Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik

Pairwise Comparison of RNA Secondary Structures via Exact Pattern Matches

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Bioinformatiker

eingereicht von Steffen Heyne
geboren am 21. März 1980 in Dresden

Gutachter:	Prof. Dr. Rolf Backofen
	Prof. Dr. Stefan Schuster
Betreuer:	Dr. Sven Siebert
	Dr. Sebastian Will
	Prof. Dr. Rolf Backofen

Jena, 07. November 2007

Dank im Besonderen an

Rolf Backofen, Stefan Schuster, Sven Siebert, Sebastian Will, Martin Mann.
Annkatrin Hammann, Anita Maercker, Janice Kielbassa, Dorothée Marth,
Ralf-Peter Weiß, Sara Sundermann, Axel Schäfer, Kris Adler, Can Senguel.
Margit Heyne, Konrad Heyne, Andrea Langwald, Rico Langwald.

Zusammenfassung

Ribonukleinsäuren (RNAs) sind in lebenden Organismen an vielen wichtigen zellulären Prozessen beteiligt. Lange Zeit waren für RNAs nur grundlegende Zellfunktionen wie die der mRNAs zur Informationsübertragung von DNA zu Proteinen bekannt. Die Entdeckung von Ribozymen sowie im besonderen Forschungen aus jüngster Zeit brachten Funktionen für RNAs ans Tageslicht, die sonst nur mit Proteinen assoziiert wurden. So können zum Beispiel RNAs die nicht für Proteine codieren, die Aktivierung oder Unterdrückung von Genen sowie das Niveau der Genexpression beeinflussen. Diese und eine vielzahl weiterer Entdeckungen haben dafür gesorgt, dass RNAs sich wieder im Blickpunkt aktueller Forschung befinden.

Die Funktion eines RNA Moleküls wird bestimmt durch seine dreidimensionale Struktur. Darüberhinaus sind spezifische Funktionen mit speziellen Teilstrukturen oder so genannten *Motiven* innerhalb des RNA Moleküls verbunden. Beispiele für solche Motive sind SECIS Elemente oder IRES Sequenzen. Das Charakteristische dieser Motive besteht in der Kombination von sequentiellen und strukturellen Merkmalen. Für die Identifizierung von neuen als auch von bekannten Motiven sind deshalb Vergleichsmethoden für RNAs notwendig, die auf Sequenz *und* Struktur basieren. Verschiedene methodische Ansätze existieren dafür, aber viele von diesen arbeiten nicht auf der Basis von Motiven und behandeln identische Teilstrukturen nicht als eine Einheit. Desweiteren sind diese Methoden oft nicht schnell genug für große RNAs.

In der vorliegenden Diplomarbeit werden zwei Methoden für den paarweisen Vergleich von RNA Sekundärstrukturen auf der Basis von exakten Teilstrukturen vorgestellt. Diese Teilstrukturen werden „Exact Pattern Matches“ genannt. Für den Vergleich werden dazu in einem ersten Schritt eine Menge von sich überlappenden und kreuzenden Teilstrukturen der beiden vorgegebenen genesteten RNA Sekundärstrukturen bestimmt. Dazu wird der Ansatz für gemeinsame Teilstrukturen von Siebert/Backofen genutzt. Die erste neu entwickelte Methode bestimmt nun die größte globale Teilmenge von sich nicht kreuzenden und nicht überlappenden Teilstrukturen von zwei gegebenen RNAs. Dieses Problem wird als Longest Common Subsequence of Exact RNA Patterns bezeichnet und steht in Verbindung zum bereits bekannten LAPCS Problem. Zur Lösung wird ein dynamischer Programmieralgorithmus entwickelt, welcher dieses Problem in $O(n^2m^2)$ Zeit und $O(nm)$ Speicherplatz löst. Die zweite entwickelte Methode findet lokale Cluster von exakten Teilstrukturen. Ein Cluster ist eine Anordnung von sich nicht kreuzenden und nicht überlappenden Teilstrukturen mit einer zusätzlichen Distanzbedingung zwischen einzelnen Teilstrukturen des Clusters. Die entwickelte „Clustering Strategie“ für die Bestimmung von Clustern ist schnell und flexibel für verschiedene analytische Probleme. Beide Methoden wurden mit zwei Hepatitis C Virus IRES RNAs und zwei 16S ribosomalen RNAs getestet. Die Ergebnisse zeigen, dass beide Methoden klare Ähnlichkeiten zwischen zwei RNA Sekundästrukturen auf schnelle Weise finden können.

Abstract

In living organisms, ribonucleic acids (RNA) are involved in important cellular processes. For a long time, only basal functions had been known for RNAs like messenger RNAs as the information carrier between DNA and proteins. The discovery of ribozymes and especially recent findings revealed functions for RNAs formerly assumed only for proteins. For example, RNAs that do not code for proteins can influence processes like the activation and repression of genes as well as the regulation of gene expression levels. These discoveries set RNAs in the focus of current research.

The function of an RNA molecule is determined by its three-dimensional structure. Moreover, specific functions are associated to specific substructures or *motifs* within an RNA molecule. Examples are SECIS elements, iron-responsive elements and IRES sites. The key feature for such motifs is usually a combination of sequential and structural properties. Sequence-structure based comparison methods are necessary to identify already known motifs as well as putative new motifs. Different approaches exist which deal with a sequence-structure comparison of RNA molecules, but most of them are not motif-based and they do not treat identical substructures as hole unit. Further, they are often not fast enough for large RNAs.

In this thesis we have developed two pairwise comparison methods on the basis of exact matching substructures, called *exact pattern matches*. In a first step, a set of overlapping and crossing substructures for two nested RNA secondary structures is found with the approach of pairwise common substructures from Siebert/Backofen. Our first method deals with the task to identify the best global subset of Non-Crossing exact pattern matches for two given RNAs. In relation to the LAPCS problem, we call this problem the Longest Common Subsequence of Exact RNA Patterns. The developed dynamic programming algorithm needs $O(n^2m^2)$ time and $O(nm)$ space. Our second approach detects (local) *clusters* of exact pattern matches. A cluster is a Non-Crossing arrangement of exact pattern matches with a distance constraint between the substructures included in a cluster. The developed clustering strategy to find clusters is fast and flexible enough for different analytical problems. We have tested both methods with two Hepatitis C virus RNAs and two 16s ribosomal RNAs. The results show that both methods are able to identify significant similarities between two RNA secondary structures in a fast way.

Contents

Chapter 1

Introduction

1.1 Motivation

Ribonucleic acid (RNA) is an important biopolymer which attracts more and more researchers' attention since recent discoveries have revealed its wide range of functions in living organisms. For a long time it has been assumed that only proteins can catalyze biochemical reactions. With the discovery of the first ribozyme in 1980, a catalytic active RNA that facilitates its own splicing, the view on RNA as a simple carrier between DNA and proteins started to crumble. The "breakthrough" in RNA research has then been proclaimed in 2002 by the readers of the Science journal for the numerous newly discovered functions for small RNAs [Cou02].

The group of functional RNAs which are not translated into proteins are often summarized as non-coding RNAs (ncRNA) and they are involved in different cellular processes. For example, the translation machinery (ribosome) is built to a large extent of ribosomal RNAs (rRNA). Transfer RNAs (tRNA) here realize the translation of codons into amino acids. In the spliceosome RNAs guarantee the exact cleavage of introns.

Recent findings for RNAs add especially functions formerly assigned to proteins. Examples are the influence of RNAs in the activation or repression of genes as well as their potential to control the expression levels of genes. There are even indications that RNAs take part in the cell development [Cou02].

Understanding this broad range of different functions, an analysis of the concrete three-dimensional structure of RNA molecules is necessary. Comparison methods revealed that specific functions are associated to specific motifs in the RNA structure. For example, the presence of a SECIS element in the 3' untranslated (UTR) mRNA region of mammalians facilitates the integration of the 21st amino acid selenocysteine in the peptide chain of proteins. The key feature within the SECIS element is the combination of sequential and structural properties that form the motif [KCN$^+$03, WSPB97, FCDK01].

The task of bioinformatical approaches is to provide automatic methods for RNA molecules which identify such motifs as well as identify putative new motifs. For example, see the conserved structural pattern in figure 1.1. The indicated substructure could represent a necessary part of the SECIS motif in this organism. Furthermore, one can think about motifs which consist of several such substructures [SB07].

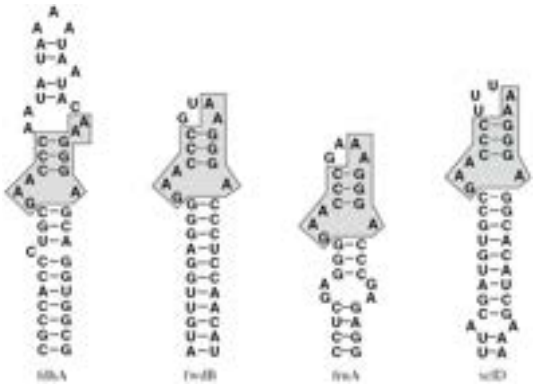

Figure 1.1: Putative SECIS elements in non-coding regions of *Methanococcus jannaschii* according to [WSPB97]. The indicated substructure is found by the `MCS` algorithm and represents an *exact pattern match*. Figure taken from [Sie06].

The computational problem is the high complexity of the structural interactions in an RNA molecule. Nevertheless, most of the known motifs can be described reasonable on the basis of nested secondary structures. This makes the motif-finding problem tractable for algorithms and there exist several methods for the comparison of RNA molecules. However, most methods treat at least a base-pair as a whole unit and common substructures are not part of their "alphabet" [BMR95, JLMZ02]. This could be a drawback for the discovery of motifs comprising different substructures. Moreover, existing methods are often not fast enough for large RNAs. This motivates the development of motif-based and fast approaches for the comparison of two RNA molecules.

In this thesis we deal with this task and develop two fast pairwise sequence-structure comparison methods based on identical substructures between two given RNA secondary structures. The first method aims on global similarities whereas the second method tries to find clusters of common substructures.

1.2 Related Work

Several methods exist which address the problem of sequence-structure comparison for RNA molecules. Based on the given structural information, one can distinguish three main approaches. If no structural information is given at all, Sankoff-like methods try to predict the secondary structure along with a sequence alignment [San85]. Another group of methods try to derive a secondary structure for a given RNA sequence from another RNA with its primary and secondary structure [BMR95, LRV98]. A third group of methods compare two RNAs given with their primary and secondary structures. This type of RNA sequence-structure comparison best represents the approach treated in this thesis. Here one can further distinguish global and local methods.

Global Comparison Methods

Due to the complex base pairings, RNA structures are in general crossing structures. However, this class of RNA structures is algorithmically hardly tractable, because the

comparison problem becomes easily NP-hard. In the case of pairwise comparison this is often denoted as Problem(Type1,Type2), whereby the two types describe the complexity of the given RNA secondary structures. A standard comparison approach is the computation of edit distances between given RNAs, but even the problem Edit(Crossing, Plain) is MAX-SNP-hard [JLMZ02]. The most tractable variant for the comparison of RNA structures is the reduction to nested RNA secondary structures. But the problem Edit(Nested,Nested) is still NP-hard. With some restrictions to the scoring scheme methods exist, which achieve a pairwise comparison in polynomial time. In the field of global alignment-based methods, which allow mismatches and gaps, the most prominent approach is given by Jiang et al. with the general edit distance for RNA secondary structures [JLMZ02]. See section 2.4 for more details and how this is achieved for the edit distance problem. Previous proposed structural alignment methods have treated base pairs as a whole [BMR95].

With the focus on exact matchings, the LAPCS problem (longest arc-preserving common subsequence) received much attention in literature in the last years [Eva99]. Here the problem is to find an arc-preserving subsequence from two given RNA structures. The problem LAPCS(Nested,Nested) is NP-hard as well but there exist several approximation algorithms [JLMZ00].

In contrast to these methods based on arc-annotated sequences a tree representation for a nested secondary structure is possible as well. These representations are shown in figure 2.4 c) and 2.4 d). Tree-based methods are proposed by Zhang and Shasha [ZS89] for the edit distance between two ordered labeled trees as well as by Jiang et al. for the alignment of trees [JWZ95]. An improved version of the tree alignment method with the extension to global and local forest alignments is given by Höchsmann et al. (RNAforester) [HTGK03]. Two general drawbacks of tree alignment methods are discussed in section 2.3.2.

Local Comparison Methods and Related Approaches

Local comparison approaches are suitable for the search of sequence-structure motifs in RNA. A local alignment-based method is the local sequence-structure alignment algorithm (LSSA), with a scoring scheme comparable to the general edit distance [BW04]. The approach from Gorodkin et al. identifies common stem loops in different RNA structures [GSS01]. The RNAforester algorithm handles local alignment by finding the most similar subtree [HTGK03].

The problem of exact sequence-structure patterns is handled by the maximum common substructure algorithm from Siebert and Backofen [SB07]. This method identifies all exact common substructures for two given nested RNA secondary structures. The main advantage of this approach is that the algorithm needs only $O(nm)$ time to compute all substructures for two RNAs.

In general, protein structure alignment is related to RNA structure alignment if the three-dimensional structure is known. These are for example methods as from Gerstein et al. [GL98]. Further, protein contact maps are a structural representation which is related to crossing secondary structures [LCWI01].

1.3 Contribution

Like mentioned above, the maximum common substructure (MCS) algorithm from Siebert and Backofen [SB07] aims at exact matching substructures between two RNA secondary structures. Moreover, its fast $O(nm)$ running time is at least two magnitudes faster than most sequence-structure comparison methods.

With the motivated importance of sequence-structure motifs for the large variety of RNA functions, the question arises if the found substructures from the MCS algorithm can be used for a pairwise RNA comparison method. The fact that these substructures can represent parts of sequence-structure motifs like for SECIS elements [WSPB97] (see figure 1.1) and that each substructure comprises at least two nucleotides, encourages their usage for a motif-based comparison [SB07]. In addition, the running times of related methods demand for faster algorithms which still yield reasonable results.

In this thesis we present two approaches which are based solely on a precomputed set of exact sequence-structure patterns from two given nested RNA secondary structures. We call these exact substructures *exact pattern matches* (EPMs). Although the MCS algorithm is able to compute *all* exact pattern matches, the matchings itself overlap and cross each other. Algorithms are needed in order to find meaningful subsets of EPMs which represent pairwise similarities between the considered RNAs. For this goal we have developed a suitable Non-Crossing notion for exact pattern matches.

In relation to the LAPCS problem, our first method identifies the best global subset of exact pattern matches for two RNAs. We call this problem the Longest Common Subsequence of Exact RNA Patterns (LCS-ERP) and propose an $O(n^2m^2)$ time and $O(nm)$ space dynamic programming algorithm to solve it.

The second approach tries to identify clusters of EPMs. A cluster is defined as an arrangement of exact pattern matches with a distance constraint, i.e. the distance between two EPMs is below a given threshold value. This opens the possibility of differently defined distance functions. The proposed algorithm uses a greedy strategy to find such clusters in a fast way [Cor01].

Both methods were applied to two pairs of RNA molecules and the results are compared to the solutions found by state-of-the-art approaches RNA_align and RNAforester [JLMZ02, HTGK03].

1.4 Overview

In chapter 2 we give some preliminaries for the later developed methods. We introduce different formalisms for the representation of RNA structures and explain two sequence-structure motifs in detail. In the following we explain basics about related comparison methods like sequence comparison methods as well as sequence-structure comparison methods. In chapter 3 we introduce all notions about exact pattern matches and how they are obtained by the MCS algorithm. We give all necessary definitions on exact pattern matches needed for their algorithmic usage.

Chapter 4 and 5 now present the two developed approaches. The LCS-ERP problem as well as the dynamic programming algorithm to solve it is explained in chapter 4. The proposed clustering algorithm with the different clustering strategies and distance functions is presented in chapter 5.

Chapter 6 shows the achieved results for both methods applied to two pairs of RNA molecules. The solution obtained by the LCS-ERP algorithm is also compared to two existing methods. At the end of this chapter follows a discussion of the results. Chapter 7 concludes this thesis with an outlook for future work.

Chapter 2

Preliminaries

This chapter gives at first an overview of the biopolymer ribonucleic acid (RNA). Starting with its biochemical properties in section 2.1, we explain next the different levels of abstraction needed for RNA comparison methods. In section 2.2 we describe two motifs as examples for functional sequence-structure motifs in RNA. In the following section 2.3 we give basic aspects of pairwise sequence structure comparison. Finally we review in section 2.4 the general edit distance algorithm from Jiang et al. as the most general method for this task.

2.1 Ribonucleic Acid (RNA)

Nucleic acids are biopolymers which consist of covalently linked nucleotides. In the case of ribonucleic acid (RNA) a nucleotide is composed of a heterocyclic base, a ribose and a phosphate group. The nucleotides are linked together by a phosphodiester bond between the 3' carbon and the 5' carbon of adjacent ribose rings. This chain forms the backbone of each RNA molecule. Additionally, each ribose is linked with either a purine or pyrimidine base. Possible purine bases are adenine (A) and guanine (G) and possible pyrimidine bases are cytosine (C) and uracil (U). DNA in comparison to RNA uses thymine instead of uracil as well as a deoxyribose. Uracil is very similar to thymine, but energetically less expensive to produce. Figure 2.1 below illustrates the backbone linkage and the chemical structure of the four bases. In the following we define different levels of abstraction for the structure of RNA molecules and give examples for their representation.

Primary Structure

The asymmetric linkage of the nucleotides induces a direction on the strand. By convention, the 5' end denotes the starting point and determines therewith the order of the linked bases as well. This sequence of nucleotides is called primary structure and is defined as follows. We assume throughout this work for every RNA the four letter alphabet $\Sigma = \{A, C, G, U\}$ as abbreviation for the above mentioned bases.

Definition 2.1.1 (Primary Structure)
Let Σ be a finite alphabet of nucleotides. A primary structure S is a sequence of nucleotides $S = \langle s_1, s_2, ..., s_n \rangle$, where $n \in \mathbb{N}$ and $s_i \in \Sigma$, for $1 \leq i \leq n$.

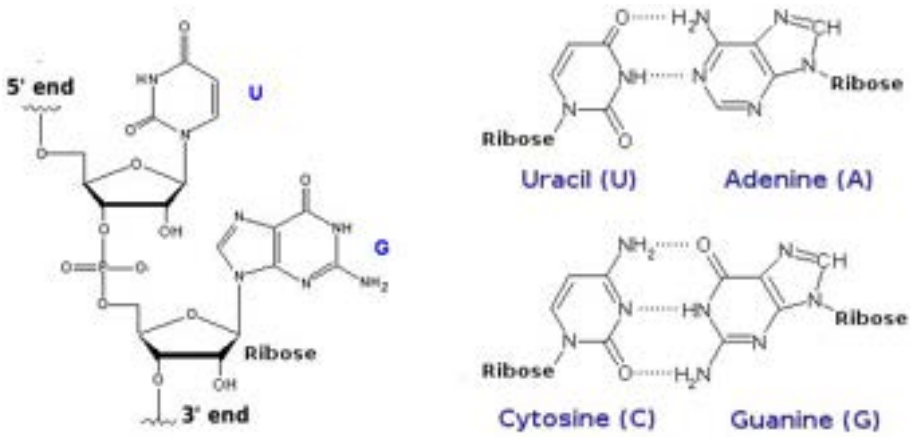

Figure 2.1: The left image shows the linkage between the nucleotides. The right image shows the four standard bases with the two standard Watson-Crick pairs.

With $|S|$ we denote the length of the sequence S and $S[i]$ denotes the nucleotide at position i in sequence S. With $S[i...j]$ we indicate a substring from $S[i]$ to $S[j]$ for $1 \leq i < j \leq |S|$. Please note that it is unimportant at this level of abstraction to distinguish between bases and nucleotides. Therefore we refer to both terms equally.

Due to the fact that the primary structure determines the three-dimensional shape to a large extend for many kinds of biopolymers, it is often sufficient to compare RNAs on this data. From a sequence with a known structure one can infer the structure of homologous sequence. In the first place the primary structure enables to find homologous sequences and provide therewith the basis for many alignment methods given in section 2.3. An example for a primary sequence is shown in figure 2.2.

Secondary Structure

Most RNAs occur as single stranded molecules that fold back onto itself. The formed structure is stabilized by hydrogen bonds between certain pairs of bases and stacking interactions between neighbouring base pairs. The most prominent base pairs are formed between G-C, A-U and G-U bases, ordered by their strength. The first two are usually called canonical base pairs and are shown in figure 2.1, the G-U pair is a wobble base pair. In fact, in nature exist a vast variety of base pairs and nearly all combinations occur even base triplets. But their contribution to the overall stability is minor. For more information see books like "The RNA world" [GCA06].

The following definition formalizes the structural interactions between bases. We call a set of base pairs secondary structure. It is assumed that only pairs of bases are allowed and that each base take part in at most one base pair.

Definition 2.1.2 (Secondary Structure)
Given a primary structure S, a secondary structure B over S is a set of pairs $B = \{(i, i') \mid 1 \leq i < i' \leq |S|\}$, where the tupel (i, i') represents positions in S and indicate

a hydrogen bond between $S[i]$ and $S[i']$. Further it is required that no two base pairs (i, i'), $(j, j') \in B$ share an endpoint, i.e.

$$\forall (i,i'),(j,j') \in B: \quad i \neq j', i' \neq j \text{ and } i = j \iff i' = j' \ .$$

Figure 2.2 shows an example of a secondary structure with indicated base pairs.

Tertiary Structure

The tertiary structure of an RNA describes the three-dimensional arrangement of its atoms and further structural motifs like helical regions. Although the previous abstraction levels are reasonable, the tertiary structure is the key to understand all biological functions and activities of the considered molecule. The main problem is to acquire this data. Our knowledge of exact tertiary structure information is mainly obtained from methods like X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy [GCA06]. These methods are still expensive and time consuming.

Approaches exist which try to predict the secondary and tertiary structure information from the primary structure alone. Methods like `RNAfold` try to predict the thermodynamically most stable secondary structure [HFS+94]. However, the process of folding in vivo is influenced by different parameters and factors as well. Figure 2.2 shows a tertiary structure model for a yeast PHE-tRNA obtained from X-ray diffraction (Protein Data Bank accession id: 1EHZ [BWF+00]).

