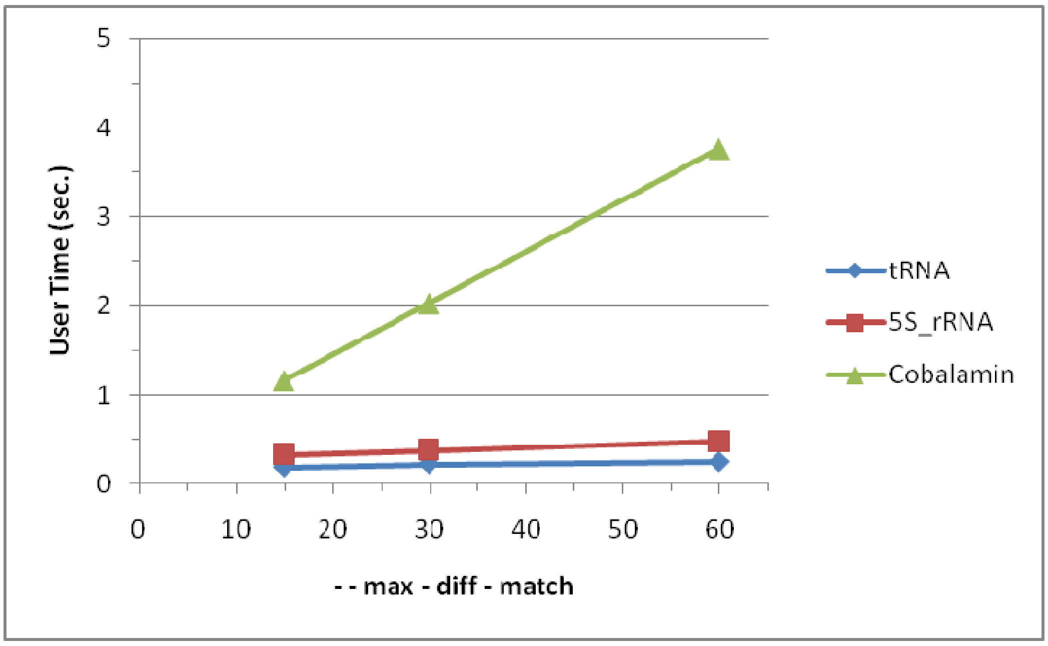


Figure 4.1.4.b.(1): This figure shows the time behavior (in second) against the different values Δ_{match}-parameter (i.e. parameter --max-diff-match) and turn off Δ_{am}-parameter (i.e. parameter --max-diff-am) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005).