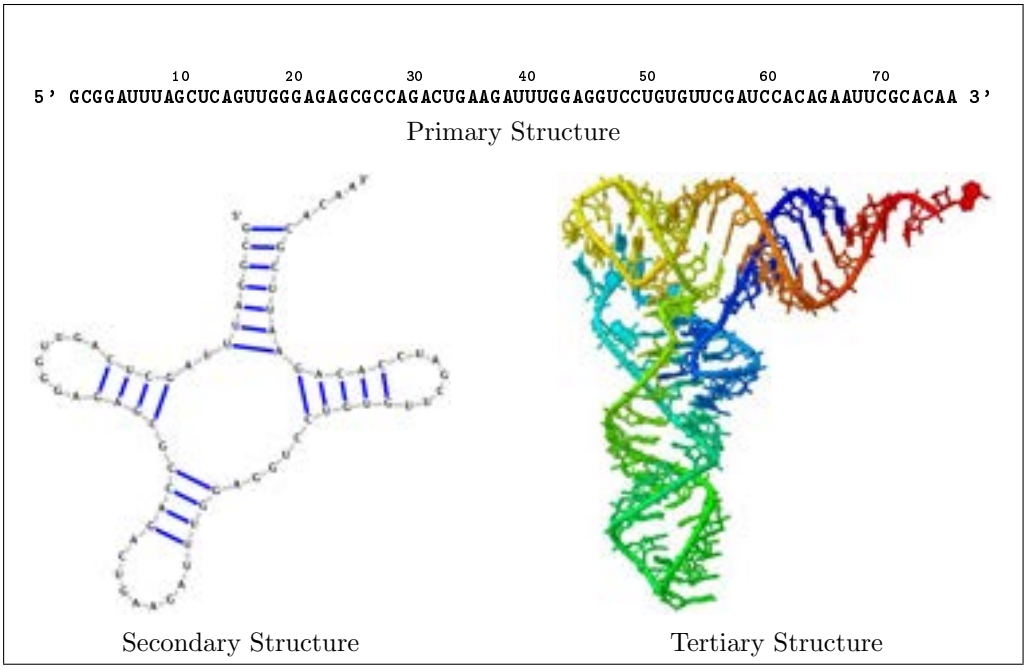

Figure 2.2: Three levels of structural information for a yeast PHE-tRNA. The tertiary structure is taken from the Protein Data Bank, PDB id: 1EHZ.

Nested RNAs

The increased degrees of freedom in a tertiary structure are computationally hard to treat. Therefore we focus in this thesis on the secondary structure as additional information. Depending on a given set of base pairs according to definition 2.1.2, one can distinguish different classes of secondary structures. This is necessary because even on the basis of the secondary structure comparison problems become easily NP-hard.

Definition 2.1.3 (Classes of Secondary Structures)
Given a primary structure S and a secondary structure B over S. Then the secondary structure B is called

CROSSING	:	*if there is at least one crossing base pair in B, i.e.* $\exists (i,i'),(j,j') \in B$ *with* $i < j < i' < j'$,
NESTED	:	*if any two base pairs* $(i,i'),(j,j') \in B$ *are either independent, i.e.* $i < i' < j < j'$, *or nested, i.e.* $i < j < j' < i'$,
PLAIN	:	*if there are no base pairs at all, i.e.* $B = \emptyset$.

Note, that the term tertiary structure is also associated to a CROSSING secondary structure. Specific crossing base pairs are also called pseudoknots [SB05]. Figure 2.4 d) shows a secondary structure which contains crossing base pairs.

For the pairwise comparison of RNAs these classes determine the complexity of the alignment or edit distance problem. For example, it is shown by Jiang et al. that even the edit distance for EDIT(CROSSING,PLAIN) is MAX-SNP-hard [JLMZ02]. With a restriction to the scoring scheme it exists a polynomial time algorithm for EDIT(NESTED,NESTED). Details on these problems are given in sections 2.3 and 2.4.

The two approaches we develop in this thesis require *nested* RNA secondary structures. If it is not mentioned differently, we assume throughout this work RNAs given with a nested secondary structure. For example, figure 2.2 shows a nested secondary structure. We define an RNA molecule with a nested secondary structure as follows.

Definition 2.1.4 (RNA)
Given a primary sequence S and a nested secondary structure B over S. The according RNA $\mathcal{R}$ is denoted by the pair

$$\mathcal{R} = (S, B)$$

Structural Elements

Due to the fact that we consider only single stranded RNA molecules, each base pair $(i,j) \in B$ encloses a chain of nucleotides from $S(i+1)$ to $S(j-1)$. Such a chain is called loop and bases from this loop can form base pairs as well. The resulting structure can be discriminated in six different structural elements. Normally such a loop decomposition is used to determine the energy contributions for the different elements. We use it especially as reference for different shapes of pattern. Figure 2.3 summarizes the structural elements.

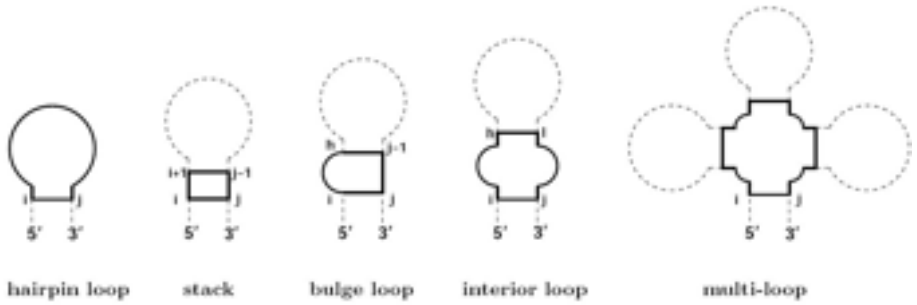

Figure 2.3: Loop decomposition for a nested RNA secondary structure. Figure taken from [Sie06]

Considering a base pair $(i, j) \in B$. If all nucleotides within $S[i+1 \ldots j-1]$ are not part of a base pair, this element is called **hairpin** loop. Now we consider a second base pair $(h, l) \in B$ such that $i < h < l < j$. Depending on the number of nucleotides between i and h as well as between l and j, we distinguish the following elements. If there is at least one nucleotide between i and h and $l = j + 1$, then this is a **left bulge** and if there is at least one nucleotide between l and j and $h = i + 1$, then this is a **right bulge**. If both pairs are not adjacent, this is called **internal loop**. The case of two adjacent base pairs is called **stack**. A several number of stacking base pairs is called **stem**. If there are several stems which branch inside base pair (i, j), this is called **multi-loop**.

Representations of Secondary Structures

Different approaches for the comparison of secondary structures require alternative ways of representation. In figure 2.2 we have already used the two-dimensional structure plot. This is the most convenient way of representation and is the best approximation of the underlying tertiary structure. We use this format especially to indicate the found solutions from our approaches.

Another widespread format is the dot-bracket notion for secondary structures shown in figure 2.4 a). Here a sequence is simply annotated with a sequence of dot and bracket symbols for the representation of the secondary structure. Dots indicate unbound bases and brackets indicate an outgoing hydrogen bond. This text-based format is often used as input format for programs like `RNAfold`. Our implementation uses this format as well. The tree representation from figure 2.4 c) is necessary for tree based algorithms like the tree edit distance algorithm [ZS89].

Figure 2.4 d) shows an arc-annotated sequence. Here arcs represent base pairs. This representation is beneficial to indicate edit operations on single bases and base pairs in one figure. For example see figure 2.7. Arc-annotated sequences are also used to indicate common subsequences.

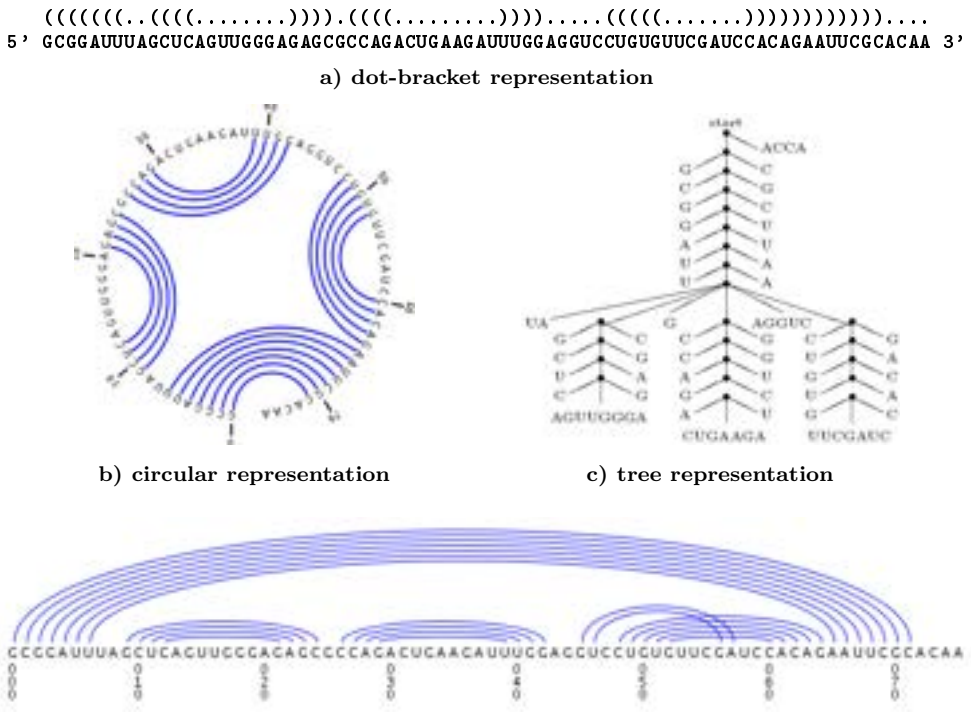

Figure 2.4: Different possibilities to represent an RNA secondary structure. All figures show a yeast tRNA. Please note that figure c) is a slight different tRNA and figure d) includes crossing base pairs. Figures b) and d) are generated with jViz.RNA [WG06], figure c) is taken from [Sie06].

2.2 Motifs and Locality

Motif is a widely used term in biology with different meanings. In terms of RNA a motif is often a three-dimensional part of the RNA molecule with a known or implied function in several RNA molecules. Well known examples are iron-responsive elements (IRE), selenocysteine insertion sequences (SECIS), internal ribosomal entry sites (IRES) and different riboswitches. A good resource for different kinds of motifs is the Rfam database [GJMM+05].

The IRE motif is a small stem with a hairpin. The function of IREs is to bound to iron-responsive proteins (IRP) which are involved in the iron metabolism. For example, the mRNA of ferritin (an iron storage protein) contains an IRE in the 5' UTR. When the iron concentration is low, other IRPs bind to the IRE which leads to translation repression.

A SECIS element is a structural motif that directs the cell to translate the UGA codon as selenocysteine. Normally the UGA codon is the stop codon. This is fundamental for selenoproteins which contain one or more selenocysteine residues. Figure 2.5 shows a set of mammalian SECIS elements occurring in different 3' UTR mRNA regions.

IRES elements occur often in viral genomes. They allow the translation of the virus' RNA in a cap-independent manner. IRES elements bind to the 40S ribosomal subunit

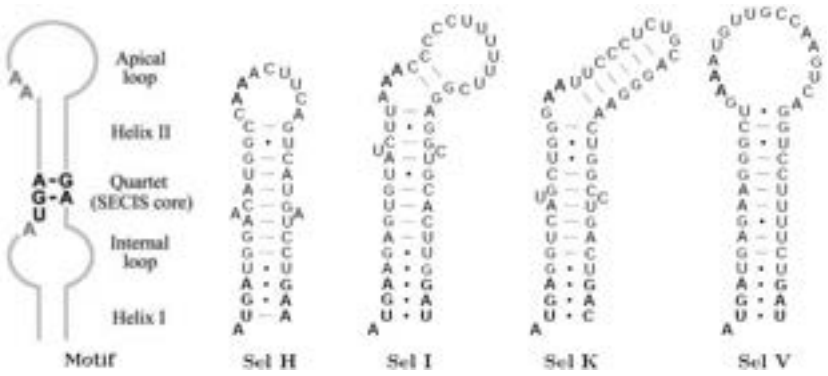

Figure 2.5: A sample set of mammalian SECIS elements with the consensus motif AUGA-AA-GA. Figure according to [KCN+03].

and initiate the translation of the viral mRNA. A riboswitch is a structural motif in mRNAs that can bind a target molecule. The absence or presence of this target molecule affects the gene's activity.

These mentioned motifs provide a classification based on their function. A different classification is based on their structure. According to the introduced structural levels in section 2.1 we can distinguish sequence based motifs and sequence-structure based motifs. This leads to the following view of locality.

Note on Locality

The comparison of RNAs reveal that motifs have a sharp locality, i.e some parts of the molecules share a great similarity, whereas other parts are unrelated. Considering the primary structure alone, motifs can be described as pairs of subsequences. The nucleotides of these subsequences are connected via backbone bonds which constitutes their dependency. Local sequence alignment methods find local motifs in this way.

However, considering motifs like shown in figure 1.1 as well as the given examples above, these motifs express a different locality. Here the indicated motif is not local if only the primary structure is considered. The apparent similarity and locality is given by sequential and structural features together, notably the included base pairs. Biologically this is meaningful because sequence and structure is conserved during evolution.

This view imply the definition of a motif as (connected) substructure and provide the basis for the developed methods in this thesis. The maximum common substructure algorithm from Siebert and Backofen [SB07] described in section 3.4 is able to find such local sequence-structure relationships. This algorithm identifies all exact matching substructures between two RNAs. For example. the substructure shown in figure 1.1 can be found with this algorithm. With the same view on locality but with the focus on inexact matchings, the local sequence-structure alignment (LSSA) algorithm can be used [BW04].

2.3 Pairwise Sequence-Structure Comparison

Pairwise comparison of RNA or DNA sequences is an essential task in biological sequence analysis with two main goals. According to the theory of evolution, sequences are derived from common ancestral sequences. First, it is interesting to trace the evolutionary history of mutations and other evolutionary changes. Sequence comparison in this context is understood as a measure of the evolutionary relatedness, called homology. The second line tries to figure out similar sequences or regions with putative similar functions. Due to the fact that a biological function in RNAs often coincides with similar structural properties as shown in section 2.2, the incorporation of structural information is mandatory. The approaches developed in this thesis follow this line of research.

From a theoretical point of view the similarity or homology of two sequences can be measured with the number of mutations, insertions and deletions of bases which are necessary to transform one sequence into the other. First, we introduce this concept for sequences alone and afterwards we give an overview in which way secondary structure information can be incorporated. A different measure for similarity are common subsequences. We introduce this concept for sequences and sequences with additional structural information.

2.3.1 Sequence-Based Comparison

The comparison of sequences can bee seen under two different aspects. The first tries to quantify the similarity between two sequences whereas the other focus on their distance. A similarity measure associates a numeric value with a pair of sequences with the idea that a higher value indicates greater similarity. In biology similarity measures are associated with alignments which shows the conserved regions. The concept of distance is dual to this with the idea that a larger distance imply a smaller similarity and vice versa.

The simplest notion of distance is the so-called Hamming Distance [Gus97]. For two sequences of equal length the number of different characters is count. A more general distance measure is defined as the minimum number of edit operations to transform one sequence into the other. In order to compare sequences with different lengths, an additional gap symbol "-" is needed. The permitted edit operations are defined as follows.

Definition 2.3.1 (Edit Operation)
Given an finite alphabet Σ, we define an edit operation as pair

$$(x, y) \in (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}).$$

Further we call an edit operation (x, y)

$$\begin{array}{ll} \textit{insertion,} & \textit{if } x = - \textit{ and } y \in \Sigma, \\ \textit{deletion,} & \textit{if } x \in \Sigma \textit{ and } y = -, \\ \textit{substitution,} & \textit{if } x, y \in \Sigma \textit{ with } x \neq y. \end{array}$$

In the following we assume two given sequences a, b over a finite alphabet Σ. Then we write $a \rightarrow_{(x,y)} b$ if sequence b can be obtained from sequence a by a replacing of one occurrence of x by y, or by deleting one occurrence of x (if $y = -$), or by inserting one occurrence of y (if $x = -$). Usually a single edit operation is not sufficient to transform one sequence into the other. Then we need a sequence of edit operations $E = e_1, \ldots, e_r$ with $a = a^{(0)} \rightarrow_{e_1} a^{(1)} \rightarrow_{e_2} \cdots \rightarrow_{e_r} a^{(r)} = b$. In short we will write this as

$$a \Rightarrow_E b \ .$$

The edit operations *insertion* and *deletion* can be seen as symmetric cases and in biology they often combined as *indel* operation. With these definitions it is possible to formulate optimization problems on edit distances. The edit transcript that comprises the minimal number of edit operations is the so-called Levenshtein Distance [Gus97]. More general approaches assign a weight $\omega(x, y)$ to edit operations and score the *cost* of a sequence of edit operations $E = e_1, \ldots, e_r$ as

$$\omega(E) = \sum_{i=1}^{r} \omega(e_i) \ .$$

Such a weight-based cost function allows a good adaptation to different problems and scenarios in biology. Now we can formulate the general optimization problem for edit distances as follows.

Definition 2.3.2 (Edit Distance)
Let Σ a finite alphabet and a, b two sequences over Σ. Further let $\omega : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$ a cost function and E a sequence of edit operations. Then we define the edit distance of a, b as

$$d_\omega(a, b) = \min\{\omega(E) \mid a \Rightarrow_E b\}.$$

Obviously the solution depends mainly on the chosen cost function. A reasonable class of cost functions is defined as metric. This imply that equal sequences have no distance, i.e. if $a = b$ then $d(a, b) = 0$, as well as the cost function is symmetric, i.e. $d(a, b) = d(b, a)$. Further, a metric holds the triangle inequality, i.e. $d(a, c) \leq d(a, b) + d(b, c)$.

Sequence Alignment

Like introduced above, alignments are strongly related to the problem of edit distances. Therefore it is possible to solve both problems with a similar method. In the following we first define alignments in general. Second we show in short their relationship to edit distances. Then we describe a standard method to solve both problems.

Definition 2.3.3
Let Σ be a finite alphabet without the gap symbol, i.e. $- \notin \Sigma$, and a, b two sequences over Σ. Further let Σ_A the alignment alphabet $(\Sigma \cup \{-\})$. Two sequences $\hat{a}, \hat{b}$ over Σ_A denote a pairwise alignment $\mathcal{A}$ if

1. *The aligned sequences have equal length, i.e. $|\hat{a}| = |\hat{b}|$,*

2. *Sequence $\hat{a}$ gives a and sequence $\hat{b}$ gives b if all gaps are removed,*
3. *There is no position i such that $\hat{a}_i = - = \hat{b}_i$.*

For example suppose two sequences a =AUGCACAGAA and b =AUCCGACGAC. A possible alignment is

$$\hat{a} = \texttt{AU--GCACAGAA}$$
$$\hat{b} = \texttt{AUCCG-AC-GAC}$$

Similar to the cost of an edit transcript, we can define the cost of an alignment as $\omega_{\mathcal{A}}(\hat{a}, \hat{b}) = \sum_{i=1}^{|\hat{a}|} \omega(\hat{a}_i, \hat{b}_i)$. Now we can measure the distance of an alignment similar to the edit distance as

$$d_{\omega}^{\mathcal{A}}(a, b) = \min\{\omega_{\mathcal{A}}(\hat{a}, \hat{b}) \mid (\hat{a}, \hat{b}) \text{ is alignment of } (a, b)\}.$$

In other words, the alignment $(\hat{a}, \hat{b})$ is optimal if $d_{\omega}^{\mathcal{A}}(a, b) = \omega_{\mathcal{A}}(\hat{a}, \hat{b})$. The important implication is that for every alignment $(\hat{a}, \hat{b})$ of (a, b) there exists a sequence of edit operations E such that $a \Rightarrow_E b$ and $\omega(E) = \omega_{\mathcal{A}}(\hat{a}, \hat{b})$ for a metric cost function ω. Second, for every sequence E such that $a \Rightarrow_E b$, there exists an alignment $(\hat{a}, \hat{b})$ of (a, b) that hold $\omega(\hat{a}, \hat{b}) \leq \omega(E)$.

For more details on sequence alignments and edit distances we refer to standard literature like [Gus97, CB00]. Summarizing, there are three important implications as a consequence from cost models for sequence alignment:

1. *The cost of an alignment of two sequences a and b is the sum of the costs of all edit operations that lead from a to b.*
2. *An optimal alignment of a and b is an alignment which has minimal cost among all possible alignments.*
3. *The edit distance of a and b is the cost of an optimal alignment of a and b under a cost function ω.*

Global Sequence Alignment Methods

Due to the strong relation between edit distances and optimal alignments it is possible to solve both problems with similar methods. Today exists a vast variety of algorithms for sequence alignment with different approaches. Most of them use dynamic programming (DP) techniques for finding optimal solutions in an efficient manner. The main idea of DP is constructing the overall optimal solution from optimal subproblems. Richard Bellman introduced this technique in the 1940s. The central result of dynamic programming is a single recursion formula for the optimization problem.

Basically, one can distinguish between methods which find alignments covering the whole sequence and methods which calculate optimal local alignments covering only subsequences. Global methods are useful for comparing sequences of a functional family of

different species whereas local methods are used to find particular domains or functional subunits. First methods for global sequence alignment were given by Needlemann and Wunsch [NW70] and Gotoh (affine gap cost) [Got82]. First local methods were developed by Smith and Waterman [SW81] and Altschul et al. (BLAST) [AGM+90].

In the following we give some details for the Needleman-Wunsch algorithm as one of the first approaches for global sequence alignment. Given a metric cost function ω and two sequences S_1 and S_2, the recursion formula to obtain an optimal alignment is given as

$$D(i,j) = \min \begin{cases} D(i-1,j-1) & + \omega(S_1[i], S_2[j]) \\ D(i-1,j) & + \omega(S_1[i], -) \\ D(i,j-1) & + \omega(-, S_2[j]) \end{cases} \tag{2.1}$$

with $1 \leq i \leq |S_1|$ and $1 \leq j \leq |S_2|$. The three cases indicate in which way a single field in a 2-D matrix is filled. The matrix needs an appropriate initialization of $D(0,0), D(i,0), D(0,j)$ with the indel costs up to position i and j. The algorithm fills the matrix from $D(1,1)$ at the upper left corner to the lower right corner $D(|S_1|, |S_2|)$. This position contains the alignment distance of both sequences. The corresponding alignment is obtained from a traceback through the filled matrix. With a constant cost model, this algorithm needs $O(nm)$ time and $O(nm)$ space. The traceback needs additional $O(n+m)$ time [Gus97].

In general, the term "cost" implies typically positive values which have to be minimized for the overall task. Terms like "scores" or "weights" are positive or negative and they are used for similarity measures, which means they have to be maximized. Moreover, gap costs can be modeled separately in order to adopt the alignment to biological processes like the intron/exon structure of cDNA. For a good summary we refer to standard literature [Gus97].

Longest Common Subsequence

A special case of optimal sequence similarity is the longest common subsequence (LCS) [Hir77]. With an adopted scoring scheme, the LCS problem can be solved with methods for normal sequence alignment.

Definition 2.3.4 (Subsequence)
Given two Sequences S and S' over some alphabet Σ, S is a subsequence of S', if S can be obtained from S' by deleting some letters from S'.

Note that the subsequence S need not consist of consecutive letters in S'. That is the main difference to a substring. As a *common* subsequence one denotes a subsequence shared by two or more sequences. For the pairwise case we can formulate the problem as follows.

Definition 2.3.5 (Longest Common Subsequence)
Given two Sequences S_1 and S_2 over some alphabet Σ. The longest common subsequence is a sequence T which is a subsequence of both S_1 and S_2 and has maximal length.

The problem LCS can be computed with a recursion given in formula (2.1), if one uses a maximization together with a scoring scheme that scores a match with one and all mismatches with zero. Given two RNAs with their sequence lengths $n = |S_1|$ and $m = |S_2|$, the problem LCS for two RNAs is solvable in $O(nm)$ time and $O(nm)$ space.

2.3.2 Sequence-Structure Comparison

According to the edit distances for plain sequences one can define edit distances for sequences given with their primary and secondary structures. The main problem for the design of algorithms is that the problem becomes easily NP-hard. Zhang et al. have shown that for EDIT(CROSSING,CROSSING) [ZWM00] and Jiang et al. [JLMZ02] showed that already the case EDIT(CROSSING,PLAIN) is MAX SNP-hard under arbitrary scoring schemes. Recently the most interesting problem for molecular biology EDIT(NESTED,NESTED) was proven to be NP-hard as well [BFRS03]. These findings imply that there exists no polynomial time algorithms to solve these problems efficiently.

Here we focus on nested structures because our approaches are based on nested RNAs as well. Although the general problem is NP-hard, there exist polynomial time algorithms which compare sequential information along with structural information. In the case that two RNAs are given with their primary and nested secondary structure several methods exist like from Zhang and Shasha [ZS89], Eddy [Edd02], Bafna et al. [BMR95], Jiang et al. [JWZ95, JLMZ02].

The proposed methods for nested structures differ in general in the representation of the secondary structure. Visualized in figure 2.4 c), Zhang and Shasha [ZS89] and Jiang et al. [JWZ95] use ordered labeled trees with base pairs as internal nodes. The Zhang/Shasha algorithm needs $O(|T_1|\ |T_2|\ \min(depth(T_1), leaves(T_1))\min(depth(T_2), leaves(T_2)))$ time to compute the minimum edit distance between two trees T_1 and T_2.

Although tree alignments achieve good results, there are two general drawbacks for using trees in the context of alignments. First, edit distance and alignment distance can be different for two trees. Like for plain sequences the edit distance of trees describes a sequence of predefined edit operations (deletion, insertion, relabeling) on nodes and leaves to transform one tree into the other. However, the alignment of two trees consists of inserting nodes with gap symbols in order to get two identical trees except for their labels. See figure 2.6 for this difference.

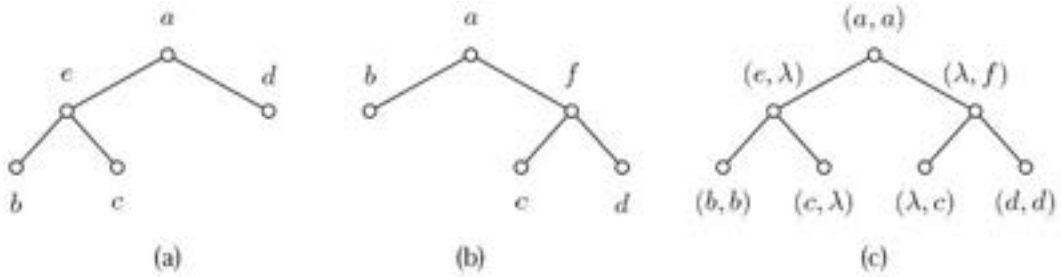

Figure 2.6: (a) and (b) show two different trees. (c) shows the alignment of both trees. Figure according to Jiang et al. [JWZ95].

For example, assume a cost function with cost 1 for all edit operations. The optimal sequence of edit operations is obtained by deleting node e and inserting the node labeled f. Hence, the edit distance is 2. The optimal alignment however is the tree shown in 2.6 c) with a value of 4. In contrast, as stated in point three for the general results of sequence alignment in section 2.3.1, edit and alignment distance are equal for plain sequences.

Second, and that is the most decisive point against tree alignment, an arbitrary sequence alignment is not necessarily a valid tree alignment. This is the case, if a base pair is aligned with one or two gap symbols. This case is not recognized as change operation and hence it is not an allowed tree edit operation.

Better suiteable for arbitrary alignments are methods based on arc-annotated sequences like the works from Bafna et al. [BMR95] and Jiang et al. [JLMZ02]. An example of an arc-annotated sequence is shown in figure 2.4 d). The algorithm from Bafna et al. [BMR95] was already capable to align a base pair as whole or not within a time complexity of $O(n^2m^2)$. Jiang et al. [JLMZ02] proposed an algorithm which needs $O(n^3m)$ time to handle any arbitrary alignment but with a specific scoring scheme. This is the most important work in the context of sequence-structure comparison and is reviewed in section 2.4.

Local sequence-structure comparison methods exists as well. Notably is the local sequence-structure alignment algorithm from Backofen and Will [BW04]. This approach uses a scoring scheme comparable to the general edit distance scheme from Jiang et al. [JLMZ02]. The algorithm has a time complexity of $O(n^2m^2 \max(n, m))$ and a space complexity of $O(nm)$.

Longest Arc-Preserving Common Subsequence (LAPCS)

According to the LCS problem for primary structures, the Longest Arc-Preserving Common Subsequence Problem (LAPCS) describes the extension to higher structural levels. This model for sequence similarity was introduced by Evans [Eva99] and received much attention in literature in the last years [JLMZ00, LCJW02, GGN02]. In biology the LAPCS problem is useful as a similarity measure for comparing sequence with secondary structure information. The problem can be defined as follows.

Definition 2.3.6 (Longest Arc-Preserving Common Subsequence (LAPCS)) *Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. LAPCS is the problem to find the longest common subsequence of S_1 and S_2 which preserves the arcs, i.e. to find a mapping $\mathcal{M}_{\mathsf{LAPCS}} \subseteq \{1, ..., |S_1|\} \times \{1, ..., |S_2|\}$ of maximal size such that:*

1. *$\mathcal{M}_{\mathsf{LAPCS}}$ is a bijective mapping and preserves the order of the subsequence:*
   $\forall (i, j), (i', j') \in \mathcal{M}_{\mathsf{LAPCS}} : i = i' \iff j = j',\ i < i' \iff j < j'$
2. *the base pairs induced by $\mathcal{M}_{\mathsf{LAPCS}}$ are preserved:*
   $\forall (i, j), (i', j') \in \mathcal{M}_{\mathsf{LAPCS}} : (i, i') \in B_1 \iff (j, j') \in B_2$
3. *$\mathcal{M}_{\mathsf{LAPCS}}$ produces a common subsequence:*
   $\forall (i, j) \in \mathcal{M}_{\mathsf{LAPCS}} : S_1[i] = S_2[j]$

Depending on the complexity of the arc set B, the complexity to solve the problem varies. Similar to the edit distance problem, it is shown by Jiang et al. that LAPCS(NESTED, NESTED) is NP-hard [LCJW02]. If the complexity of the second structural type is PLAIN or CHAIN, which means all $(i, i') \in B$ hold only the independence condition, LAPCS can be solved in polynomial time [JLMZ00].

2.4 General Edit Distance of RNA Structures

In order to handle any arbitrary pairwise alignment from primary and secondary structures, Jiang et al. introduced a method for generally scoring alignments [JLMZ02]. The main idea from Jiang et al. is to define the edit distance for arc-annotated sequences via sequence alignment instead of an edit transcript. Arcs are treated as basic unit of comparison and they could be aligned to single bases as well as to gaps. Tree alignments and also the covariance model lack this generality as they focus on local structures and treat them as subunits.

According to Jiang et al. and the notions from section 2.1, an RNA is defined with its primary and secondary structure, denoted as pair (S, B). Further any $(i, i') \in B$ is drawn as arc in addition to the straight sequence. A picture of such an arc-annotated sequence is given in figure 2.4 d). Consequently edit operations from sequence alignment (see def. 2.3.1) have to be extended with structural edit operations performed on arcs.

In the following we describe the general problem and in section 2.4.2 we describe in detail the proposed polynomial time algorithm for EDIT(NESTED,NESTED).

2.4.1 Edit Operations and Problem Description

Considering two arc-annotated sequences (S_1, B_1) and (S_2, B_2) and a specific sequence alignment $\mathcal{A}$. The performed edit operations given by $\mathcal{A}$ can be distinguished between operations on arcs and its incident bases and operations on bases. A base without an incident arc is called *free* base.

Base Operations: Possible edit operations for bases are the same as for standard sequence alignment, i.e. base match, base mismatch and base deletion/base insertion. If a base $S_1[i]$ is aligned to $S_2[j]$ and $S_1[i] = S_2[j]$ then $\langle S_1[i], S_2[j] \rangle$ is a *base-match* if it is not involved in any arc operation. If $S_1[i] \neq S_2[j]$, then $\langle S_1[i], S_2[j] \rangle$ is a *base-mismatch*. Aligning a base $S_1[i]$ with a gap, this is a *base-insertion* in S_1 and a *base-deletion* in S_2. If there is an alignment of a base $S_2[j]$ with a gap in S_1, this holds vice versa. Note that the bases in these mutation operations are not necessarily free. An example is given after the arc operations.

Arc Operations: Suppose two arcs $(i, i') \in B_1$ and $(j, j') \in B_2$ such that $S_1[i]$ is aligned to $S_2[j]$ and $S_1[i']$ is aligned to $S_2[j']$. If $S_1[i] = S_2[j]$ and $S_1[i'] = S_2[j']$, this is an *arc-match* operation and if $S_1[i] \neq S_2[j]$ or $S_1[i'] \neq S_2[j']$, then they form an *arc-mismatch*. An *arc-breaking* occurs if the bases are aligned as above but $(j, j') \notin B_2$. If an arc $(i, i') \in B_1$ is aligned with one base, say j, and with one gap, this is an *arc-altering* operation, as the arc from B_1 is broken. Further an *arc-removing* occurs if an arc is

aligned with two gaps. This completely removes the arc as the two bases are deleted. The last three operations can be summarized as arc-deletion operation, as they have a break of an arc in common. Biologically these arc operations can be interpreted as evolutionary events like changing or removing bases on base pairs. A summery of the edit operations shows figure 2.7. Further a distance measure requires a cost associated to each edit operation.

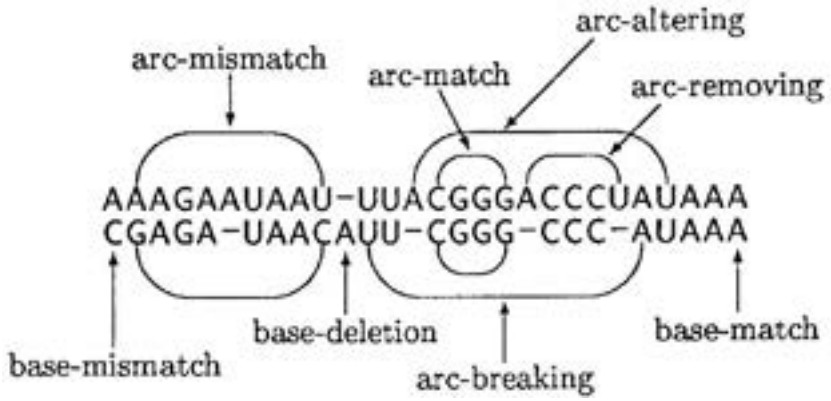

Figure 2.7: The allowed edit operations for an alignment of two arc-annotated sequences. Figure according to [JLMZ02].

Cost Scheme: The main idea is to separate costs for arc operations and that for base operations and computing costs for arcs first. For example, suppose an alignment of an arc $(i, i') \in B_1$ with a base $j \in S_2$, $S_1[i] \neq S_2[j]$ and a gap. Then the costs are composed from an *arc-altering* operation plus a *base-mismatch* operation. An arc-match and a base-match cost nothing as a distance measure is used. Further a base-mismatch has cost ω_m and a base deletion ω_d. An arc-mismatch has cost $\frac{w_{am}}{2}$ or ω_{am} depending on the bases of the two arcs involved. Suppose again two arcs with $S_1[i]$ is aligned to $S_2[j]$ and $S_1[i']$ is aligned to $S_2[j']$. If $S_1[i] \neq S_2[j]$ or $S_1[i'] \neq S_2[j']$, then the arc-altering cost is $\frac{w_{am}}{2}$, whereas if both pairs of aligned bases are unequal, the cost is ω_{am}. An arc-breaking has cost ω_b, an arc-altering has cost ω_a and an arc-removing has cost ω_r which is usually $\omega_r \geq \omega_d$.

Problem Description

With these six edit operations $(\omega_m, \omega_d, \omega_{am}, \omega_b, \omega_a, \omega_r)$ it is possible to form a legal series of edit operation for any alignment. Recall now the three results for alignment and edit distances from section 2.3. Similar to the edit distance of two sequences, the edit distance of two arc-annotated sequences is defined as the minimum cost of the alignments of the two sequences. Equivalently the problem is to compute the optimal series of edit operations which transforms the first sequence into the second one along with an optimal sequence alignment. Jiang et al. called this the Edit Distance Problem for Arc-Annotated Sequences under a fixed scoring scheme for the six parameters. The complexity of this problem depends mainly on the complexity of the given arc structures of the two input sequences, denoted as Edit(Type1,Type2). Jiang et al. showed that the problem Edit(Nested, Plain) is MAX-SNP-hard for an arbitrary scoring scheme. But with

some restrictions to the scoring scheme, they were able to formulate a polynomial time algorithm for the case Edit(Crossing, Nested) and Edit(Nested, Nested).

2.4.2 A polynomial time algorithm for EDIT(Nested, Nested)

Jiang et al. proposed an algorithm which solves the case Edit(Crossing, Nested) in $O(n^3m)$ time and $O(n^2m)$ space. In addition they give an improved algorithm which needs $O(n^2m^2)$ time and only $O(nm)$ space. Here we focus on the case Edit(Nested, Nested) for which the same bounds hold and go into detail to the improved space version.

The reduction in complexity is achieved with a class of scoring schemes satisfying the condition $2\omega_a = \omega_b + \omega_r$. Now it is possible to omit the explicit calculation of the operations arc-altering and arc-removing as they could be incorporated in the remaining operations. Every arc-altering operation is handled as an arc-breaking operation plus a base-deletion which has cost $\frac{\omega_r - \omega_b}{2}$. Further any arc-removing operation is handled as an arc-breaking operation plus two base-deletions each of which has cost $\frac{\omega_r - \omega_b}{2}$. In addition, Jiang et al. introduced two functions to simplify the algorithmic writing:

$$\psi_k(l) = \begin{cases} 1, & \text{if } S_k[l] \text{ is } \textit{not} \text{ a free base} \\ 0, & \textit{otherwise} \end{cases} \tag{2.2}$$

$$\chi(i,j) = \begin{cases} 1, & \text{if } S_1[i] \neq S_2[j] \text{ (base-mismatch)} \\ 0, & \textit{otherwise} \end{cases} \tag{2.3}$$

With formula 2.2 the cost of deleting a base $S_1[i]$ is $(1 - \psi_1(i))\omega_d + \psi_1(i)\omega'_d = \omega_d + \psi_1(i)(\frac{\omega_r - \omega_b}{2} - \omega_d)$, with $\omega'_d = \frac{\omega_r - \omega_b}{2}$ as stated above as cost for deleting a base with an incident arc. The key idea to discard also the arc-breaking operation is to split its costs equally among its two incident bases and charge an additional cost of $\psi_1(i)\frac{\omega_b}{2}$ for base $S_1[i]$ if the arc is broken. The improvement in space complexity is achieved with the same idea of splitting costs applied to base-match and base-mismatch operations. The remaining arc operations arc-match and arc-mismatch can be handled by using formula 2.3. Consequently, the following formulas handle all arc and base operations:

$$\text{base-deletion:} \quad \omega_d + \psi_k(l)(\tfrac{\omega_r}{2} - \omega_d) \tag{2.4}$$

$$\text{base-match/ base-mismatch:} \quad \chi(i,j)\,\omega_m + (\psi_1(i) + \psi_2(j))\,\tfrac{\omega_b}{2} \tag{2.5}$$

$$\text{arc-match/ arc-mismatch:} \quad \big(\chi(i',j') + \chi(i',j')\big)\,\tfrac{\omega_{am}}{2} \tag{2.6}$$

According to formula 2.5, the cost of a base-match with two free bases is zero; if one base is free, the cost is $\frac{\omega_b}{2}$ and if both bases are not free, the cost is $2 \cdot \frac{\omega_b}{2} = \omega_b$. Applied to a base-mismatch, formula 2.5 yields for two free bases the cost ω_m; if one base is free, the the cost is $\omega_w + \frac{\omega_b}{2}$ and if both bases are not free, the cost is $\omega_m + 2 \cdot \frac{\omega_b}{2} = \omega_m + \omega_b$. Using formula 2.6, an arc-match costs zero and an arc-mismatch costs either $\frac{\omega_{am}}{2}$ or ω_{am}, depending on the number of mismatches.

The recurrence relation for a dynamic programming algorithm to solve Edit(Nested, Nested) is given as follows. Note that (i, i') and (j, j') are not necessarily base pairs.

For any $1 \leq i \leq i' \leq n$ and $1 \leq j \leq j' \leq m$,

$$DP(i,i',j,j') = \min \begin{cases} DP(i,i'-1,j,j') + \omega_d + \psi_1(i')(\frac{\omega_r}{2} - \omega_d), \\ DP(i,i',j,j'-1) + \omega_d + \psi_2(j')(\frac{\omega_r}{2} - \omega_d), \\ DP(i,i'-1,j,j'-1) + \chi(i',j')\omega_m + (\psi_1(i') + \psi_2(j'))\frac{\omega_b}{2}, \\ DP(i,r-1,j,s-1) + DP(r+1,i'-1,s+1,j'-1) \\ \quad +(\chi(r,s) + \chi(i',j'))\frac{\omega_{am}}{2} \\ \qquad \textbf{if } \; i \leq r, j \leq s, (r,i') \in B_1, (s,j') \in B_2. \end{cases} \tag{2.7}$$

To compute all combinations, the algorithm needs $O(n^2m^2)$ time. To store all entries, one could expect the same space complexity. However, the reduction to $O(nm)$ is achieved as it is sufficient to maintain $DP(i+1, i'-1, j+1, j'-1)$ only if $(i, i') \in B_1$ and $(j, j') \in B_2$. This could be clarified as follows. There are maximal $O(n)$ arcs for the first sequence and maximal $O(m)$ arcs for the second sequence. If the arcs are computed from inside to outside and taking arcs with minimal size first, this needs only $O(nm)$ space to store for all combinations of arcs their minimum costs. Another $O(nm)$ matrix is now filled in the manner of classical alignment while using the stored minimum costs for the arc intervals. With this analysis we can give the following conclusion according to Jiang et al. 2002. Under any scoring scheme satisfying $2\omega_a = \omega_b + \omega_r$, the problem Edit(Nested, Nested) is solvable in $O(n^2m^2)$ time and $O(nm)$ space.

A reasonable cost scheme for $(\omega_m, \omega_d, \omega_{am}, \omega_b, \omega_a, \omega_r)$ was also given by Jiang et al. with the values $(1, 1, 1.8, 1.5, 1.75, 2)$.

Chapter 3

Exact Matchings in RNA Structures

In this chapter we give prerequisites needed to use common substructures in RNA molecules for the pairwise comparison approaches in chapter 4 and 5. Section 3.1 introduces our concept of *exact pattern matches* (EPMs) as common substructures with exact sequential and structural properties of two RNA secondary structures. Considering two RNAs with a known secondary structure, there exist maximal $n \cdot m$ different EPMs which cross and overlap each other. However, this limited set imply additional properties which are useful for the algorithmic usage of exact pattern matches. These features are summarized in section 3.2.

The main goal is to find a good arrangement of a selection of non-crossing and non-overlapping exact pattern matches. Therefore notions and definitions are introduced in section 3.3 to handle the sequential and structural properties of EPMs as well as the relationships of different exact pattern matches in an algorithmic manner. A fast detection of exact pattern matches is an important precondition. Therefore, we review in section 3.4 the maximum common substructure algorithm (MCS) from Siebert and Backofen [BS04, SB07] which identifies all crossing and overlapping EPMs for two nested RNA secondary structures.

3.1 Basic Definitions for Matchings

In the following we give notions for exact patterns in two nested RNA secondary structures. Our terms are based on the articles for the MCS-algorithm [BS04, SB07]. First we define a substructure as a connected pattern in a single RNA secondary structure. In the following we extend this to an exact matching pattern in two RNAs. This is achieved with an exact matching path of nucleotides with identical sequential and structural properties in both RNAs. We call this the exact pattern match (EPM) problem.

Patterns in one RNA

Here we focus on exact sequence-structure patterns in a single RNA in contrast to approximate patterns. In addition to pure sequential patterns, the crucial point for sequence-structure patterns is the incorporated structural context of each single nucleotide part of the pattern. Clearly, the structural context of a nucleotide is formed by structural adjacent nucleotides. Considering a single RNA secondary structure, this is achieved with

either a backbone bond (phosphodiester bond) or an hydrogen bond. A set of nucleotides connected by these bonds is a primitive pattern. We call this *path* and it is defined as follows. Note, it doesn't matter how often a nucleotide is taken into the path. The function $S[i]$ returns the nucleotide for position i.