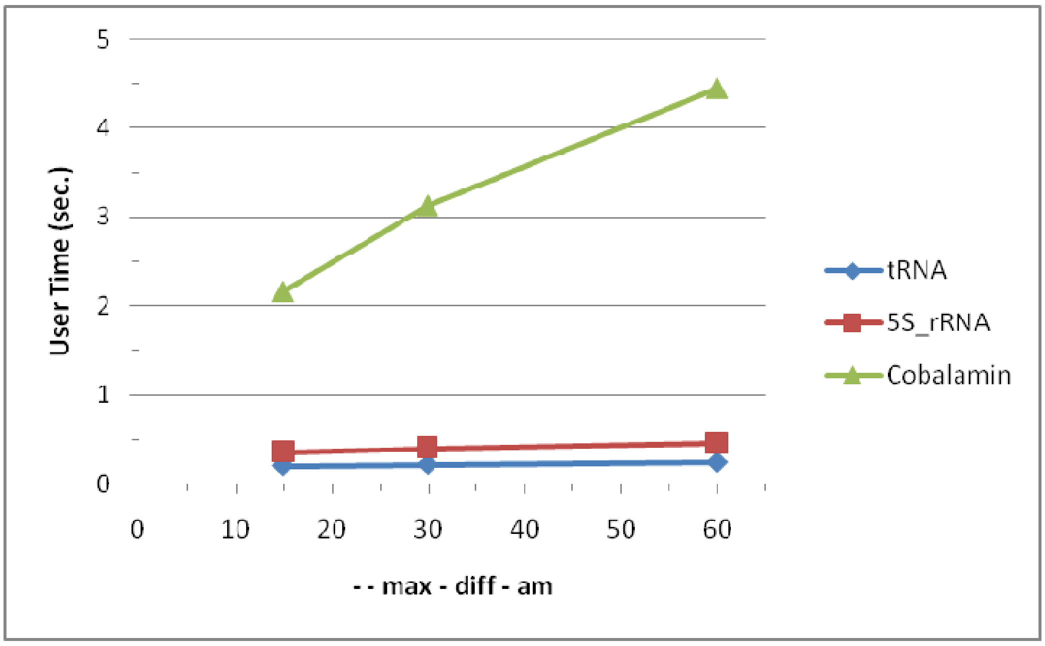


Figure 4.1.4.b.(2): This figure shows the time behavior (in second) against the different values of Δ_{am}-parameter (i.e. parameter --max-diff-am) and turn off Δ_{match}-parameter (i.e. parameter --max-diff-match) by setting it to -1 in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005).

Table 4.1.4.c: This table shows the time behavior against different values of Δ_{am}-parameter (i.e. parameter --max-diff-am) and with varying the values of Δ_{match}-parameter (i.e. parameter --max-diff-match) to two times the values of Δ_{am}-parameter (i.e. parameter --max-diff-am) in addition to fixing the cut-off probability value to default value (i.e. parameter --min-prob = 0.0005).

RNA Families	Options			Run Time
tRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	30	0.20
	0.0005	30	60	0.22
	0.0005	60	120	0.25
5S_rRNA	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	30	0.33
	0.0005	30	60	0.40
	0.0005	60	120	0.46
Cobalamin	--min-prob	--max-diff-am	--max-diff-match	User Time (sec.)
	0.0005	15	30	1.43
	0.0005	30	60	2.66
	0.0005	60	120	4.26

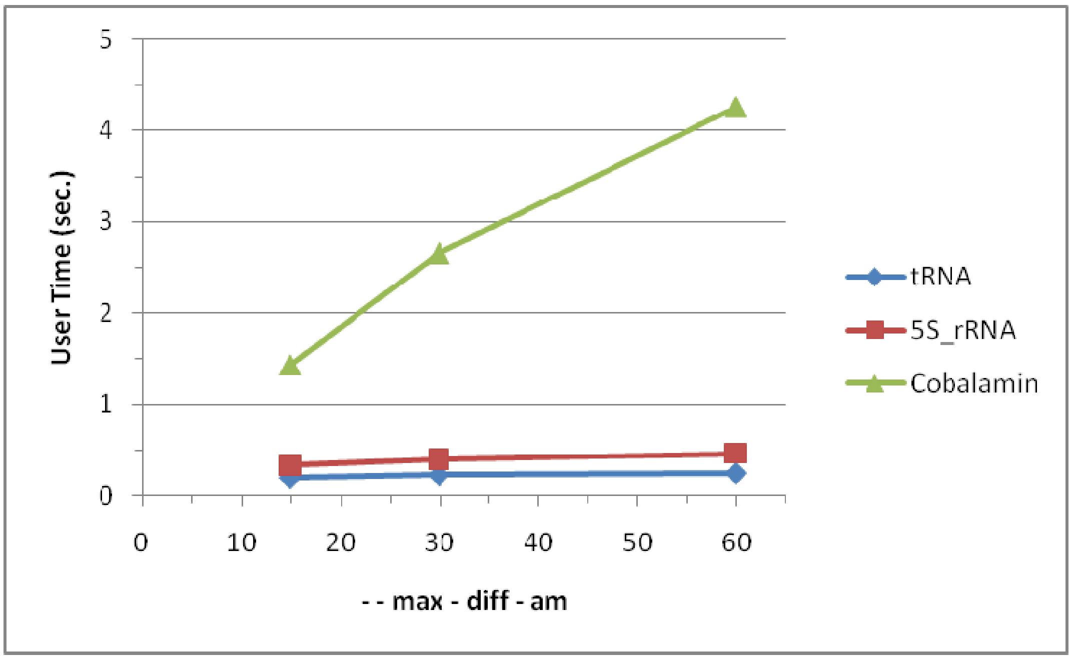


Figure 4.1.4.c: This figure shows the time behavior against different values of Δ_{am}-parameter (i.e. parameter --max-diff-am) where the values of Δ_{match}-parameter (i.e. parameter --max-diff-match) were varied to two times the values of Δ_{am}-parameter (i.e. parameter --max-diff-am) in addition to fixing the cut-off probability value to default value (i.e. o parameter --min-prob = 0.0005).