Definition 3.1.1 (Path)
Let $\mathcal{R} = (S, B)$ be a given RNA. A path in $\mathcal{R}$ from a nucleotide at position i to a nucleotide at position j is a sequence of positions $\langle p_1, p_2, .., p_k \rangle$ such that $p_1 = i$ and $p_k = j$ and there exist either a backbone bond or a hydrogen bond between $S[p_{l-1}]$ and $S[p_l]$, for $l = 2, ..., k$,.

A pattern in a single RNA is defined as a set of positions which holds the path condition. Each nucleotide position in a pattern is connected via a path with every other nucleotide position in the same pattern. With other words, the locality of a pattern ends with the "borders" of the pattern. We denote such borders as *bounds*. Different definitions on bounds are given in section 3.3.

Definition 3.1.2 (Pattern)
Let $\mathcal{R} = (S, B)$ be a given RNA. A pattern $\mathcal{P}$ of size k in $\mathcal{R}$ is a set of positions $\mathcal{P} = \{p_1, p_2, ..., p_k\}$ such that for any two nucleotides $S[p_i]$ and $S[p_j]$, $p_i, p_j \in \mathcal{P}$, there exists a path from p_i to p_j completely lying in $\mathcal{P}$.

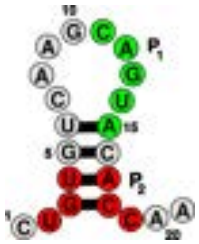

Figure 3.1: The figure shows two patterns P_1 and P_2 with $P_1 = \{11, 12, 13, 14, 15\}$ and $P_2 = \{2, 3, 4, 17, 18, 19\}$.

Figure 3.1 shows a simple example of two patterns. For later it is important to show, if two patterns have one nucleotide position in common, then these two patterns are connected.

Proposition 3.1.1 (Connected Pattern)
Given two patterns $\mathcal{P}_1 = \{p_1, p_2, ..., p_i\}$ and $\mathcal{P}_2 = \{q_1, q_2, ..., q_j\}$ for an RNA $\mathcal{R} = (S, B)$. If there exists at least one nucleotide position n_c such that $n_c \in \mathcal{P}_1$ and $n_c \in \mathcal{P}_2$, then the union of the two patterns is connected, i.e. it is a pattern.

Proof. Suppose there exists a position n_c with $n_c \in \mathcal{P}_1$ and $n_c \in \mathcal{P}_2$. For any nucleotide at position $n_p \in \mathcal{P}_1$, there exists a path $\langle n_p, ..., p_c \rangle$ such that each nucleotide on the path is lying in the first pattern. The same holds for any nucleotide position $n_q \in \mathcal{P}_2$. Hence, there exists a path from any nucleotide at position $n_p \in \mathcal{P}_1$ to n_c to any nucleotide at position $n_q \in \mathcal{P}_2$. □

Matchings Over Two RNAs

Given the definition of a pattern in a single RNA, we extend this definition to two RNAs. This requires for each nucleotide position of an exact matching identical sequential and structural properties as well as an identical structural context.

Clearly, the smallest matching between two RNAs is an identical nucleotide with an identical structural property. In contrast to pure sequence alignment, our approach implies looking at the structural type of a nucleotide.

Consider two arbitrary RNAs with $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. Further let V_1 with $V_1 = \{r \mid 1 \leq r \leq |S_1|\}$ the set of positions for $\mathcal{R}_1$ and V_2 with $V_2 = \{s \mid 1 \leq s \leq |S_2|\}$ the set of positions for $\mathcal{R}_2$. $S_i[j]$ denotes the nucleotide at position j in sequence i. The function $STRUCT_i(j)$ yields the structural type for a nucleotide at position j in structure i. For a secondary RNA structure, three *structural types* for any single nucleotide are feasible: *single stranded* (ss), *left paired* (lp) or *right paired* (rp). If the nucleotide is not involved in any base-pairing interaction, then this is called single stranded or unbound. If the nucleotide is left paired, then the base-pair partner has a higher position in the sequence. If a nucleotide is right paired, then the base-pair partner has a lower position.

The set of nucleotides identical with their primary and secondary structure in two RNAs is called *partial matching* and is defined as follows.

Definition 3.1.3 (Partial Matching)
The partial matching $\mathcal{M}$ between two RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$ is a set of pairs $\mathcal{M} \subseteq V_1 \times V_2$. $\mathcal{M}$ describes a partial mapping between V_1 and V_2 with the following conditions:

1. $\forall (r, s) \in \mathcal{M} : \; S_1[r] = S_2[s]$ *(nucleotide condition)*
2. $\forall (r, s) \in \mathcal{M} : \; STRUCT_1(r) = STRUCT_2(s)$ *(structure condition)*

The first two conditions apply to single nucleotides only, but single bases can be further part of a base pair. This case is already given with both conditions, because for any two partial matchings $(r, s), (r', s') \in \mathcal{M}$ with $(r, r') \in B_1$ and $(s, s') \in B_2$ follows that $S_1[r] = S_2[s]$ and $S_1[r'] = S_2[s']$. Note that it is not implicated that always the whole bond has to be part of the matching. Here, we refer to one or a set of pairs $(r, s) \in \mathcal{M}$ as a single partial matching or some partial matchings, if we are in the context of two specific RNAs with their partial matching $\mathcal{M}$.

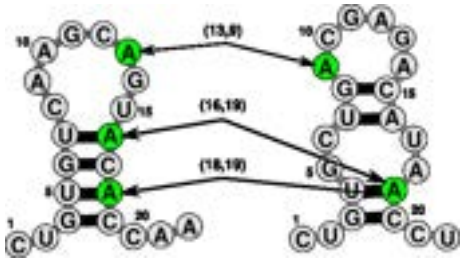

Figure 3.2: The figure shows three arbitrary partial matchings between the left and right RNA. The pairs $(13, 9)$, $(16, 19)$, $(18, 19)$ are elements from the set $\mathcal{M}$.

Now we want to combine these partial matchings in a way that ensures the same structural context in both secondary structures. Clearly, a pattern in the first RNA is matched with a pattern in the second RNA and vice versa. In our context, a match is always an exact match which means that the connected condition is not sufficient for a set of partial matchings. Further on, such a set have to guarantee the same sequential and structural context of the partial matchings, i.e. there is a backbone bond with a similar orientation or a hydrogen bond between two partial matchings. According to the original article [BS04], we make use of the *transition type* function:

$$\tau_{\mathcal{R}}(i,i') = \begin{cases} +1, & \text{if } i = i'+1 \\ -1, & \text{if } i = i'-1 \\ 0, & \text{if } (i,i') \in B \\ \text{undefined}, & \textit{otherwise} \end{cases} \tag{3.1}$$

for a given RNA $\mathcal{R} = (S, B)$ and two nucleotide positions i, i'. The three transition types describe the relative order of two adjacent sequence positions. The cases $+1$ and -1 denote two consecutive nucleotides in the sequence strand. The third case denotes two structural adjacent nucleotides in form of a base pair. According to the path and pattern definitions, two partial matchings with the same transition type in the respective RNA form a pattern and therefore a path as well. Analogously the definitions for one RNA, we first define a matching path and then a matching pattern.

Definition 3.1.4 (Matching Path)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$, $\mathcal{R}_2 = (S_2, B_2)$ and the partial matching $\mathcal{M}$ over $\mathcal{R}_1$ and $\mathcal{R}_2$. A matching path is a sequence of pairs $\langle (r_1, s_1), ..., (r_k, s_k) \rangle$ with $(r_i, s_i) \in \mathcal{M}$, $1 \leq i \leq k$, such that:

1. *$\langle r_1, ..., r_k \rangle$ is a path in $\mathcal{R}_1$,*
2. *$\langle s_1, ..., s_k \rangle$ is a path in $\mathcal{R}_2$,*
3. *the transition types are defined and equal: $\tau_{\mathcal{R}_1}(r_i, r_{i+1}) = \tau_{\mathcal{R}_2}(s_i, s_{i+1})$ for each $1 \leq i < k$.*

The definition of a matching pattern is now straightforward, as it is only necessary to ensure that for any two positions of the matching pattern the matching path is completely part of the pattern.

Definition 3.1.5 (Matching Pattern)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$, $\mathcal{R}_2 = (S_2, B_2)$ and the partial matching $\mathcal{M}$ over $\mathcal{R}_1$ and $\mathcal{R}_2$. A matching pattern of size k is a set of pairs $\mathcal{M}_{\mathcal{P}} \subseteq \mathcal{M}$ with $k = |\mathcal{M}_{\mathcal{P}}|$ such that for any two pairs $(r_i, s_i), (r_j, s_j) \in \mathcal{M}_{\mathcal{P}}$ there exists a matching path completely lying in $\mathcal{M}_{\mathcal{P}}$, i.e. $(r, s) \in \langle (r_1, s_1), \ldots, (r_k, s_k) \rangle \longleftrightarrow (r, s) \in \mathcal{M}_{\mathcal{P}}$

Obviously the sets $\{r \mid (r,s) \in \mathcal{M}_{\mathcal{P}}\} \subseteq V_1$ and $\{s \mid (r,s) \in \mathcal{M}_{\mathcal{P}}\} \subseteq V_2$ are patterns according to definition 3.1.2. Therefore we can call a matching pattern also connected matching. For later definitions and algorithmic usage is is necessary to verify that a matching pattern preserves the backbone order.

Proposition 3.1.2 (Backbone Order)
Let $\mathcal{M}_\mathcal{P}$ be a matching pattern over two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$, $\mathcal{R}_2 = (S_2, B_2)$. Further let $(r, s), (r', s') \in \mathcal{M}_\mathcal{P}$. Then it follows $r < r'$ if and only if $s < s'$.

Proof. Suppose two pairs $(r, s), (r', s')$ with $r > r'$ and $s < s'$. Further suppose two paths, one from r to r' in $\mathcal{R}_1$ and the other from s to s' in $\mathcal{R}_2$.

First, we suppose $(r, s), (r', s')$ are connected by one backbone bond which imply that they form a path completely lying in $\{(r, s), (r', s')\}$. As $r > r'$, the transition type is $\tau_{B_1} = +1$ on the path in $\mathcal{R}_1$. For the path in $\mathcal{R}_2$ the type is $\tau_{B_2} = -1$ as $s < s'$. But this contradicts the condition of equal transitions types.

Second, suppose $(r, s), (r', s')$ form a base pair, i.e $(r, r') \in B_1$ and $(s, s') \in B_2$. The transitions types are equal (both 0), but from $r > r'$ it follows that r is right paired and r' is left paired. For s and s' the types are vice versa. But that contradicts the condition of equal structure types for partial matchings.

Third, suppose equal structure types for both pairs, but $(r, s), (r', s')$ are not directly adjacent. Then there are at least two nucleotides on each path which are adjacent but violate either the condition for equal transition types or equal structure types according to first or second case. Note that there is no base pair on each path which violates the nested condition. □

Definition 3.1.5 is already capable to describe the highlighted substructure in the putative SECIS elements in figure 1.1. However, a matching pattern do not necessarily preserves bonds. Figure 3.3 a) shows a correct matching pattern, but the base pair from the stem in the left RNA is matched to different stems from a multi-loop in the right RNA. Such a mutational event could happen, but in general structural properties are conserved during evolution and therefore we want to prefer bond-preserving matchings. Note that backbone bonds are not necessarily preserved in a matching pattern as shown in figure 3.3 b). Here the matching pattern is connected via an alternative matching path.

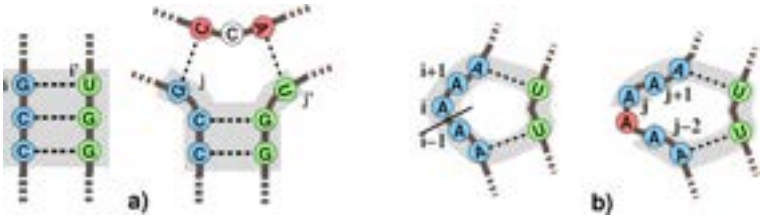

Figure 3.3: Matchings which do not preserve bonds. a) The hydrogen bond (i, i') is not preserved. This case is excluded for an exact pattern match. b) Here the backbone bond between i and $i-1$ is not preserved. This is unimportant for an exact pattern match. Figure according to [BS04].

As a second restriction we exclude all sub-optimal matching patterns, because no additional information is obtained from these matchings. Only matching patterns with the largest possible size are considered. We call such patterns *maximal extended*. This leads to the following definition for the final object for our later approaches.

Definition 3.1.6 (Exact Pattern Match)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$. *Further let* $\mathcal{M}_\mathcal{P} \subseteq \mathcal{P}_1 \times \mathcal{P}_2$ *a matching pattern of size* k *over two patterns* $\mathcal{P}_1 = \{r_1, ..., r_k\} \subseteq V_1$ *and* $\mathcal{P}_2 = \{s_1, ..., s_k\} \subseteq V_2$. *An exact pattern match* $\mathcal{E}$ *is defined as* $\mathcal{E} \subseteq \mathcal{M}_\mathcal{P}$, *such that:*

1. *for any two pairs* (r, s), $(r', s') \in \mathcal{E}$: $(r, r') \in B_1 \Leftrightarrow (s, s') \in B_2$ *(bond-preserving condition) and*
2. $\mathcal{E}$ *is maximally extended, i.e.* $\forall \mathcal{E}' : \mathcal{E} \subseteq \mathcal{E}' \Rightarrow \mathcal{E}' = \mathcal{E}$.

An example for an exact pattern match is given in figure 3.4. The maximal extension of that match can be verified as well. Every extension leads to a mismatch and every exclusion of one position from the match leads to an sub-optimal match.

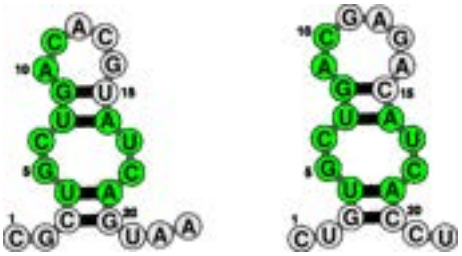

Figure 3.4: The green nucleotides denote a maximally extended exact pattern match between the left and the right RNA.

As the size of the patterns is $|\mathcal{P}_1| = |\mathcal{P}_2| = k$, we can denote the size of an exact pattern match with $|\mathcal{E}| = k$, as well. Note again that the minimal size of an exact pattern match is 2, as we are interested in relationships higher than single base matches. This follows also from the definition of a pattern. Note that we refer with the term *matching* to an exact pattern match.

The algorithm in section 3.4 follows the maximal extension condition and exclude all sub-patterns from the output set. Moreover this condition has some important consequences on the set of all exact pattern matches for two RNAs. See section 3.2 for details. Nevertheless, we are left with some cases with an unclear situation how to define a maximally extended match. See figure 3.11 b) on page 47 for such a case. The algorithm given in 3.4 solve these cases correct and returns only one maximally extended match for each matching pattern. See section 3.4 for details.

3.2 Properties of the Set of Exact Pattern Matches

An exact pattern match $\mathcal{E}$ describes an exact matching substructure between two RNAs. Usually, there are many EPMs between two RNAs. The algorithm in section 3.4 deals with this task and returns matchings for two RNAs according to definition 3.1.6. With an appropriate traceback it is possible to retrieve *all* exact pattern matches in $O(nm)$ time and $O(nm)$ space. As the discussed approaches in chapter 4 and 5 are based on this set, here some general properties are given.

Definition 3.2.1 (Set of Exact Pattern Matches)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$*. The set of all exact pattern matches* $\mathcal{E}$ *over* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *is defined as*

$$\mathbf{E}_{\gamma}^{1,2} = \left\{ \{\mathcal{E}\} \mid \mathcal{E} \textit{ is } \mathsf{EPM} \ \wedge \ |\mathcal{E}| \geq \gamma \right\}.$$

Assuming that smaller motives are less meaningful, γ denotes the lower bound for the size of the exact pattern matches included in $\mathbf{E}_{\gamma}^{1,2}$. This can be also useful for complexity reasons of the used comparison method. If no value is given, this refers to *all* possible exact pattern matches between the two RNAs. Clearly, the smallest possible size of an exact pattern match is 2.

Uniqueness of EPMs

Recall that any sub-matching pattern $\mathcal{E}' \subseteq \mathcal{E}$ of an exact pattern match $\mathcal{E}$ is not contained in $\mathbf{E}_{\gamma}^{1,2}$. The important implication is that any two exact pattern matches do not share the same partial matchings, i.e. for any two $\mathcal{E}_i, \mathcal{E}_j \in \mathbf{E}_{\gamma}^{1,2}$ follows $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$. With other words, any single exact matching base $(r, s) \in \mathcal{E}$ with $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$, $(r, s) \in \mathcal{M} \subseteq V_1 \times V_2$, is part of exactly one $\mathcal{E}$ and therefore this partial matching is unique in $\mathbf{E}_{\gamma}^{1,2}$.

Proposition 3.2.1 (Unique EPM)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$ *with their set of exact pattern matches* $\mathbf{E}_{\gamma}^{1,2}$*. Then it follows* $\forall \mathcal{E}_i, \mathcal{E}_j \in \mathbf{E}_{\gamma}^{1,2}$*,* $\mathcal{E}_i \neq \mathcal{E}_j \Longrightarrow \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$*.*

Suppose the case that two exact pattern matches contain the same exact partial matching. Then it follows that they are either identical or one of them is not maximally extended. Of course, two exact pattern matches can overlap, even in both RNAs. But this case imply that one exact pattern match has to match to another region of the other exact pattern match.

Maximal Number of EPM

From proposition 3.2.1 it is possible to determine the maximal size of $\mathbf{E}_{\gamma}^{1,2}$.

Proposition 3.2.2 (Maximal Number of EPMs)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$ *with their set of exact pattern matches* $\mathbf{E}_{\gamma}^{1,2}$*. Further the lengths of the sequences are given with* $|S_1| = n$*,* $|S_2| = m$*. Then then there are maximal* $(n \cdot m)$ *different exact pattern matches within* $\mathbf{E}_{\gamma}^{1,2}$*.*

There are maximal $V_1 \times V_2$ different combinations for a partial matching. Thus, the size of the set is $0 \leq |\mathbf{E}_{\gamma}^{1,2}| \leq (n \cdot m)$, as the set can be also empty. The size of each exact pattern match is limited as well. Obviously, the maximal size is bounded by the RNA with the minimal sequence length.

Library of EPMs

The set $\mathbf{E}_{\gamma}^{1,2}$ can be seen as a library of all similarities between two given RNAs. Finding a simple, but unique representation for each exact pattern match is straightforward from

above. According to proposition 3.2.1, a single partial matching $(r, s) \in \mathcal{E}$ is sufficient to identify the whole exact pattern match. Using always a specific partial matching out of the whole EPM, this can be advantageous in algorithmic usage. For example, we select always the partial matching with the maximal indices to simplify the exclusion of overlapping cases.

The following lemma summarizes the given properties of the set $\mathbf{E}_{\gamma}^{1,2}$.

Lemma 3.2.1 (Properties of a Set of Exact Pattern Matches)
Given two RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$ with their sequence lengths $|S_1| = n$ and $|S_2| = m$. Further $V_j = \{i \mid 1 \leq i \leq |S_j|\}$ denote the set of positions for RNA j. Providing there is a method to find exact pattern matches $\mathcal{E}$ over two RNAs, the properties of the set of discovered exact pattern matches, written as $\mathbf{E}_{\gamma}^{1,2}$, are summarized as follows:

1. *The set $\mathbf{E}_{\gamma}^{1,2}$ contains $|\mathbf{E}_{\gamma}^{1,2}| = k$ exact pattern matches over $\mathcal{R}_1$ and $\mathcal{R}_2$, with $0 \leq k \leq (n \cdot m)$.*
2. *The set $\mathbf{E}_{\gamma}^{1,2}$ comprises* all *possible exact pattern matches over $\mathcal{R}_1$ and $\mathcal{R}_2$.*
3. *Each exact pattern match $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$ is of size $2 \leq \gamma \leq |\mathcal{E}| \leq \mathsf{MIN}(n, m)$.*
4. *Any two exact pattern matches are disjoint: $\forall \mathcal{E}_i, \mathcal{E}_j \in \mathbf{E}_{\gamma}^{1,2} : \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$.*
   Similar is that for all $(r, s) \in \mathcal{M} \subseteq V_1 \times V_2$, there exists exactly one $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$ with $(r, s) \in \mathcal{E}$ or (r, s) is not part of any $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$.

Visualization of Exact Pattern Matches

Within the scope of this thesis, it has been a task to find a way for visualizing exact pattern matches as well as sets of exact pattern matches $\mathbf{E}_{\gamma}^{1,2}$. For applications in biology it is very helpful to have a more convenient representation of the underlying objects. With the above given properties a matrix representations seems predetermined. In bioinformatics and biology such dot plots are often used to visualize pairwise information.

Suppose a matrix with the positions of sequence S_1 at the x-axis and the positions of S_2 at the y-axis. According to point 4 from lemma 3.2.1 above, it is possible to mark all exact pattern matches for the given RNAs in a plain way, i.e. there are no overlapping entries in this matrix. Here we just give a short example of such a dot plot of exact pattern matches.

Figure 3.5 below shows the set $\mathbf{E}_{\gamma}^{1,2}$ for two Hepatitis C virus IRES RNAs (see section 6.3 for details). Each exact pattern match is indicated in a different color. All dots belonging to the same exact pattern match are either diagonal adjacent or connected with a small additional line. This line indicates further that a base pair is part of the exact pattern match. Such an illustration is also helpful to visualize compatible arrangements of exact pattern matches. See section 3.3 for details.

The shown dot-plot in figure 3.5 is the output of a Java program which was written during this thesis. The input file is a structured XML file with all data about the exact pattern matches. This file can be generated with our implementation for the algorithms of chapter 4 and 5.