4.2 Discussion of Results

The dataset examples that were tested by all programs, they are three different dataset types of RNA families which have different lengths and approximately the same similarity. The different sequence lengths of these examples give an intuition about the speed influences which interested us for comparing between the programs. Different time behaviors are observed in each example of dataset against the options of each program. In addition to set the alignment type to global alignment which is used in all programs, alignment type corresponds in the Sankoff algorithm. Some methods like Foldalign 2.0, 2.1 versions and LocARNA implement local alignment but can also be applied for global alignment.

In general, each parameter has a significant influence on the time depending on program type and sequence lengths.

Since the global alignment is studied here, the λ-heuristic in the Foldalign 2.0, 2.1 programs is not used. We analyzed only the δ-heuristic (the length difference between two sequences which being aligned). However, we tested how fast the stem-loop structure as compared with branching structure by turning on the option -nobranch in the Foldalign 2.0. Moreover, the pruning-heuristic in Foldalign 2.1 which was much faster as compared with no pruning.

As mentioned before in the theoretical part of the Foldalign method the recursions of 2.0 and 2.1 versions are pretty much the same except for the few improvements or simplifications in the energy model. The main difference between the 2.0 and 2.1 versions is the use of the pruning heuristic which throws away all alignments with a score below a cut off.

It is assumed at some examples in Foldalign 2.1 that we could not get an optimal solution, but when these examples are applied, Foldalign did solve them. As it seems, this is explained by recalculation of stems. Without such recalculation the example would not work. Therefore, we should expect in most cases to get an optimal solution, but it is not guaranteed.

The parameters of LocARNA program have great influence on the run time, where the two parameters (--max-diff-match and --max-diff-am) can be compared with the parameter M-heuristic and δ-heuristic respectively. In addition it uses the cutoff probability-heuristic, which filters the base pairs.

As seen from our results, LocARNA represents the fastest method even when turning off its stronger parameters, and that because it considers as a simplified method of the Sankoff algorithm. In contrast with Dynalign which is used significant heuristics but it is still slow and that because it has the full energy model. Foldalign 2.1 is faster than Dynalign due to its special scoring scheme, but it still slower than LocARNA even using the stronger heuristics.

Since the used dataset of RNA families are only three examples, the accuracy estimation of the alignments cannot calculated because it is required the significant numbers of sequences.

Chapter Five

Conclusion

5.1 Conclusion

After looking into the original form of the Sankoff algorithm for simultaneous Folding and Alignment of RNA sequences, as well as the different methods are implemented to this algorithm by using diverse restrictions on either folding or alignment parts. These methods are tools used for pairwise structural alignment of RNA sequences. In this thesis, we have concentrated on the heuristics of these methods that make the Sankoff algorithm applicable in practice.

The comparison of these methods to the original form “Sankoff Algorithm” is described as follows: Dynalign is the method closest to the Sankoff algorithm. It has several heuristics but in this thesis we are interested in the most significant two heuristics which are parameter M and probabilistic alignment constraint. Foldalign is the first implementation of the simultaneous Folding and Alignment of RNA sequences. It represents a very special method based on energy model that includes a scoring system of one matrix and different states; the most significant heuristics that are used for the Foldalign method, are λ and δ-restrictions with stem-loop for the Foldalign 1.0, these restrictions with branching-loop for Foldalign 2.0 and the pruning for Foldalign 2.1. PMcomp and LocARNA are a simplification of the Sankoff algorithm, which are based on the probabilities of base pairs of RNA sequences as a structure model. The strongest heuristics used in LocARNA, are the cutoff-probability, Δ_{am} and Δ_{match}- parameters for arc matching and matched structural positions respectively.

The heuristics of these methods are considered strong heuristics which have significant influence on the speed. The comparisons among the heuristics can give such ideas to combine between the heuristics of these methods in such a way that implement either a new method or the improved versions of the current methods on the basis of the original form "Sankoff Algorithm".

The traditional M parameter in Dynalign method is comparable to Δ_{match}- parameter in LocARNA, as shown here:

Dynalign: $|i - k| \leq M$,

LocARNA: $|i - k| \leq \Delta_{match}$ and $|j - l| \leq \Delta_{match}$,

but with some differences. For the Dynalign method, the constraint of M parameter has to be satisfied for all alignment edges (i, k). In contrast, in the LocARNA method, the constraint of Δ_{match}-parameter has to be satisfied only for all structural alignment edges.