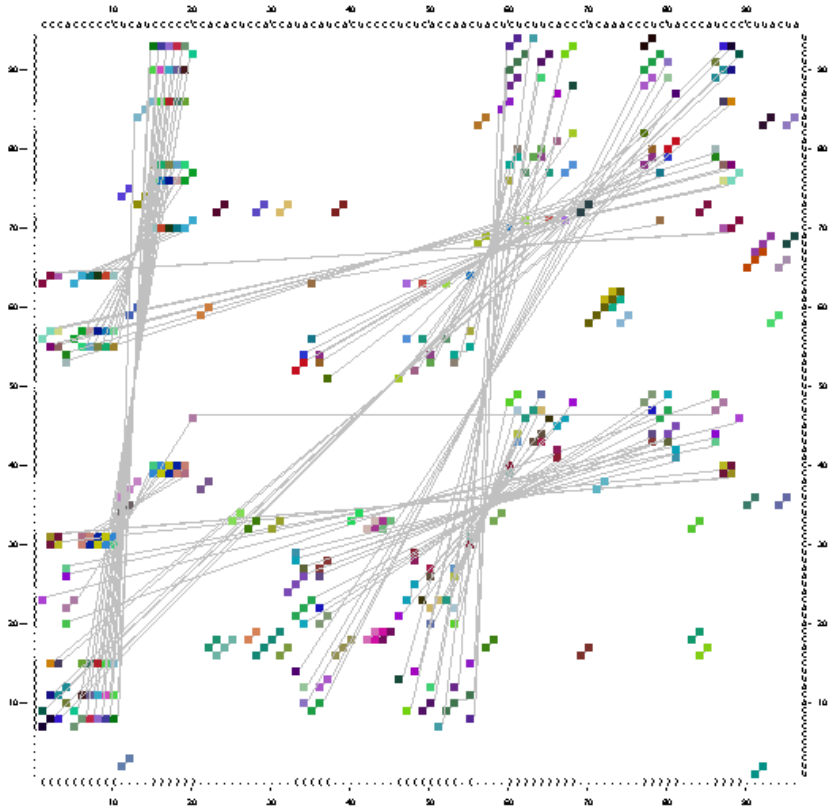

Figure 3.5: A dot-plot of exact pattern matches for two Hepatitis C virus IRES RNAs (bases 1-96). Each EPM has a different color. Grey lines indicate that a base pair is part of the EPM.

3.3 Structural Definitions on Exact Pattern Matches

The exact pattern matches included in $\mathbf{E}_{\gamma}^{1,2}$ differ in their size and shape as well as in their structural positions in both RNA. Considering for example the maximum common substructure, i.e. the EPM with the largest size, or a special found EPM like the indicated substructure in figure 1.1, this can be sufficient information from an analytic or biologically point of view. Considering two or several exact pattern matches, then they probably overlap or cross each other as well as they can be "near" in the first RNA, but "far" away in the other structure.

The main goal of the approach treated in this thesis is to identify sets of exact pattern matches for two RNAs which can be used for a pairwise comparison. This means, such a set should exclude overlapping and crossing patterns in general. For example, see figure C.1 in the appendix which was taken from the article for the `MCS` algorithm. The five highlighted EPMs satisfy this condition. If one follows the backbone in both RNAs, the EPMs appear in the same order. For a comparative analysis it is obviously important, if two or more substructures occur in such a way. Suppose for example a motif which needs some correct arranged substructures for its working. Therefore we also speak about *arrangements* of exact pattern matches. In the following we give basic definitions for exact pattern matches in their structural context.

Non-Crossing

In order to maintain the structural order of exact pattern matches in two RNAs, we define an invariant condition for any arrangement of exact pattern matches. Consequently, we are able to check if two EPMs are compatible or not. Suppose a set $\mathbf{E}_{\gamma}^{1,2}$ of EPMs as shown in figure 3.6 below. For example, a good subset comprises the exact pattern matches $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$, but we want to exclude $\mathcal{E}_5$ and $\mathcal{E}_6$. The match $\mathcal{E}_5$ is crossing $\mathcal{E}_2$ and $\mathcal{E}_3$ whereas $\mathcal{E}_6$ is overlapping with $\mathcal{E}_3$ in $\mathcal{R}_1$ and with $\mathcal{E}_4$ in $\mathcal{R}_2$. Note that not all possible EPMs are indicated in the figure.

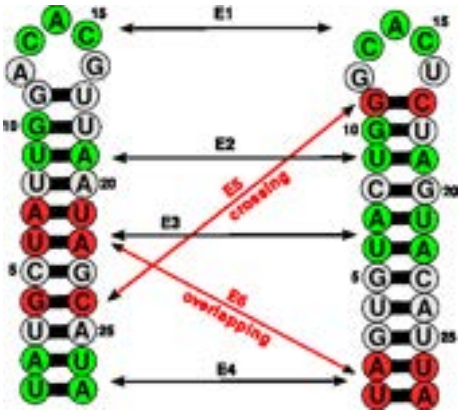

Figure 3.6: A set of possible exact pattern matches between the left RNA ($\mathcal{R}_1$) and the right RNA ($\mathcal{R}_2$). The set $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$ can be used for a comparison, whereas $\{\mathcal{E}_5, \mathcal{E}_6\}$ should be excluded. $\mathcal{E}_5$ crosses $\mathcal{E}_2$ and $\mathcal{E}_3$. $\mathcal{E}_6$ is overlapping with $\mathcal{E}_3$ in $\mathcal{R}_1$ and with $\mathcal{E}_4$ in $\mathcal{R}_2$.

From the viewpoint of a fixed EPM, we distinguish the following relative structural orderings. According to definition 2.1.4, any two base pairs $(i, i'), (j, j') \in B$ are either nested or independent. Following the nesting condition $i < j < j' < i'$, we denote a base pair (j, j') as *inside* of base pair (i, i') and analogously is (i, i') *outside* of (j, j'). Further, any nucleotide or base pair within $\langle 1, .., i-1 \rangle$ is located *before* base pair (i, i') and similarly the positions $\langle i'+1, ..., |S| \rangle$ are located *after* base pair (i, i').

Now we apply these orderings to EPMs. This is possible, because an exact pattern match is connected. See figure 3.7 below for an illustration of these cases. Consequently, we define two EPMs as structural compatible, if they preserve the structural ordering for any two positions in both RNAs. We denote structural compatible matchings as Non-Crossing and they are defined as follows.

Definition 3.3.1 (Non-Crossing)
Given two RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$ and further two exact pattern matches $\mathcal{E}_1$, $\mathcal{E}_2$ over $\mathcal{R}_1$, $\mathcal{R}_2$. Two exact pattern matches $\mathcal{E}_1$, $\mathcal{E}_2$ are Non-Crossing *if either*

1. $\forall (r_i, s_i) \in \mathcal{E}_1$,
   $\forall (r_j, s_j) \in \mathcal{E}_2$ *with* $r_i < r_j$: $s_i < s_j$ *($\mathcal{E}_1$ before $\mathcal{E}_2$) or*
2. $\forall (r_i, s_i), (r_{i'}, s_{i'}) \in \mathcal{E}_1$,
   $\forall (r_j, s_j) \in \mathcal{E}_2$ *with* $r_i < r_j < r_{i'}$: $s_i < s_j < s_{i'}$ *($\mathcal{E}_1$ outside $\mathcal{E}_2$).*

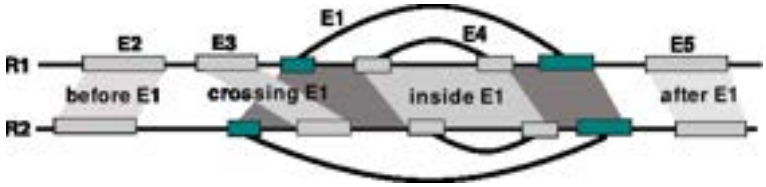

Figure 3.7: Ordering of exact pattern matches relative to EPM $\mathcal{E}_1$. The cases *before, inside* and *after* do not violate the Non-Crossing condition. Only EPM $\mathcal{E}_3$ crosses $\mathcal{E}_1$. Note that an arc denote a connected matching via base pairs.

Note that the orderings *after* and *inside* are symmetric cases of the given ones. Further we define the following short forms for the structural ordering of two exact pattern matches over two RNAs:

$$\begin{aligned} \mathcal{E}_1 \text{ crossing } \mathcal{E}_2 &: \mathcal{E}_1 \between \mathcal{E}_2 \\ \mathcal{E}_1 \text{ before } \mathcal{E}_2 &: \mathcal{E}_1 \Vdash \mathcal{E}_2 \\ \mathcal{E}_1 \text{ outside } \mathcal{E}_2 &: \mathcal{E}_1 \Subset \mathcal{E}_2 \end{aligned} \tag{3.2}$$

The Non-Crossing condition is only satisfied, if an exact pattern match is located completely before, after or inside the other connected matchings. Therefore it is not necessary to treat Non-Overlapping in an extra condition, because it is already included in Non-Crossing. Note that definition 3.3.1 is given in a general form. From figure 3.7 it is evident that checking the start and end from consecutive nucleotide positions is sufficient for verifying Non-Crossing.

With proposition 3.1.2 we have already shown, that a single EPM preserves the backbone order. According to other pairwise comparison methods like the general edit distance approach or the LAPCS problem, we want to find a subset of EPMs which is a plain mapping as well as an arc-preserving subsequence.

Proposition 3.3.1 (Non-Crossing Preserves Backbone Order)
Given two RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$ *and two* Non-Crossing *exact pattern matches* $\mathcal{E}_1, \mathcal{E}_2$ *from a set* $\mathbf{E}_\gamma^{1,2}$ *over* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *. Further let* $(r_1, s_1), (r_1', s_1') \in \mathcal{E}_1$ *and* $(r_2, s_2), (r_2', s_2') \in \mathcal{E}_2$*. Then it follows either (1)* $r_1 < r_2$ *if and only if* $s_1 < s_2$ *or (2)* $r_1 > r_2$ *if and only if* $s_1 > s_2$ *or (3)* $r_1 < r_2 < r_1'$ *if and only if* $s_1 < s_2 < s_1'$ *or (4)* $r_2 < r_1 < r_2'$ *if and only if* $s_2 < s_1 < s_2'$*, i.e.* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *preserve the backbone order in* S_1 *and* S_2*.*

Clearly, two Non-Crossing EPMs preserve the backbone order as well. This is achieved in definition 3.3.1 with the similar nucleotide orderings (before/outside) for both EPMs in both RNAs. This can be also verified in figure 3.7 above.

Matching Bounds

Each exact pattern match is embedded into the secondary structure at an specific point. Clearly, there exist nucleotide positions which limit the substructure from the structure around. For example, if an exact pattern match is part of a multi-loop, then it can be part of different stems. Here we give some definitions and notions to describe these boundaries.

First, we focus on patterns according to definition 3.1.2, i.e. patterns in a single RNA. Writing the nucleotide positions of a pattern as an increasing sequence, there exists a minimum and maximum position. In the view of the secondary structure, these two position determine the outside borders of the pattern. Therefore we call them *outside-bounds*. In the view of an arc-annotated sequence, we denote the minimum as *left* bound and the maximum as *right* bound.

If a pattern contains a base pair, the structural shape is more complex and the outside-bounds are not sufficient to describe all structural borders of a pattern. Suppose a base pair $(i, j) \in B$ within a pattern. Then the pattern not necessarily contains all nucleotides from $S[i+1]$ to $S[j-1]$. With other words, there exist two positions (i', j') with $i \leq i' < j' \leq j$ that form an additional structural border, lying *inside* the range of the outside-bounds. Clearly, if a pattern contains several independent base pairs, there can be several inside borders. The set of all such borders is called *inside-bounds*. The following definition summarizes all bounds for a pattern.

Definition 3.3.2 (Pattern Bounds)
Let $\mathcal{R} = (S, B)$ a given RNA and $\mathcal{P}$ be a pattern of size k. Further let $\mathcal{P} = \langle p_1, p_2, ..., p_k \rangle$ be an increasing sequence of positions of $\mathcal{P}$ for that hold: $\forall\, i, j : i < j \Leftrightarrow p_i < p_j$, then the following bounds are defined:

$$
\begin{array}{rcccl}
\textit{outside-bound-left} & : & \mathsf{LEFT} & = & p_1 \\
\textit{outside-bound-right} & : & \mathsf{RIGHT} & = & p_k \\
\textit{outside-bounds} & : & \mathsf{OUT} & = & (p_1, p_k) \\
\textit{inside-bounds} & : & \mathsf{IN} & = & \{\, (p_i, p_{i+1}) \mid p_{i+1} > p_i + 1 \}
\end{array}
$$

Note again that *outside-bounds* always exists, whereas the set *inside-bounds* can be empty. If a pattern comprises only unbound nucleotides or a complete hairpin inclusive the closing bond, this results in a complete consecutive sequence. In contrast, an inside-bound (p_i, p_{i+1}) represent two non consecutive positions of a pattern. All nucleotides between $S[p_i + 1]$ and $S[p_{i+1} - 1]$ are not part of the pattern. If a pattern consists of only one base pair, then inside and outside bounds are identical.

Next, we apply these notions for a pattern in one RNA to an exact pattern match over two RNAs. We have shown in section 3.1 that a single EPM preserves the backbone order. This implies an ordering on the included partial matchings, i.e. $\forall (r, s), (r', s') \in \mathcal{E} : r < r' \Leftrightarrow s < s'$. Now we can define the matching bounds for an exact pattern match as follows.

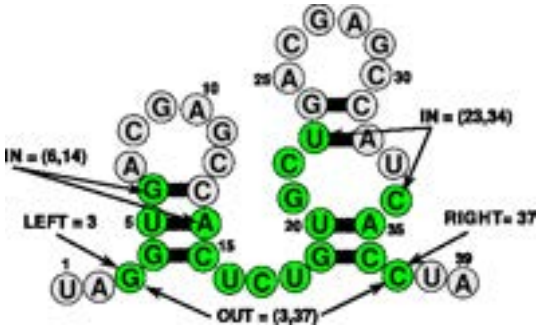

Figure 3.8: The green nucleotides denote a pattern of an exact pattern match in one RNA. The indicated positions represent the different pattern bounds.

Definition 3.3.3 (Matching Bounds)
Given an exact pattern match $\mathcal{E}$ of size k over two RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$ with its corresponding pattern positions $\mathcal{P}_1$ in $\mathcal{R}_1$ and $\mathcal{P}_2$ in $\mathcal{R}_2$. Further the two patterns are given with their increasing sequences. For pattern $\mathcal{P}_1$ the sequence is $\langle r_1, r_2, ..., r_k \rangle$, $\forall r_i, r_j \in \mathcal{P}_1, \forall\, i, j : i < j \Leftrightarrow r_i < r_j$ and for pattern $\mathcal{P}_2$ the sequence $\langle s_1, s_2, ..., s_k \rangle$ is defined similarly. Then we define the following matching bounds:

$$
\begin{array}{rcll}
\textit{outside-bounds-left} & : & \mathsf{LEFT}_{\mathcal{E}} & = & (r_1, s_1) \\
\textit{outside-bounds-right} & : & \mathsf{RIGHT}_{\mathcal{E}} & = & (r_k, s_k) \\
\textit{outside-bounds} & : & \mathsf{OUT}_{\mathcal{E}} & = & \big\langle (r_1, r_k), (s_1, s_k) \big\rangle \\
\textit{inside-bounds} & : & \mathsf{IN}_{\mathcal{E}} & = & \big\{ \big\langle (r_i, r_{i+1}), (s_j, s_{j+1}) \big\rangle \mid \\
 & & & & \quad r_{i+1} > r_i + 1 \Leftrightarrow s_{j+1} > s_j + 1 \big\}
\end{array}
$$

Note the slight difference to the pattern bounds. Each element in $\mathsf{IN}_{\mathcal{E}}$ represents a loop excluded in $\mathcal{R}_1$ and $\mathcal{R}_2$ if and only if in *both* patterns two positions are not consecutive. An EPM can be consecutive in one RNA, but not in the other RNA. For example, imagine two hairpins of different size and an exact pattern match between these hairpins. One pattern comprise the whole hairpin, whereas the other comprise only a part. In this case the set $\mathsf{IN}_{\mathcal{E}}$ is empty.

Further we define for all different matching bounds a notion to retrieve the bounds from a single RNA. For example, we denote the matching bounds for RNA $\mathcal{R}_1$ with $\mathsf{LEFT}^1_{\mathcal{E}} = r_1$, $\mathsf{RIGHT}^1_{\mathcal{E}} = r_k$ and $\mathsf{OUT}^1_{\mathcal{E}} = (r_1, r_k)$. For clarity we give the definition for the inside-bounds. Suppose the two pattern sequences $\langle r_1, r_2, ..., r_k \rangle$ and $\langle s_1, s_2, ..., s_k \rangle$ given as above. Then we define for RNA $\mathcal{R}_1$:

$$
\begin{aligned}
\mathsf{IN}^1_{\mathcal{E}} = \{ (r_i, r_{i+1}) \mid \exists \, \big\langle (r_i, r_{i+1}), (s_j, s_{j+1}) \big\rangle \in \mathsf{IN}_{\mathcal{E}} : \\
r_{i+1} > r_i + 1 \Leftrightarrow s_{j+1} > s_j + 1 \}
\end{aligned}
\tag{3.3}
$$

Matching Closure

In the following we introduce our notion of a matching closure. A closure is depicted by filling up the two corresponding patterns of an EPM with all positions from base pair partners not included in the original matching.

Definition 3.3.4 (Matching Closure)
Given an exact pattern match $\mathcal{E}$ over two RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$ with the corresponding pattern positions $\mathcal{P}_1$ in $\mathcal{R}_1$ and $\mathcal{P}_2$ in $\mathcal{R}_2$. Then we define the matching closure $\overline{\mathcal{E}}$ as:

$$\overline{\mathcal{E}} = \mathcal{E} \cup \Big\{ (r', s') \mid \exists r \in \mathcal{P}_1, \exists s \in \mathcal{P}_2 : (r, r') \in B_1 \Leftrightarrow (s, s') \in B_2 \vee (r', r) \in B_1 \Leftrightarrow (s', s) \in B_2 \Big\}$$

Due to the maximal extension condition of an EPM, the additional pairs of nucleotide positions do not represent partial matchings. This case occur for exact pattern matches with different base pair partners in the different structures, i.e. there is a non-canonical base pair like G-U in one structure and a standard pair in the second. Otherwise the pattern is not maximally extended. Figure 3.9 shows an example of a matching closure.

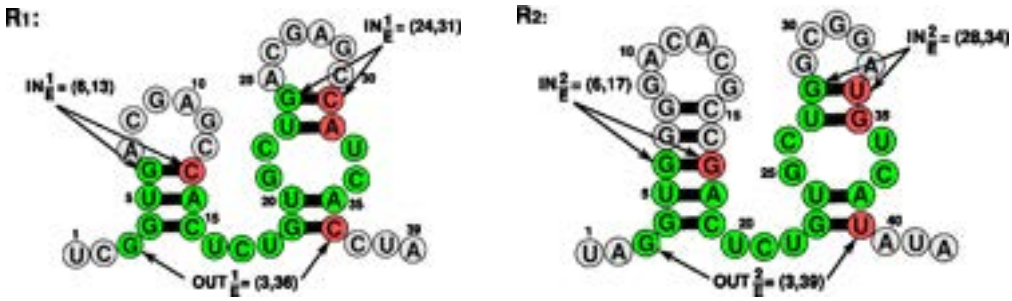

Figure 3.9: Additional nucleotides in a matching closure. Suppose the green nucleotides denote an exact pattern match. Then the red-marked nucleotide positions are additional pairs in $\overline{\mathcal{E}}$. The given matching bounds are $\mathsf{IN}_{\overline{\mathcal{E}}} = \{\langle (6,13),(6,17)\rangle, \langle (24,31),(28,34)\rangle\}$ and $\mathsf{OUT}_{\overline{\mathcal{E}}} = \langle (3,36),(3,39)\rangle$.

The clustering approach in chapter 5 uses always a matching closure $\overline{\mathcal{E}}$ instead of the original $\mathcal{E}$ to determine the matching bounds. Due to the additional nucleotide positions, we can rewrite the patterns from a matching closure $\overline{\mathcal{E}}$ as $\overline{\mathcal{P}}_1 = \{r_1, r_2, ..., r_{k'}\}$ and $\overline{\mathcal{P}}_2 = \{s_1, s_2, ..., s_{k'}\}$ with $k' \geq k$. The resulting matching bounds $\mathsf{IN}_{\overline{\mathcal{E}}}$, $\mathsf{OUT}_{\overline{\mathcal{E}}}$, $\mathsf{LEFT}_{\overline{\mathcal{E}}}$ and $\mathsf{RIGHT}_{\overline{\mathcal{E}}}$ are now determined via the increasing sequences based on $\overline{\mathcal{P}}_1$ and $\overline{\mathcal{P}}_2$. We omit an extra definition for that case.

3.4 A Fast Method to Detect Exact Pattern Matches

In this section we review the fast algorithm from Siebert and Backofen [SB07] to obtain the set $\mathbf{E}_{\gamma}^{1,2}$ of all maximally extended and bond-preserving substructures between two nested RNA secondary structures. The developed dynamic programming algorithm detects these substructures in form of exact pattern matches in $O(nm)$ time and $O(nm)$ space. In this

thesis we usually refer to this approach as the maximum common substructure (MCS) algorithm. In the following we describe the main algorithmic steps of the MCS algorithm according to the notions from Siebert and Backofen [SB07].

Basic Notions for the Algorithm

The input of the MCS algorithm consists of two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_1 = (S_2, B_2)$ with their lengths $|S_1| = n$ and $|S_2| = m$. In order to find an exact pattern match for each combination of nucleotide position i from the first RNA and j from the second RNA, the key idea of the algorithm is to maintain three $n \times m$ matrices M^{eb}, M^{nb} and M^{loop}. These matrices correspond to the different cases of matchings: M^{eb} handles complete base-pair matchings, M^{nb} handles matchings of only left or right paired nucleotides and M^{loop} handles matchings of inner loops.

Inner loops are all loops which are enclosed by a base pair. All inlying nucleotides of an inner loop are simply numbered consecutively. The resulting sequence of positions for an inner loop enclosed by a base pair $(i, i') \in B$ is given as $\langle l_1, l_2, \ldots, l_{size} \rangle$ and is called *loop-walk*. This scheme is applied to all loops like hairpins, bulges, internal loops and multi-loops. Figure 3.10 shows the nucleotide numbering. The global position of a loop position is accessed via the function $pos(l_i)$. For example, $pos(l_{size}) = i' - 1$, if $(i, i') \in B$ encloses a loop from $l_1, \ldots, l_{size}$.

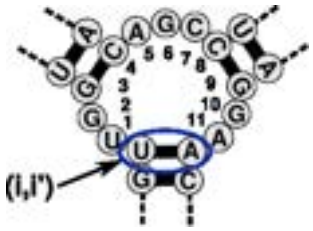

Figure 3.10: Numbering of nucleotide position for an inner loop. The figure shows an inner loop with $l_{size} = 11$. The loop is enclosed by base pair (i, i').

The algorithm works from inside to outside. Suppose an internal loop closed by a base pair (i, i'). Then all base pairs (j, j') that hold $i < j < j' < i'$ are treated before (i, i'). We start with a description of the subfunctions which handle the loop-walk, base-pair matchings and none base-pair matchings. For the loop-walk an additional function is needed which combines the matrices M^{eb} and M^{nb} and returns the size of a maximally extended pattern found so far for a inner loop position in the first and second RNA. This function is called `max-matching`. **Function** `loop-walking`(i, j)

This function is called from the main function with `LoopWalking`(i, j) to determine exact pattern matches in inner loops. Here i and j correspond to positions from the first and second RNA. The pseudocode is given in algorithm C.1 on the appendix.

The function `Loop-walking` iterates through all combinations of inner loop positions. The necessary values $l_{i_{size}}$ and $l_{j_{size}}$ for the loop sizes can be determined in advance. First, the while loop from line 8 to line 11 determines the number of matching inner loop nucleotides for a pair (k, l) of loop positions clockwise. The index r denotes this

number and the maximally extended matching for a pair (k, l) of loop positions is then determined with a call to the function `max-matching`(i, j, r). Note that each pair (k, l) is only considered once (line 7). This means that if the size of the matching nucleotide array is at least 2, then the value of M^{loop} need not be recomputed. This information can be stored in a binary $n \times m$ matrix.

Auxiliary Function `max-matching`(i, j, r)

This function is invoked with `max-matching`(i, j, r) from the `loop-walking` function, whereby i and j denote inner loop positions from the first and second RNA. The number of matching loop positions is given with r. `max-matching` returns the size of the new common substructure found up to the given positions. The pseudo code for `max-matching` is given in algorithm C.2 in the appendix.

This function operates on loop positions as well. The size of the current substructure is found with the help of M^{eb} and M^{nb}. The necessary entries are already determined due to the inside to outside scheme. Starting in line 3, the function checks whether the nucleotides are equal. If not, the current size is returned. If yes, both nucleotides have to be tested for a equal structure type (line 4: single stranded, line 6: left paired, line 11: right paired). Clearly, if both nucleotides are unpaired, the size is increased with one. If also both nucleotides with their base pair partner are equal, the size for this base-pair match is found in M^{eb}. If the base-pair partner are not equal, the size is found in M^{nb}. This implies that the matching cannot be further extended and the current size is returned.

Function `base-pair-match`(i, j)

This function determines the value M^{eb}, i.e. a base pair $(i, i') \in B_1$ in the first RNA is matched with a base pair $(j, j') \in B_2$ in the second RNA. The function have to distinguish two cases, because the nucleotides of a matching path in the inner loop to the right can overlap or not with the nucleotides of a matching path to the left. Both cases are shown in figure 3.11 below. The pseudocode is omitted.

Case 1: The matching paths to the right of i and j do not overlap with the matching paths to the left of i' and j'. Then the entry for the matching base pair is $M^{eb}(i, j) = M^{loop}(i + 1, j + 1) + M^{loop}(i'_r, j'_r) + 2$, i.e. the length of the left matching path plus the length of the right matching path plus the matching base pair. The ends of the right matching paths are denoted with i'_r and j'_r. See figure 3.11 a) for this case. After $M^{eb}(i, j)$ is computed, $M^{loop}(i+1, j+1)$ and $M^{loop}(i'_r, j'_r)$ are set to 0, because these values are now included in $M^{eb}(i, j)$. This prevents the output of not maximally extended matchings.

Case 2: In this case the matching path to the left and to the right overlap. This case is shown in figure 3.11 b). The two "A"s before the cutting line in the left RNA can be part of a matching path to the left or to the right in the right RNA. This is the only situation where the assignment of an exact pattern match becomes ambiguous. To avoid an exponential number of solutions, only one maximal extended matching can be considered. This can be easily extracted by looking at a subarray which contains the overlapping nucleotides. For this array, the algorithm add and subtract the corresponding `max-matching` values. Then the cut which provides a maximum value is chosen, whereby

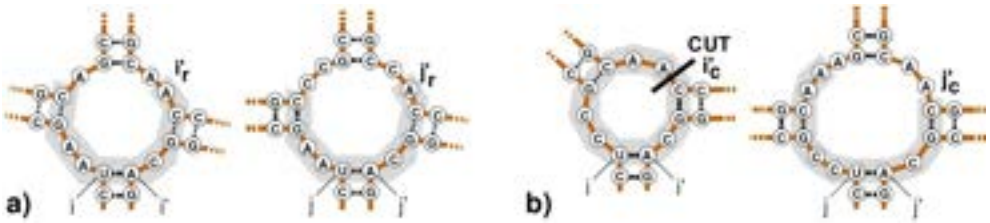

Figure 3.11: Two cases for a base-pair matching (i, i') with (j, j'). a) The matching path to the left and to the right from (i, i') and (j, j') do not overlap (case 1). b) Here the matching path overlap in the left RNA (case 2). The cut determines a single maximally extended exact pattern match for the matching base pairs. Figure taken from [BS04]

i'_c and j'_c denote the positions directly after the cut. Note that base pairs are preserved in this step. The value $M^{eb}(i, j)$ is similar to case 1 and also the M^{loop} entries are set to 0 afterwards.

Function `none-base-pair`(i, j)

This function handles the case if a left paired or a right paired nucleotide in the first RNA is matched to a nucleotide of the same structure type, but the base pair partners are different. Consequently, an overlapping of such nucleotides is not possible and each entry of M^{nb} can be computed independently. The algorithmic code is omitted, but the structure is simply. The function is called only on left-paired nucleotides and checks either i and j are matching left-paired nucleotides or i' and j' are matching right-paired nucleotides. If this succeeds, the entry $M^{nb}(i, j)$ is computed from the corresponding M^{loop} entries.

Main Function, Traceback and Complexity

The main function determines exact pattern matches from inner to outer loop. This is achieved with a list of base pairs $(i, bp(i))$ from the first RNA and base pairs $(j, bp(j))$ from the second RNA ordered from inside to outside. For each loop, the function `loop-walking`$(i + 1, j + 1)$ is called. The functions `base-pair-match`(i, j) and `none-base-pair`(i, j) are called according to the current matching cases. Note that a virtual base pair is assumed which closes the external loop $(1, |S_1|)$ and $(1, |S_2|)$.

The sizes of all exact pattern matched are stored in M^{loop}. With a traceback through this matrix according the given matching cases, all matchings can be extracted. If also the cut positions are stored, the traceback is possible in linear time for each exact pattern match.

The time complexity is given with $O(nm)$, because for each (i, j) combination the `loop-walking` function is executed. All subfunctions do not increase the time complexity, because inner loops do not overlap each other in one RNA. Further, all considered nucleotides are marked by the function `max-matching`. Therefore, each combination of inner loop positions is considered almost twice. The function `base-pair-match` works also on inner loop positions. Finding the cutting line costs not more than the size of the inner

loop. The function `none-base-pair` has only the cost of finding the matching path to the right of the right base-pair partner.

The space complexity is $O(nm)$, because the necessary marices M^{eb}, M^{nb} and M^{loop} are of size $n \times m$. Additional matrices for the storage of the cut positions as well as to save already considered nucleotides do not exceeds the given space complexity.

Results

According to the article [SB07], the `MCS` algorithm was performed in a first test on two Hepatitis C virus IRES sites. The GenBank codes are AF165050 and D45172. The optimal secondary structures have been determined with `RNAfold` [HFS+94]. Figure C.1 on page 109 in the appendix shows both RNAs. The five largest exact pattern matches are highlighted. We have tested our methods we the same RNAs. For example, the LCS-ERP approach is able to combine these five exact pattern matches in one solution. For example, see figure 6.1 in chapter 6 as well as section 6.3.1.

In a second test, the `MCS` algorithm was performed on putative SECIS elements in non-coding regions of Methanococcus jannaschii. Figure 1.1 shows a strongly conserved region in different RNAs found with the `MCS` algorithm by pairwise comparison. The RNAs for the putative SECIS elements were chosen according to [WSPB97].

Chapter 4

The Longest Common Subsequence of Exact RNA Patterns

The problems LCS and LAPCS analyse pairwise similarities of RNAs at a sequence and sequence-structure level, respectively. In this chapter, we present a related, but new problem of finding the **Longest Common Subsequence of Exact RNA Patterns** (LCS-ERP). It respects the secondary structure in form of exact matchings obtained by the approach from Siebert and Backofen [SB07] given in the last chapter. In section 4.1, we give a formal description of LCS-ERP. In section 4.2 we develop a dynamic programming algorithm solving the LCS-ERP problem. This problem is solvable in $O(n^2m^2)$ time and $O(nm)$ space (section 4.3), in contrast to the LAPCS problem which is in general NP-hard. However, the obtained arc-preserving subsequence for the LCS-ERP problem is based exclusively on exact pattern matches, which excludes single nucleotide matchings.

4.1 Problem Description for LCS-ERP

Recall from the last chapter, the set $\mathbf{E}_{\gamma}^{1,2}$ contains all exact pattern matches for two RNAs and can be computed in $O(nm)$ time and $O(nm)$ space [SB07]. Relating to the LCS and LAPCS problems, we introduce the problem **Longest Common Subsequence of Exact RNA Patterns** (LCS-ERP).

The formulation of LCS-ERP is motivated by the fact that different RNA secondary structures share the same complex sequence-structure patterns. For example, the SECIS motif shown in figure 2.5 on page 19 includes two necessary substructures. The two consecutive unbound alanine nucleotides are separated by helix II from the quartet substructure. Another example is the comparison of experimentally verified secondary structures. For biology, it would be interesting to know a "common core" of identical substructures in two rRNAs to identify relationships between different species. Such an example is shown in figure 6.5 on page 84.

Here, we are interested in the *maximal* arrangement of substructures shared by two RNAs. If the motives are given in the form of exact pattern matches according to definition 3.1.6, we call this the LCS-ERP problem. Note that we make use of the properties of the set $\mathbf{E}_{\gamma}^{1,2}$ of all exact pattern matches given in lemma 3.2.1 as well as the fact that a single exact pattern match preserves the backbone order.

Definition 4.1.1 (Longest Common Subsequence of Exact RNA Patterns)
Given two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$ *and a set of exact pattern matches* $\mathbf{E}_\gamma^{1,2}$ *over these two RNAs. LCS-ERP is the problem of finding the longest common subsequence of* S_1 *and* S_2 *which preserves the exact pattern matches in* $\mathbf{E}_\gamma^{1,2}$*; i.e. finding a mapping* $\mathcal{M}_{\mathsf{ERP}} \subseteq \{1, ..., |S_1|\} \times \{1, ..., |S_2|\}$ *of maximal length such that:*

1. *for each pair* $(r, s) \in \mathcal{M}_{\mathsf{ERP}}$ *there exists one exact pattern match in* $\mathbf{E}_\gamma^{1,2}$*:* $\forall (r, s) \in \mathcal{M}_{\mathsf{ERP}} : \exists \mathcal{E} \in \mathbf{E}_\gamma^{1,2}$ *with* $(r, s) \in \mathcal{E}$ *and* $\mathcal{E} \subseteq \mathcal{M}_{\mathsf{ERP}}$
2. $\mathcal{M}_{\mathsf{ERP}}$ *is a bijective mapping and preserves the order of the nucleotides:* $\forall (r, s), (r', s') \in \mathcal{M}_{\mathsf{ERP}} : r = r' \iff s = s',\ r < r' \iff s < s'$

The definition follows the LAPCS problem given in definition 2.3.6 on page 25. In condition one we claim that for each pair $(r, s) \in \mathcal{M}_{\mathsf{ERP}}$, i.e. for any exact matching nucleotide, there exists one exact pattern match in $\mathbf{E}_\gamma^{1,2}$. In addition, condition one includes that the complete EPM is part of $\mathcal{M}_{\mathsf{ERP}}$. The second condition ensures that the found subsequence is a common subsequence, i.e. a sequence which preserves the backbone order. Arcs or base pairs are induced by the EPMs itself. In contrast to the LAPCS problem, an isolated nucleotide is not part of any solution for LCS-ERP due to the fact that the set $\mathbf{E}_\gamma^{1,2}$ contains only exact matchings with comprise at least two connected nucleotides.

With the idea that each EPM has a size according to the matched bases, we search for a subset of maximal size out of $\mathbf{E}_\gamma^{1,2}$.

Proposition 4.1.1 (Maximal Subset)
A subset $\mathbf{E}_{\mathsf{LCS}}^{1,2} =\subseteq \mathbf{E}_\gamma^{1,2}$ *of* Non-Crossing *exact pattern matches with maximal size* $s = \sum_{\mathcal{E} \in \mathbf{E}_{\mathsf{LCS}}^{1,2}} |\mathcal{E}|$ *yields a solution for LCS-ERP.*

Proof. The proof is straightforward, because each pair $(r, s) \in \mathcal{M}_{\mathsf{ERP}}$ can be seen as partial matching from $\mathcal{M}$ and all EPMs are defined on these partial matchings. Further, each EPM itself preserves the backbone order and any two EPMs preserve the backbone order via the Non-Crossing condition. This holds for any induced base pairs as well. Any two EPMs with base pairs are either nested (inside/outside condition) or independent (before/after condition) by the Non-Crossing demand. Thus, a maximal subset is a longest common subsequence of exact RNA patterns. □

Recall from section 3.2 that we operate on a limited set of only $n \cdot m$ possible matchings in form of exact pattern matches. Further, there is also only a limited number of matchings which fulfil the Non-Crossing condition for each matching, both for EPMs with base pairs and for EPMs without base pairs. This reduces the overall complexity and enables a polynomial-time algorithm.

In the next section we propose an algorithm to solve LCS-ERP in polynomial time.

4.2 Dynamic Programming Algorithm for LCS-ERP

Here, we develop a dynamic programming algorithm solving the LCS-ERP problem in $O(n^2m^2)$ time and $O(nm)$ space. For the rest of this section we consider two nested RNAs given as $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. With the notion $S_i[m, n]$ we refer to a substring from $S_i[m]$ to $S_i[n]$ which denotes a consecutive part out of the complete sequence S_i. Further we denote the set of nucleotide positions for a sequence S_j with $V_j = \{\ i \mid 1 \leq i \leq |S_j|\}$.

The proposed algorithm combines ideas from sequence and sequence-structure comparison methods. Sequence alignment methods such as the Needleman-Wunsch recursion scheme shown in formula 2.1 uses a two-dimensional matrix to compute the optimal alignment. The incorporation of the secondary structure requires a comparison of substructures, i.e. of the subsequences enclosed by a base pair. This leads to a four-dimensional matrix, denoted as $D(i, j, k, l)$, because all pairs of possible subsequences have to be treated. The indices i, j refer to a substring $S_1[i, j]$ and the indices k, l to a substring $S_2[k, l]$, respectively. Methods such as the simultaneous folding algorithm from Sankoff [San85] as well as the alignment of base pair probability matrices from Hofacker et al. [HBS04] uses such four-dimensional matrices.

In a first version of the algorithm, we use a similar four-dimensional matrix $D(i, j, k, l)$ to save the best partial solution for LCS-ERP for any pair of substrings over S_1 and S_2. In the following, we propose an improved algorithm, which only needs a two-dimensional matrix in each step and reduces therewith the space complexity to $O(nm)$.

Recursion Formula

In relation to alignment-based methods, we use a similar bottom-up approach to construct the overall solution for the complete sequences S_1 and S_2 from subsequences $S_1[1, i]$ and $S_2[1, l]$. However, we have to treat an exact pattern match as whole unit, whereas alignment approaches are based on single nucleotides or base pairs. Moreover, we have to find a Non-Crossing subset of exact pattern matches.

A solution is achieved with the different notions of bounds for an exact pattern match $\mathcal{E}$ introduced in section 3.3. We make use of these bounds to easily find Non-Crossing regions relative to an EPM as well as to find the start and end of an EPM. For example, all nucleotides before the left-outside-bounds $\mathsf{LEFT}_{\mathcal{E}}$, i.e $S_i[1, \mathsf{LEFT}^i_{\mathcal{E}} - 1]$, fulfil the Non-Crossing condition. Similar all nucleotides after the right-outside-bounds, i.e. $S_i[\mathsf{RIGHT}^i_{\mathcal{E}} + 1, |S_i|]$. With other words, any EPM with its outside-bounds $\mathsf{OUT}_{\mathcal{E}}$ in these region is Non-Crossing relative to the considered EPM.

Similar we handle EPMs that contain base pairs. For example consider an EPM as shown in figure 3.8. Here the indicated pattern exclude two subsequences given through the inside-bounds. Consequently, any pair of inside-bounds $\{(r_i, r_{i+1}), (s_j, s_{j+1})\} \in \mathsf{IN}_{\mathcal{E}}$ describe the borders of a "hole" in the covered sequence. All nucleotides inside these bounds are Non-Crossing, i.e. all EPMs which have outside-bounds within these regions satisfy the inside condition. The set of all holes for an EPM $\mathcal{E}$ is given as follows.

Definition 4.2.1 (Holes of an Exact Pattern Match)
Given a exact pattern match $\mathcal{E}$ over two RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$ with their inside-bounds $\mathsf{IN}_\mathcal{E}$. Then, the set of holes with minimal size γ for this EPM is defined as:

$$\mathsf{HOLES}_\mathcal{E} = \left\{ \left\langle (l^1, r^1), (l^2, r^2) \right\rangle \mid r^1 \geq l^1 + \gamma \ \wedge \ r^2 \geq l^2 + \gamma \ \wedge \ r^i, l^i \in V_i \right\}$$

for that hold:

$$\forall \left\langle (l^1, r^1), (l^2, r^2) \right\rangle \in \mathsf{HOLES}_\mathcal{E} : \exists \left\langle (l^1 - 1, r^1 + 1), (l^2 - 1, r^2 + 1) \right\rangle \in \mathsf{IN}_\mathcal{E}.$$

Clearly, a hole $\left\langle (l^1, r^1), (l^2, r^2) \right\rangle \in \mathsf{HOLES}_\mathcal{E}$ defines a substring $S_1[l^1, r^1]$ in the first RNA and a substring $S_2[l^2, r^2]$ in the second RNA. With γ we refer to the same size as indicated by $\mathbf{E}_\gamma^{1,2}$. This is important, as γ should denote the minimal size of an EPM included in $\mathbf{E}_\gamma^{1,2}$. For example, consider a set which comprises only EPMs with minimal size $\gamma = 4$ and a specific EPM with an inside-bound that encloses three nucleotides. The resulting hole needs not to be considered, because there exists no EPM which fits into this hole. The same holds for holes of size one in the case of the complete set $\mathbf{E}_\gamma^{1,2}$.

Further we need a scheme that treats each $\mathcal{E} \in \mathbf{E}_\gamma^{1,2}$ at a helpful position. Here we can benefit from the properties of $\mathbf{E}_\gamma^{1,2}$. For any pair $(r, s) \in V_1 \times V_2$ of nucleotide positions, there exists exactly one $\mathcal{E}$ or there exists none. The Non-Crossing demand is achieved with the correct usage of the different bounds as motivated above. Any $\mathcal{E}$ is handled only one time at its maximal sequence position, given by its right-outside-bound $\mathsf{RIGHT}_\mathcal{E}$. The score for $\mathcal{E}$ is clearly composed of the score *before* $\mathcal{E}$, given at the position $\mathsf{LEFT}_\mathcal{E} - 1$, plus the size of $\mathcal{E}$ itself, denoted by the function ω, plus possible scores of inside-bounds, given recursively by the computation of $\mathsf{HOLES}_\mathcal{E}$.

This leads to the following recursion formula for any $1 \leq i < j \leq |S_1|$ and $1 \leq k < l \leq |S_2|$:

$$\mathbf{D}(i,j,k,l) = \max \begin{cases} \mathbf{D}(i, j-1, k, l) \\ \mathbf{D}(i, j, k, l-1) \\ \mathbf{D}(i, i'-1, k, k'-1) + \omega(\mathcal{E}) + \sum_{h \in \mathsf{HOLES}_\mathcal{E}} \mathbf{D}_h \\ \quad \text{if } \exists \mathcal{E} \in \mathbf{E}_\gamma^{1,2} \textit{ with } \mathsf{RIGHT}_\mathcal{E} = (j, l),\ \mathsf{LEFT}_\mathcal{E} = (i', k'), \\ \quad i' > i, k' > k \end{cases} \tag{4.1}$$

The first two cases indicate the best found LCS-ERP for the subsequences $S_1[i, j-1]$ and $S_2[k, l-1]$. The third case indicate a matched EPM, i.e. there is an exact pattern match $\mathcal{E}$ in $\mathbf{E}_\gamma^{1,2}$with its right end $\mathsf{RIGHT}_\mathcal{E} = (j, l)$ in both RNAs. The function $\omega(\mathcal{E})$ denotes the score for the EPM itself and is clearly the size $|\mathcal{E}|$. Furthermore we add the score for each enclosed substructure. For any existing hole $h \in \mathsf{HOLES}_\mathcal{E}$ we access a submatrix $\mathbf{D}_h$ from $\mathbf{D}$ as follows:

$$\mathbf{D}_h = \mathbf{D}(l^1, r^1, l^2, r^2) \quad \text{with } h = \left\langle (l^1, r^1), (l^2, r^2) \right\rangle \tag{4.2}$$

This calculates recursively the best score for a given hole h from position l^1 to position r^1 in S_1 and from position l^2 to position r^2 in S_2. Therefore, the score $\omega(\mathcal{E})+\sum_{h\in \mathsf{HOLES}_\mathcal{E}} \mathbf{D}_h$ denotes the total score of $\mathcal{E}$. At last, this score is combined with the already computed score $\mathbf{D}(i, i'-1, k, k'-1)$ before the left-outside-bounds, given by $\mathsf{LEFT}_\mathcal{E} = (i', k')$, and is saved at the entry $\mathbf{D}(i, j, k, l)$, given by the right-outside-bounds $\mathsf{RIGHT}_\mathcal{E} = (j, l)$. Figure 4.1 shows an example for the case of an EPM with two holes.