As well as, for the δ-restriction in Foldalign method is comparable to Δ_{am}- parameter in LocARNA, as shown here:

Foldalign: $|(j - i) - (l - k)| \leq \delta$,

LocARNA: $|(j - i) - (l - k)| \leq \Delta_{am}$,

but also for the same reason there is a difference. For the Foldalign method, the δ-restriction has to be satisfied for all sub-alignments. In contrast, in the LocARNA method, the constraint of Δ_{am}-parameter has to be satisfied only for all structural sub-alignments.

5.2 Future Work

There are many possible ways for extending Sankoff-style methods further. One can find new methods or improved versions of the current methods on the basis of the original form “Sankoff Algorithm” for simultaneous Folding and Alignment of RNA sequences, by combining some heuristics of the current methods, such that it gives the good accurate alignments as well as significantly fast and requiring low memory.

If we use the idea of probabilistic alignment constraint of Dynalign in the LocARNA method, we expect that the new generated improved version of LocARNA will be much faster and taking into consideration the preservation of the accurate alignments. The same positive effect is expected, if we use the pruning idea of the Foldalign 2.1 in the LocARNA method.

On the other side, if we use the idea of significant base pairs of LocARNA in the Dynalign or Foldalign (especially for Foldalign 2.1), we expect that the new generated versions for Dynalign or Foldalign 2.1 will improve to be more efficient methods much faster and still have accurate alignments.

Bibliography

[BW04] Rolf Backofen and Sebastian Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, **2** no. 4 pp. 681-698, 2004.

[CB2000] P. Clote and R. Backofen. Computational Molecular Biology: An Introduction, pages 1–20. John Wiley and Sons Ltd, 2000.

[CK91] Chiu DK, Kolodziejczak T: Inferring consensus structure from nucleic acid sequences. *Comput Appl Biosci* 1991, 7:347-352.

[DE06] Dowell R, Eddy S (2006) Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. BMC Bioinformatics 7: 400.

[DEK+M99] Durbin R, Eddy SR, Krogh A, Mitchison G: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* Cambridge, UK: Cambridge University Press; 1999.

[G99] Gotoh O: Multiple sequence alignment: algorithms and applications. *Adv Biophys* 1999, 36:159-206.

[GG04] Gardner, P.P. and Giegerich, R. 2004. A comprehensive comparison of comparative RNA structure prediction approaches. BMC Bioinformatics 5**:** 140.

[GHB+S97] Gorodkin J, Heyer L, Brunak S, Stormo G: Displaying the information contents of structural RNA alignments. *Comput Appl Biosci* 1997, 13:583-586.

[GHS97a] Gorodkin J, Heyer L J, Stormo G D. Finding common sequence and structure motifs in a set of RNA sequences. Nucleic Acids Res., 1997, 25(18): 3724-3732.

[GHS97b] Gorodkin J., Heyer,L.J. and Stormo,G.D. (1997) Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res.*, 25, 3724–3732.

[GPH+PS92] Gutell RR, Power A, Hertz GZ, Putz EJ, Stormo GD: Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *Nucleic Acids Res* 1992, 20:5785-5795.

[H05] Holmes I. Accelerated probabilistic inference of RNA structure evolution. BMC Bioinformatics. 2005 Mar 24; 6:73.

[HBS04] Hofacker, I.L., Bernhart, S.H., and Stadler, P.F., Alignment of RNA base pairing probability matrices, *Bioinformatics*, 20(14):2222–2227, 2004.

[HFB+S94] Hofacker IL, Fontana W, Bonhoeffer S, Stadler PF: Fast folding and comparison of RNA secondary structures. *Monatshefte fur Chemie* 1994, 125:167-188.

[HFS02] Hofacker I, Fekete M, Stadler P: Secondary structure prediction for aligned RNA sequences. *Journal of Molecular Biology* 2002, 319(5):1059-1066.

[HLS+G05] Havgaard J, Lyngsø R, Stormo G, Gorodkin J (2005) Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. Bioinformatics 21: 1815–1824.

[HSM07] Harmanci AO, Sharma G, Mathews DH: Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics* 2007, 8:130.

[HTG+K03] Hchsmann M, Tller T, Giegerich R, Kurtz S: Local similarity of RNA secondary structures. *Proc of the IEEE Bioinformatics Conference* 2003:159-168.

[HTG07] Havgaard JH, Torarinsson E, Gorodkin J: Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput Biol* 2007, 3:1896-1908.

[JLM+Z02] Jiang T, Lin G, Ma B, Zhang K: A general edit distance between RNA structures. *Journal of Computational Biology* 2002, 9(2):371-388.

[KE03] Klein R, Eddy S: RSEARCH: Finding homologs of single structured RNA sequences. *BMC Bioinformatics* 2003, 4:44.

[KH03] Knudsen B, Hein J: Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research* 2003, 31(13):3423-3428.

[M90] McCaskill JS: The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure. *Biopolymers* 1990, 29:1105-1119.

[MSZ+T99] Mathews, D. H., Sabina, J., Zuker, M. & Turner, D. H. (1999). Expanded sequence dependence of thermodynamic parameters provides improved prediction of RNA secondary structure. J. Mol. Biol. 288, 911-940.

[MT02] Mathews DH, Turner DH. Dynalign: An Algorithm for Finding the Secondary Structure Common to two RNA Sequences. *J Mol Biol.* 2002; 317: 191–203. doi: 10.1006/jmbi.2001.5351.

[MT78] Michael S. Waterman and Temple F. Smith. RNA secondary structure: a complete mathematical analysis. Math. Biosci., 42(3–4):257–266, 1978.

[N80] Ruth Nussinov and Ann B. Jacobson. Fast algorithm for predicting the secondary structure of single stranded RNA. Proc. Natl. Acad. Sci. (USA), 77(11):6309–6313, 1980.

[NHH00] Notredame C, Higgins D, Heringa J: T-COFFEE: A novel method for fast and accurate multiple alignment. *Journal of Molecular Biology* 2000, 302:205-217.

[NW70] Needleman, S. B. & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48, 443-453.

[PS1] Protocol S1: Supplementary Material for “Fast pairwise local structural RNA alignments by pruning of the dynamical programming matrix” Jakob H. Havgaard, Elfar Torarinsson, Jan Gorodkin.

[PTW99] Pace, N.R., Thomas, B.C., and Woese, C.R. 1999. Probing RNA structure, function, and history by comparative analysis. In The RNA world, 2nd ed. (eds. R.F. Gesteland et al.), pp. 113–141. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

[S85] Sankoff D, Simultaneous solution of the RNA folding, alignment, and proto-sequence problems. *SIAM J Appl Math* 1985, 45:810-825.

[S88] Shapiro BA: An algorithm for comparing multiple RNA secondary structures. *Comput Appl Biosci* 1988, 4:387-393.

[SB03] Siebert S, Backofen R: MARNA A server for multiple alignment of RNAs. In *Proceedings of the German Conference on Bioinformatics* 2003:135-140.

[SW81] Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, 147, 195–197.

[SZ90] Shapiro B, Zhang K: Comparing multiple RNA secondary structures using tree comparisons. *CABIOS* 1990, 6:309-318.

[T79] Tai K: The tree-to-tree correction problem. *Journal of the ACM* 1979, 26:422-433.

[THG94] Thompson J, Higgins D, Gibson T: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research* 1994, 22:4673-4680.

[UKM06] Uzilov A, Keegan J, Mathews D: Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics* 2006, 7:173.

[WP08] from Wikipedia: http://en.wikipedia.org/wiki/RNA#cite_note-The_Cell-19. URL

[WRH+SB07] Will S, Reiche K, Hofacker IL, Stadler PF, Backofen R: Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput Biol* 2007, 3(4):400.

[WZ01] Wang Z, Zhang K: Alignment between two RNA structures. *Lecture Notes in Computer Science* 2001, 2136:690-703.

[XSB+KSJ98] Xia, T., SantaLucia, J., Jr, Burkard, M. E., Kierzek, R., Schroeder, S. J. & Jiao, X., et al. (1998). Parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick pairs. Biochemistry, 37, 14719-14735.

[ZMT99] Zuker,M., Mathews,D.H. and Turner,D.H. (1999) Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide. In Clark,J.B.B. (ed.), *RNA Biochemistry and Biotechnology*. NATO ASI Series, Kluwer Academic Publishers, Dordrecht, NL, pp. 11–43.

[ZS81] Zuker M, Stiegler P: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research* 1981, 9:133-148.

[ZSh89] Zhang K, Shasha D: Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing* 1989, 18(6):1245-1262.

List of Figures

List of Tables