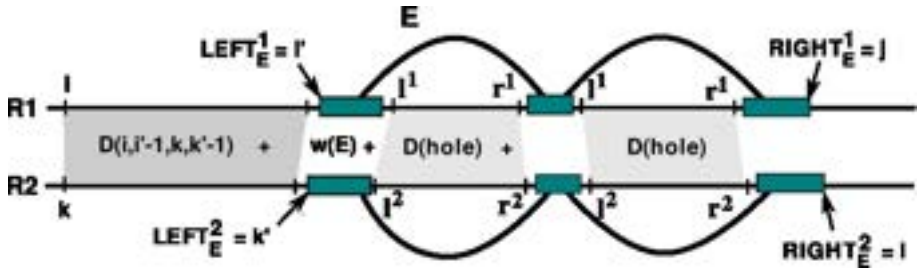

Figure 4.1: The score composition for an EPM $\mathcal{E}$ with two holes and right-outside-bounds $\mathsf{RIGHT}_\mathcal{E} = (j, l)$. The score $\omega(\mathcal{E})$ is combined with the score *before* $\mathcal{E}$ and the score *inside* $\mathcal{E}$. A hole $h = \langle (l^1, r^1), (l^2, r^2)\rangle$ is fixed by the inside-bounds (not shown).

The score for the global best LCS-ERP can be calculated from $\mathbf{D}(1, |S_1|, 1, |S_2|)$. With a traceback the sequence of EPMs can be determined. The matrix $\mathbf{D}$ is initialized with zero values.

Improved Recursion Formula

From the recursion formula 4.1 one can see that the left ends for the subsequences are fixed to the first position in both sequences. Only the recursion for a hole can start at an arbitrary position (l^1, l^2) in both sequences, determined by the inside-bounds. The question arise, whether we can order all holes in a way such that all necessary matrix entries are already computed if an EPM with holes is considered. If this is possible, we can omit the recursion for each hole in the general formula 4.1.

This goal is achieved with an ordering of all holes according to their size in one RNA for a given set $\mathbf{E}^{1,2}_\gamma$. Although we cannot improve the overall running time, we can improve the space complexity. With such an ordering it is possible to reduce the recursion to a two dimensional version. We define a partial ordering on holes in one RNA as follows:

Definition 4.2.2 ($\preceq_{\mathsf{HOLES}}$)
Given a set $\mathbf{E}^{1,2}_\gamma$ of exact pattern matches over two RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. Further let $h_i = \langle (l^1_i, r^1_i), (l^2_i, r^2_i)\rangle \in \mathsf{HOLES}_{\mathcal{E}_i}$ and $h_j = \langle (l^1_j, r^1_j), (l^2_j, r^2_j)\rangle \in \mathsf{HOLES}_{\mathcal{E}_j}$ two holes for any two $\mathcal{E}_i, \mathcal{E}_j \in \mathbf{E}^{1,2}_\gamma$. Then we can define a partial ordering $h_i \preceq_{\mathsf{HOLES}^1} h_j$ in $\mathcal{R}_1$ if and only if h_i is smaller than h_j or of equal size in $\mathcal{R}_1$, i.e.

$$h_i \preceq_{\mathsf{HOLES}^1} h_j \Longleftrightarrow (r^1_i - l^1_i) \leq (r^1_j - l^1_j)$$

Now we can order all holes in one RNA according to their size. Starting the recursion with the smallest holes, for case three in formula 4.1 it is only necessary to consider the score of

the EPM itself, i.e. $\omega(\mathcal{E})$. Any EPM with its right and left outside-bounds inside this hole cannot contain a hole not computed because all smaller holes are already treated. The best score for the hole is finally added to the EPM to which the hole belongs. This implies that for an EPM the overall score $\mathbf{S}_\mathcal{E}$ already exists if it is considered during a recursion from a larger hole. The new two dimensional recursion scheme for any $1 \leq j \leq (r^1 - l^1 + 1)$ and $1 \leq l \leq (r^2 - l^2 + 1)$ for a hole with $h = \langle (l^1, r^1), (l^2, r^2) \rangle$ is given as follows.

$$\mathbf{S}_\mathcal{E} = \omega(\mathcal{E}) \; + \sum_{h \in \mathsf{HOLES}_\mathcal{E}} \mathbf{D}(r^1, r^2) \quad \text{with } h = \langle (l^1, r^1), (l^2, r^2) \rangle \tag{4.3}$$

$$\mathbf{D}(j,l) = \max \begin{cases} \mathbf{D}(j-1, l) \\ \mathbf{D}(j, l-1) \\ \mathbf{D}(i-1, k-1) + \mathbf{S}_\mathcal{E}, \\ \quad \text{if } \exists \mathcal{E} \in \mathbf{E}_\gamma^{1,2} \text{ with } \mathsf{RIGHT}_\mathcal{E} = (j,l), \mathsf{LEFT}_\mathcal{E} = (i,k), i \geq 1, k \geq 1 \end{cases} \tag{4.4}$$

Note that i, j, k, l denote relative hole positions. They need to be transformed into global positions for the access of exact pattern matches from $\mathbf{E}_\gamma^{1,2}$. Formula 4.3 and 4.4 are valid if it is guaranteed that the holes are computed according the ordering $\preceq_{\mathsf{HOLES}}$. This ensures that for each accessed hole h the overall score $\mathbf{S}_\mathcal{E}$ for any $\mathcal{E}$ contained in $\mathbf{D}(r^1, r^2)$ is completely determined. Thus, we need only a two-dimensional matrix to compute the score $\mathbf{D}(j,l)$ of each hole. The space complexity of recursion formula 4.4 is therefore only $O(nm)$.

Algorithm and Implementation

Algorithmically, we need a precomputing of all existing holes. The pseudocode is given below. The algorithm 4.1 (`precompute-holes`) iterates through all holes from the smallest to the largest for a given set $\mathbf{E}_\gamma^{1,2}$. The function `compute-hole-D` given in algorithm 4.2 represents the recursion according to formula 4.4. Note that the given pseudocode iterates directly over global positions instead of relative hole positions. For the sake of clarity, we do not show the data structure to access each EPM by its right-outside-bounds as well as the set $\mathbf{E}_\gamma^{1,2}$itself. The function `scoreEPM(`$\mathcal{E}$`)` gives the overall score $\mathbf{S}_\mathcal{E}$ for each $\mathcal{E} \in \mathbf{E}_\gamma^{1,2}$. The data structure HOLESET contains all holes and maintains moreover the ordering $\preceq_{\mathsf{HOLES}}$. In our implementation this is achieved with the data type *multimap* from the standard template library for C++.

After the precomputing, the best score for the complete sequences can be obtained from the calculation of $\mathbf{D}(|S_1|, |S_2|)$, i.e. treating the whole sequence as hole. With a standard traceback technique through this filled matrix, the set $\mathbf{E}_{\mathsf{LCS}}^{1,2}$ can be generated which denotes the LCS-ERP. Results and examples which we produced with our implementation are given in chapter 6. For a discussion of the problem LCS-ERP see section 6.4.

Algorithm 4.1: `precompute-holes`

Data: $\mathsf{HOLESET}_{\preceq_{\mathsf{HOLES}}} = \{h \mid h = \langle (l^1, r^1), (l^2, r^2) \rangle \in \mathsf{HOLES}_{\mathcal{E}},\ \forall \mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}\}$

Output: scoreEPM($\mathcal{E}$), $\forall \mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$

forall $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$ **do** scoreEPM($\mathcal{E}$) = $|\mathcal{E}|$;

forall $h_{\mathcal{E}} \in \mathsf{HOLESET}$ **do**
  scoreHole = `compute-hole-D`($[l^1_{h_{\mathcal{E}}} \dots r^1_{h_{\mathcal{E}}}], [l^2_{h_{\mathcal{E}}} \dots r^2_{h_{\mathcal{E}}}]$);
  scoreEPM($\mathcal{E}$) = scoreEPM($\mathcal{E}$) + scoreHole;
end

Algorithm 4.2: `compute-hole-D`

Function `compute-hole-D`($[l^1 \dots r^1], [l^2 \dots r^2]$)
  Data: Set $\mathbf{E}_{\gamma}^{1,2}$
  Init: D$[l^1 - 1 \dots r^1][l^2 - 1 \dots r^2] = 0$;
  for $j = l^1$ **to** r^1 **do**
    for $l = l^2$ **to** r^2 **do**
      if $\exists \mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$ *with* $\mathsf{RIGHT}_{\mathcal{E}} = (j, l) \wedge \mathsf{LEFT}_{\mathcal{E}} = (i', k') \wedge (i' > l^1) \wedge (k' > l^2)$
      then
        score$_{\mathcal{E}} = D[i' - 1][k' - 1]$ + scoreEPM($\mathcal{E}$);
      else score$_{\mathcal{E}} = 0$;
      D[j][l] = max {score$_{\mathcal{E}}, D[j - 1][l], D[j][l - 1]$};
    end
  end
  return $D[r^1][r^2]$;

4.3 Correctness and Complexity

Here we show that both recursion schemes give a correct solution for the problem LCS-ERP from definition 4.1.1. Suppose a set $\mathbf{E}_{\gamma}^{1,2}$ of several overlapping and crossing EPMs. First, we consider that there exists no holes at all. The left indices i, k therefore are fixed in formula 4.1 and can be set to one, i.e. we consider a matrix $\mathbf{D}(j, l)$ similar to formula 4.4.

Proposition 4.3.1
Given a set $\mathbf{E}_{\gamma}^{1,2}$ of exact pattern matches over two RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. Further suppose $\forall \mathcal{E}_i \in \mathbf{E}_{\gamma}^{1,2} : \mathsf{HOLES}_{\mathcal{E}_i} = \emptyset$. Then the entry $\mathbf{D}(j, l)$ contains the length of the LCS-ERP for the subsequences $S_1[1, j]$ and $S_2[1, l]$, if each entry is computed from the maximum of $\mathbf{D}(j-1, l)$, $\mathbf{D}(j, l-1)$ or $\mathbf{D}(i'-1, k'-1)+\omega(\mathcal{E})$, with $\mathsf{LEFT}_{\mathcal{E}} = (i', k')$, $\mathsf{RIGHT}_{\mathcal{E}} = (j, l)$.

Proof. For any (j, l) there exists exactly one $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$. We reduce from the left, i.e. suppose the leftmost $\mathcal{E}_m$ for the subsequences $S_1[1, j]$ and $S_2[1, l]$ with the left-outside-

bounds $\mathsf{LEFT}_{\mathcal{E}_m} = (i'_m, k'_m)$ and the right-outside-bounds $\mathsf{RIGHT}_{\mathcal{E}_m} = (j'_m, l'_m)$. The score *before* is $\mathbf{D}(i'_m - 1, k'_m - 1) = 0$ and the score at $\mathbf{D}(j'_m, l'_m)$ is therefore $|\mathcal{E}_m|$. This is the maximal possible length of the LCS-ERP for the subsequences $S_1[1, j'_m]$ and $S_2[1, l'_m]$. Clearly, a single EPM holds the Non-Crossing condition.

Any following entries are filled with the maximum of $\mathbf{D}(j-1, l)$, $\mathbf{D}(j, l-1)$ or $\mathbf{D}(i'-1, k'-1)+\omega(\mathcal{E}_n)$, i.e. there is an EPM $\mathcal{E}_n$ with $\mathsf{LEFT}_{\mathcal{E}_n} = (i'_n, k'_n)$, $\mathsf{RIGHT}_{\mathcal{E}_n} = (j'_n, l'_n)$. The score at $\mathbf{D}(i'_n - 1, k'_n - 1)$ contains the maximal LCS-ERP *before* $\mathcal{E}_n$ and is now extended with the score from the Non-Crossing EPM $\mathcal{E}_n$. If there is no EPM for a pair of positions (j, l), the cases $\mathbf{D}(j-1, l)$, $\mathbf{D}(j, l-1)$ simply transfer the score from the position before which ensures that any pair (j, l) contains a score of the maximal LCS-ERP. □

Now we show that the solution is correct, if the set $\mathbf{E}_\gamma^{1,2}$ contains holes, as well.

Proposition 4.3.2
Given a set $\mathbf{E}_\gamma^{1,2}$ of exact pattern matches over two RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. Then a hole for an EPM $\mathcal{E}$ is correctly computed with $\sum_{h \in \mathsf{HOLES}_\mathcal{E}} \mathbf{D}(l^1, r^1, l^2, r^2)$ or $\sum_{h \in \mathsf{HOLES}_\mathcal{E}} \mathbf{D}(r^1, r^2)$ with $h = \langle (l^1, r^1), (l^2, r^2) \rangle$.

Proof. A hole $h = \langle (l^1, r^1), (l^2, r^2) \rangle$ is given with its indices for the corresponding subsequences $S_1[l^1, r^1]$ and $S_2[l^2, r^2]$. The difference between formula 4.1 and 4.4 is only the point when the submatrix of $\mathbf{D}$ is filled. Formula 4.1 recursively computes it for any found hole, whereas in formula 4.4 it is assumed that each EPM with a hole is already computed if it is treated.

The scheme for a hole h follows exactly the scheme without holes. The difference is that a found EPM $\mathcal{E}_h$ with its right-outside-bounds have to fit also with its left-outside bounds $\mathsf{LEFT}_{\mathcal{E}_h} = (i'_h, k'_h)$ in the considered hole, i.e. $i'_h > l^1$ and $k'_h > l^2$ or $i'_h \geq 1$ and $k'_h \geq 1$. Consequently, all treated EPMs satisfy the Non-Crossing condition relative to that hole, because they lying completely *inside* a hole h. Therefore it is possible to add the score of a hole to the score of the according EPM which contains hole h. In formula 4.4 this is achieved with the defined ordering on holes $\preceq_{\mathsf{HOLES}}$. If the computation starts with the smallest hole, all compatible EPMs during the computation of $D(r^1, r^2)$ have a score inclusive their holes. □

Thus, we have shown that the scores $\mathbf{D}(i, j, k, l)$ for recursion formula 4.1 and $\mathbf{D}(j, l)$ for recursion formula 4.4 represent the length of a longest common subsequence of exact RNA patterns.

Complexity Analysis

Here we give an analysis of the improved version according to formula 4.4 and 4.3. The input for the algorithm are two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$ and a given set $\mathbf{E}_\gamma^{1,2}$. The lengths of the sequences are $|S_1| = n$, $|S_2| = m$.

The time and space complexity depends primarily on the number of exact pattern matches and the number of holes. We emphasized already the uniqueness of each single partial

matching $(r, s) \in \mathcal{M} \subseteq V_1 \times V_2$ in the set $\mathbf{E}_{\gamma}^{1,2}$. Hence we can give the following two statements:

- The set $\mathbf{E}_{\gamma}^{1,2}$contain maximal $n \cdot m$ different EPMs and we can estimate this with $O(nm)$.
- The set $\mathbf{E}_{\gamma}^{1,2}$contain maximal $n \cdot m$ different holes according to definition 4.2.1. This applies for a complexity of $O(nm)$ as well.

The first statement is given according to section 3.2. The second statement is proven next. In algorithm 4.1 we indicate the ordered set of all holes with HOLESET. We use this term in the same way for the following proof.

Proposition 4.3.3
Given a set $\mathbf{E}_{\gamma}^{1,2}$and the ordered set HOLESET *of all holes contained in $\mathbf{E}_{\gamma}^{1,2}$. Then there are maximal $(n \cdot m)$ different holes in* HOLESET*, i.e. it can be estimated with $O(nm)$.*

Proof. Given two nested RNAs $\mathcal{R}_1$ and $\mathcal{R}_2$, there are maximal $|B_1| \leq \frac{n}{2}$ base pairs in B_1 and maximal $|B_2| \leq \frac{m}{2}$ base pairs in B_2. An EPM can contain a hole if and only if it contains at least one base pair to hold the matching pattern condition. As the set $\mathbf{E}_{\gamma}^{1,2}$contains only unique partial matchings, also any base pair part of a hole in the first RNA is matched by at most one base pair part of a hole in the other RNA. This means, if an EPM contains a hole and therefore a base pair matching, this base pair cannot be matched in the same way from another EPM. Thus, there are at most $n \cdot m$ holes in HOLESET from a set $\mathbf{E}_{\gamma}^{1,2}$. □

With that proof we can estimate the running time for the given recursion scheme 4.4 as follows. Each hole from HOLESET is treated one time. To obtain a score for a hole, we fill a two dimensional matrix. The size is determined by the hole and is at most $|S_1[l^1, r^1]| \leq |S_1| = n$ and $|S_2[l^2, r^2]| \leq |S_2| = m$. Therefore, the time to determine the score of a single hole is $O(nm)$ and consequently for all holes we need $O(n^2m^2)$ time. The overall score is obtained from $\mathbf{D}(|S_1|, |S_2|)$ with all EPM scores precomputed. With a traceback the underlying LCS-ERP or the set $\mathbf{E}_{\mathsf{LCS}}^{1,2}$ can be obtained in $O(nm)$. For each hole part of the LCS-ERP a matrix has to be re-filled to find the best trace of EPMs.

The space complexity of the improved version can be estimated with only $O(nm)$. After the computation of each hole, the score is added to the overall score for the corresponding EPM and there is no need to maintain the hole matrix anymore. This is the difference to the first version given in formula 4.1. There we have to maintain a $O(n^2m^2)$ matrix. Clearly, we have to save the score for each EPM and the set $\mathbf{E}_{\gamma}^{1,2}$ itself. However, this not exceeds the previous space complexity of $O(nm)$.

Together with the fact from section 3.4 that we can determine the set $\mathbf{E}_{\gamma}^{1,2}$ in $O(nm)$, we can summarize the complexity to solve the problem LCS-ERP with the following theorem.

Theorem 4.3.1 (LCS-ERP)
Given two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$ and $\mathcal{R}_2 = (S_2, B_2)$. The problem to determine the longest common subsequence of exact RNA patterns (LCS-ERP) is solvable in $O(n^2\, m^2)$ time and $O(nm)$ space.

Chapter 5

A Local Clustering Strategy for Exact Pattern Matches

Chapter 4 deals with a global method for exact pattern matches. In this chapter, we describe our approach for local pairwise comparison of nested RNA secondary structures on the basis of exact pattern matches. Local comparison might better exhibit active sites which are responsible for biological function. Further one can find stable substructures in the considered secondary structures.

In section 5.1, we define a (local) *cluster* of EPMs via a distance constraint. Clusters are constructed from EPMs and this constraint in form of a threshold value describes the maximally allowed distance between different exact pattern matches. In section 5.2, we define different distance functions. To benefit from the fast detection of EPMs in $O(nm)$, we develop a fast clustering algorithm to find such clusters of exact pattern matches. Section 5.3 describes two clustering strategies for this algorithm. Details of our implementation for these strategies are presented in section 5.4. We end with a complexity estimation and define parameters to achieve a fast detection of clusters.

5.1 Local Clusters of Exact Pattern Matches

The local alignment methods mentioned in section 2.3 provide more compact results for less related sequences, because a local alignment do not have to span over the entire sequence. Mostly, this is achieved with setting negative scores to zero and therefore a "new" alignment could start at any other point. In relation to theses methods, we want to find *local* arrangements of EPMs. Following the global method from the last chapter, we want to discover local subsets of EPMs. Clearly, a found subset should be a plain mapping as well as an arc-preserving subsequence. In addition to a similar structural ordering, the identified EPMs have to hold a distance constraint in both RNAs.

We define this constraint via a distance function δ, i.e. two neighbouring EPMs have a distance less than a threshold value. An EPM above the threshold is not a neighbour and therefore these two EPMs build no local arrangement of exact pattern matches. The usage of a general distance function allows the definition of different functions, which makes the approach flexible for different purposes. Further, two EPMs are only neighbors if they hold the Non-Crossing condition from definition 3.3.1. The difficulties are how

to measure the distance and in which way we find fast a good local arrangement common to both RNAs.

The usage of distances instead of gaps and the local view in principle applies more to the domain of data classification and analysis. Therefore we speak of *local clusters* or simply *clusters* for local arrangements of exact pattern matches.

Definition 5.1.1 (Cluster)
Given a set of exact pattern matches $\mathbf{E}_{\gamma}^{1,2}$ *over two RNAs* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *and a threshold value* τ*. A Cluster is a set* $\mathcal{C}^{\delta,\tau} \subseteq \mathbf{E}_{\gamma}^{1,2}$ *of exact pattern matches for which hold:*

1. $\forall \mathcal{E}_i\ \mathcal{E}_j \in \mathcal{C}^{\delta,\tau}\ :\ \mathcal{E}_i$ *and* $\mathcal{E}_j$ *are* Non-Crossing *in* $\mathcal{R}_1$ *and* $\mathcal{R}_2$
2. $\forall \mathcal{E}_i\ \exists \mathcal{E}_j \in \mathcal{C}^{\delta,\tau}\ :\ \delta(\mathcal{E}_i, \mathcal{E}_j) \leq \tau$

The size of a cluster is equal to the number of matched nucleotides, i.e. $|\mathcal{C}^{\delta,\tau}| = \sum_{\mathcal{E} \in \mathcal{C}^{\delta,\tau}} |\mathcal{E}|$. Further, the distance function $\delta(\mathcal{E}_i, \mathcal{E}_j)$ needs an explicit definition. Possible definitions are given in the following section. For example, we define a distance function which uses the number of nucleotides between two exact pattern matches.

5.2 Distance Methods

The following defined distance functions $\delta(\mathcal{E}_i, \mathcal{E}_j)$ are based on the number of nucleotides between two exact pattern matches. The Non-Crossing condition is a precondition on all clusters and we can make use of this for the different distance functions. If two exact pattern matches are Crossing, then their distance is set to infinity. Further we have to treat the fact that the distance between two EPMs is different in $\mathcal{R}_1$ and $\mathcal{R}_2$. Therefore we determine in a first step the distance in each RNA separately and then we combine them in a second step.

5.2.1 DISTANCE-SEQUENCE

A simple and intuitive distance measure is based on the minimal length of the primary structure between two neighbouring exact pattern matches. To determine the distance, we make use of the matching bound definition 3.3.3. The following definition is given for a single RNA. For example, for RNA $\mathcal{R}_1$ we define δ^1_{SEQ} as follows.

$$\delta^1_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) = \begin{cases} \infty, & \mathcal{E}_i \between \mathcal{E}_j \\ r - \mathsf{RIGHT}^1_{\mathcal{E}_j}, & \mathcal{E}_i \Subset \mathcal{E}_j,\ \exists\, (l, r) \in \mathsf{IN}^1_{\mathcal{E}_i} : \\ & \qquad l < \mathsf{LEFT}_{\mathcal{E}_j} \ \wedge\ \mathsf{RIGHT}_{\mathcal{E}_j} < r \\ \mathsf{LEFT}^1_{\mathcal{E}_j} - \mathsf{RIGHT}^1_{\mathcal{E}_i}, & \mathcal{E}_i \Vdash \mathcal{E}_j \end{cases} \tag{5.1}$$

The distance function δ^1_{SEQ} for a cluster is given as the length of the sequence between the right-outside-bounds and the left-outside-bounds, if the considered EPMs satisfy the

before/after condition for NON-CROSSING (third case). In the case of nested EPMs, the distance function for a cluster is defined as the length of the sequence between the inside-bound and the right-outside-bound (second case). This is necessary because the later clustering algorithm works from the left to the right over a given sequence.

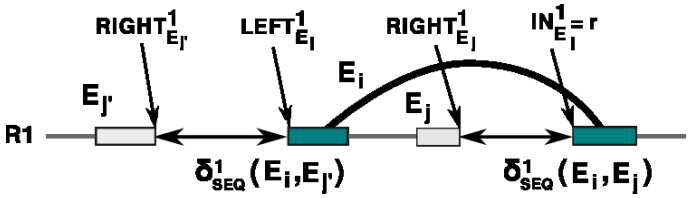

Figure 5.1: Illustration for distance function δ^1_{SEQ}. If $\mathcal{E}_{j'}$ is before $\mathcal{E}_i$, then the distance $\delta^1_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_{j'})$ is determined by the outside-bounds. If $\mathcal{E}_j$ is inside $\mathcal{E}_i$, then $\delta^1_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j)$ is determined between the inside-bound and the outside-bound.

Now we can combine the distances of each RNA to the distance function δ_{SEQ} which we call DISTANCE-SEQUENCE.

$$\delta_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) = \delta^1_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) + \delta^2_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) \tag{5.2}$$

With formula 5.2 we can calculate the distance between two arbitrary EPMs. To identify a cluster with this distance function, a threshold value τ is needed. Here τ can be interpreted as the maximal number of nucleotides allowed between two exact pattern matches.

For the clustering strategy we need a slightly different version of δ_{SEQ} which is better in an algorithmic manner. We simply change that both distances have to be below a given threshold τ. Then we write the distance function for a cluster as:

$$\delta_{\text{SEQ1}}(\mathcal{E}_i, \mathcal{E}_j) = \begin{cases} \text{TRUE}, & \delta^1_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) \leq \tau \ \wedge \ \delta^2_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) \leq \tau \\ \text{FALSE}, & \textit{otherwise} \end{cases} \tag{5.3}$$

If the length of the sequences are notably different, it is possible to use relative distances as well as relative thresholds instead of absolute values.

5.2.2 DISTANCE-SEQUENCE-EQUAL

The distance function δ_{SEQ1} handles all exact pattern matches within a given threshold equally. Finding clusters with more similar distances, we can incorporate a constraint on the differences of the distances. Such a distance can be useful to detect motives in RNAs which need a certain distance in their tertiary structure for functionality.

With the following distance function we define two EPMs as local, if they have similar distances in both RNAs. For example, suppose a given threshold $\tau = 50$ and two EPMs with a distance $\delta^1_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) = 5$ and $\delta^2_{\text{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) = 45$. Then both EPMs can be combined with δ_{SEQ1}, although the large difference of 40 nucleotides. An additional constraint could restrict the allowed difference for example to 10 nucleotides. Look at figure 5.2 below for

an illustration of that distance function. We call such a distance function DISTANCE-SEQUENCE-EQUAL.

$$\delta_{\mathsf{EQL}}(\mathcal{E}_i, \mathcal{E}_j) = \begin{cases} \text{TRUE}, & \delta_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) \leq \tau \ \wedge \ \mid \delta^1_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) - \delta^2_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j) \mid \leq \Delta_{\mathsf{DT}} \\ \text{FALSE}, & \textit{otherwise} \end{cases} \tag{5.4}$$

The threshold τ is used here as the general threshold for the allowed distance of two EPMs. In addition, the δ_{EQL} function needs a second parameter Δ_{DT} which denotes the threshold for the allowed differences between the distances in the first and second RNA.

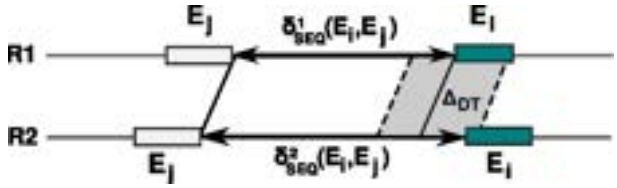

Figure 5.2: Illustration for distance function δ_{EQL}. The grey shaded region denotes the allowed difference between the distances $\delta^1_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j)$ and $\delta^2_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j)$, determined by Δ_{DT}. The black line denotes the distance $\delta^1_{\mathsf{SEQ}}(\mathcal{E}_i, \mathcal{E}_j)$.

5.2.3 DISTANCE-STRUCTURE-SHORTESTPATH

In contrast to the distance functions above which are based on the sequence length between two exact pattern matches, we want to define here a function which is based on the secondary structure. Suppose two RNA structures and one of them has a large stem-loop inserted in a common multi-loop. Then two exact pattern matches in the multi-loop with that hairpin in between are not local with the above functions based on the primary sequence. However, in a structural sense they are near if the hairpin is excluded. This means if one can "walk" over hydrogen bonds and count the passed nucleotides in that way, the resulting distance reflects better the given local structural properties of the RNA molecule. Figure 5.3 shows such an example.

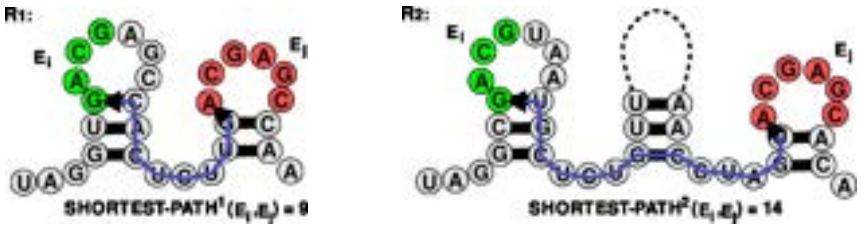

Figure 5.3: Illustration for distance function δ_{PATH}. The blue lines denote the shortest paths between the indicated EPMs $\mathcal{E}_i$ and $\mathcal{E}_j$. The distance is independent of the size of the stem in the middle of $\mathcal{R}_2$.

According to our path definition 3.1.1, we therefore search for the minimal path between two EPMs to determine their distance. We call this function DISTANCE-STRUCTURE-SHORTESTPATH and is given for a single RNA as follows.

$$\delta^1_{\mathsf{PATH}}(\mathcal{E}_i, \mathcal{E}_j) = \min \begin{cases} \infty, & \mathcal{E}_i \not\between \mathcal{E}_j \\ \mathsf{SHORTEST\text{-}PATH}^1(\mathcal{E}_i, \mathcal{E}_j), & \mathcal{E}_i, \mathcal{E}_j \text{ Non-Crossing} \end{cases} \tag{5.5}$$

The problem for an algorithm is the complexity of the path finding problem, denoted here with the function SHORTEST-PATH. It depends on the structural elements of the RNAs. A precomputation of the structural elements could be helpful, but still the path between two arbitrary EPMs have to be determined separately. This probably exceeds the desired overall complexity of an algorithm. With an inside-to-outside algorithm we are already capable to access a specific structural path in one RNA in $O(1)$. But it remains incomplete and therefore the path finding problem needs further investigation.

5.3 Clustering Strategies

In this section we describe the algorithmic approach to find local clusters of exact pattern matches. In contrast to the global case from chapter 4, no method is actually known to find *optimal* local clusters of exact pattern matches. Methods like the LSSA-algorithm from Backofen and Will [BW04] solves the related, but different local sequence-structure alignment problem optimal. However, this approach with its time complexity of $O(n^2m^2 \max(n, m))$ is not applicable to large RNA structures. We recall here again that the `MCS` algorithm [SB07] described in section 3.4 determines the set $\mathbf{E}^{1,2}_{\gamma}$ in only $O(nm)$. This is fast and opens its application to large RNA structures with even several thousands of nucleotides. For such large RNAs the comparison is rather limited by finding the secondary structure with methods like `RNAfold` [HFS+94]. Moreover, an optimal local method has probably a complexity comparable to the global case. In the following we develop an algorithm which finds arrangements of exact pattern matches in form of clusters within a reasonable and scaleable time.

The Clustering Principle

Our approach follows a greedy strategy to find local clusters. In general, a greedy strategy chooses at any point the most profitable solution. Here, in each clustering step the best solution, i.e. the cluster with the largest size, is chosen. Consequently, we follow for the local approach a strategy which maximizes the size of a cluster $|\mathcal{C}^{\delta,\tau}|$, i.e. the number of matching nucleotides.

The basic idea is to maintain a set CANDSET of not fully extended clusters. Then for each new considered EPM a set of clusters which hold the distance and Non-Crossing constraints, called *candidate* clusters, is determined. In order to expand a candidate cluster with a new EPM according to a chosen strategy, the best candidate cluster is removed from CANDSET. Next, the solution is built and finally it is inserted in CANDSET. Thus, each clustering step consists of three stages, independent from the chosen distance function and clustering strategy. For any exact pattern match $\mathcal{E} \in \mathbf{E}^{1,2}_{\gamma}$, the following steps are processed.

1. Determine all *candidate* clusters $\mathcal{C}^{\delta,\tau}_{cand}$ for $\mathcal{E}$ from CANDSET, i.e. $\exists \mathcal{E}' \in \mathcal{C}^{\delta,\tau}_{cand} : \delta(\mathcal{E}, \mathcal{E}') \leq \tau$.
2. Find the NON-CROSSING cluster $\mathcal{C}^{\delta,\tau}_{max}$ with $\max(|\mathcal{C}^{\delta,\tau}_{cand}|)$, remove $\mathcal{C}^{\delta,\tau}_{max}$ from CANDSET and build solution from $\mathcal{C}^{\delta,\tau}_{max}$ with $\mathcal{E}$ according to chosen strategy.
3. Add the solution to the pool of all candidates in CANDSET.

The first step comprises the necessary conditions to build a valid cluster according to definition 5.1.1. A *candidate* denotes a cluster which is already found and satisfying the distance constraint. The third step is necessary to increase the pool of candidate clusters. Hence, alternative strategies can be build only around step two. In the following we define two clustering strategies, differing in the way the solution is built. Both strategies use as "seed" a cluster which comprises only a single EPM.

CLUSTER-MAX-1

The scheme for that strategy is as follows. Suppose a currently treated exact pattern match $\mathcal{E}$ and further a candidate cluster $\mathcal{C}^{\delta,\tau}_{max}$ with maximal size $|\mathcal{C}^{\delta,\tau}_{max}|$. First, the cluster $|\mathcal{C}^{\delta,\tau}_{max}|$ is removed from CANDSET, i.e CANDSET $\setminus\, \mathcal{C}^{\delta,\tau}_{max}$. Then we build the solution with:

$$\mathbf{S}_{\mathsf{MAX1}} = \left\{ \mathcal{C}^{\delta,\tau}_{max} \cup \{\mathcal{E}\} \right\} \tag{5.6}$$

This means, that the best candidate cluster $\mathcal{C}^{\delta,\tau}_{max}$ is extended with the current EPM $\mathcal{E}$ and the former candidate cluster is replaced. With other words, any existing cluster is clustered only once. This imply that a cluster is not independent from the ordering of the treated EPMs. Finally, the solution is inserted in the set of candidate clusters, i.e. CANDSET $\cup\, \mathbf{S}_{\mathsf{MAX1}}$.

CLUSTER-MAX-2

This strategy is similar to the first clustering strategy except that the candidate cluster remains a candidate. Therefore, the solution for step two of the clustering principle comprise two clusters. The new cluster is build from the candidate and the current treated exact pattern match. Formally, this is as follows:

$$\mathbf{S}_{\mathsf{MAX2}} = \left\{ \mathcal{C}^{\delta,\tau}_{max} \cup \{\mathcal{E}\}, \mathcal{C}^{\delta,\tau}_{max} \right\} \tag{5.7}$$

This clustering strategy can be interpreted as a compensation strategy. A candidate cluster can be combined with different exact pattern matches as long as the candidate is within the threshold for the distance function. Here a cluster is less dependent from the ordering of the treated EPMs. Finally, the solution is inserted in the set of candidate clusters, i.e. CANDSET $\cup\, \mathbf{S}_{\mathsf{MAX2}}$

5.4 The Pairwise Pattern Clustering Algorithm

In the following we realize a fast clustering strategy of the introduced clustering principle. The algorithm proceeds an inside-to-outside scheme in general with all loop regions treated first. This means that a base pair $(i, i') \in B$ is only processed, if all base pairs $(j, j') \in B$ with $i < j < j' < i'$ are processed before.

To achieve a clustering method which preserves the fast detection of the set $\mathbf{E}_{\gamma}^{1,2}$, we iterate only through one RNA and handle each nucleotide only once. A preprocessing step is needed to modify and order the given exact pattern matches for later clustering. In the following we describe the preprocessing step and the clustering step of our algorithm in detail. By convention, we always process the first RNA and determine all dependencies within the second RNA simultaneously.

5.4.1 Preprocessing

Matching Closures: This step determines the matching closure $\overline{\mathcal{E}}$ according to definition 3.3.4 for all exact pattern matches $\mathcal{E} \in \mathbf{E}_{\gamma}^{1,2}$. We denote the set of all matching closures as $\overline{\mathbf{E}}_{\gamma}^{1,2}$ and all following operations like the determination of the matching bounds or the test for Non-Crossing are based on this set. The usage of the matching closure is necessary to ensure that the ordering according to the right-outside-bound in one RNA excludes all cases of overlapping and crossing EPMs correctly. The matching closure enforces that all right-outside-bound positions are either unbound or right-paired. This allows in stem regions the algorithmic checking of Non-Crossing with one specific sequence position. In consequence of matching closures, specific pairs of EPMs are treated as Crossing, although the underlying matchings are Non-Crossing. Figure 5.4 shows such an example.

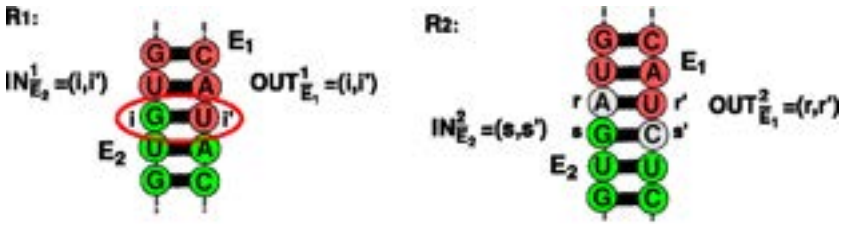

Figure 5.4: An arrangement of two EPMs which do not satisfy Non-Crossing, if the check is based on matching closures. The problem occurs in $\mathcal{R}_1$. Here the inside-bounds of $\overline{\mathcal{E}}_1$ and the outside-bounds of $\overline{\mathcal{E}}_2$ covering the same base pair (i, i').

Ordering: Further, an ordering for all $\overline{\mathcal{E}} \in \overline{\mathbf{E}}_{\gamma}^{1,2}$ along the primary structure of the first RNA is needed. This ensures that each EPM is treated only once in the clustering algorithm. Similar to the LCS-ERP algorithm, we order all EPMs $\overline{\mathcal{E}} \in \overline{\mathbf{E}}_{\gamma}^{1,2}$ according to the right-outside-bounds. Due to the fact that the input is ordered along $\mathcal{R}_1$, we separate all EPMs by the $\mathsf{RIGHT}_{\overline{\mathcal{E}}}^{1}$ value. Clearly, this position is not assigned to a unique EPM. The resulting set of EPMs assigned to nucleotide position k is denoted as

$$\mathsf{EPMORDER}_k = \{\ \mathcal{E} \mid \mathsf{RIGHT}_{\overline{\mathcal{E}}}^{1} = k\ \} \subseteq \mathbf{E}_{\gamma}^{1,2}.$$

5.4.2 Clustering

In order to realize the introduced clustering strategies from section 5.3, a set of candidate clusters for each exact pattern match is needed. We call this set CANDSET and it is best composed with an inside-to-outside scheme, because each innermost substructure contains only a small number of EPMs. This reduces the number of possible candidate clusters in general and the best chosen candidate cluster is propably a nearly optimal clusters. In the following we assume a given ordering $\mathsf{EPMORDER}_k$ for any position k with $1 \leq k \leq |S_1|$.

The outline for this section is as follows. At first, we describe the data structure for the set of candidate clusters. Next, we explain in which way a single EPM with its different shapes is clustered. Then we give details on the handling of loops and finally we give the overall algorithm using pseudocode notation.

Data Structure for the Set of Candidate Clusters

The set of candidate clusters for each nucleotide position k is maintained in the set $\mathsf{CANDSET}_k$. This set contains all clusters $\mathcal{C}^{\delta,\tau} \in \mathsf{CANDSET}_k \subseteq \mathsf{CANDSET}$, which are within the given threshold τ from sequence position k for the chosen distance function δ in $\mathcal{R}_1$. As we process all positions k from inside to outside, any new inner loop starts with an empty set $\mathsf{CANDSET}_k$. Hence, if a nucleotide $S[k]$ is part of a base pair $(k', k) \in B$, $\mathsf{CANDSET}_k$ contains only clusters with EPMs *inside* from position k. Second, if $S[k]$ is a nucleotide within a loop, all EPMs from clusters in $\mathsf{CANDSET}_k$ have outside-bounds between the beginning of the loop and position k. Note that we only treat right-paired nucleotides in consequence of matching closures. Therefore, any left-paired nucleotide k contains an empty set $\mathsf{CANDSET}_k$. In order to have all clusters within threshold τ in $\mathsf{CANDSET}_k$, each $\mathsf{CANDSET}_k$ needs an update from a position k' inside or before k. The update inserts all clusters from $\mathsf{CANDSET}_{k'}$ in $\mathsf{CANDSET}_k$ which still hold the distance constraint. During the algorithm, we then can simply iterate through each $\mathsf{CANDSET}_k$ to access all candidates for further treatment.

Clustering of Different Shapes of Matchings

Depending on the shape, an exact pattern match has to be tested against different and moreover independent sets of candidate clusters. Due to the inside-to-outside scheme, all these candidate clusters are lying *inside* or *before* the treated EPM. The necessary positions k to access $\mathsf{CANDSET}_k$ are determined by the inside-bounds and outside-bounds and are identified as follows.

First, suppose an EPM $\mathcal{E}$ with a non empty set of inside bounds $\mathsf{IN}^1_{\mathcal{E}}$. For any $(r_i, r_{i+1}) \in \mathsf{IN}^1_{\mathcal{E}}$, the set $\mathsf{CANDSET}_k$ with $k = r_{i+1} - 1$ contains the candidate clusters *inside* the treated inside-bound (r_i, r_{i+1}) of $\mathcal{E}$ in $\mathcal{R}_1$. Clearly, all candidate clusters have to be tested for the cluster constraints in $\mathcal{R}_2$, but this can be reduced to a single test between $\mathcal{E}$ and a specific EPM from the candidate cluster.

The set of candidate clusters *before* $\mathcal{E}$ depends on the structure type of the nucleotide before the left-outside-bound at position $\mathsf{LEFT}^1_{\mathcal{E}} - 1$. If the nucleotide $S_1[\mathsf{LEFT}^1_{\mathcal{E}} - 1]$ is unpaired or right-paired (!), then the set $\mathsf{CANDSET}_k$ with $k = \mathsf{LEFT}^1_{\mathcal{E}} - 1$ contains all candidate clusters *before* $\mathcal{E}$ but within the current inner loop in $\mathcal{R}_1$. Similarly to the case above, each candidate cluster has to be tested for the cluster constraints in $\mathcal{R}_2$. If the nucleotide $S_1[\mathsf{LEFT}^1_{\mathcal{E}} - 1]$ is left-paired (!), we cannot cluster anything, because $\mathcal{E}$ is inside or at the end of a stem.

These given schemes for the handling of EPMs from $\mathsf{EPMORDER}_k$ are combined in function `clusterEPM`. For the algorithm we denote the set of positions k with nonempty candidate cluster sets for an EPM $\mathcal{E}$ with

$$\mathsf{CANDPOS}_{\mathcal{E}} = \{k \mid \mathsf{CANDSET}_k \text{ contains candidate clusters for } \mathcal{E}\}.$$

In figure 5.5 all position from $\mathsf{CANDPOS}_{\mathcal{E}}$ for the shown EPM are red colored. Suppose a base pair between $S[i-1]$ and the last U nucleotide. Then position $i-1$ is not contained in $\mathsf{CANDPOS}_{\mathcal{E}}$.

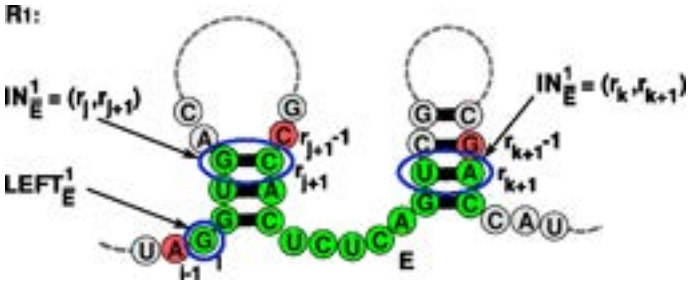

Figure 5.5: An exact pattern match $\mathcal{E}$ (green nucleotides) and positions which contain candidate clusters. The positions $\mathsf{CANDPOS}_{\mathcal{E}} = \{i-1, r_{j+1}-1, r_{k+1}-1\}$ are red marked, necessary bounds are blue marked.

Clustering of Loops and Multi-Loops

According to the loop decomposition for an RNA secondary structure shown in figure 2.3 on page 17, each loop region is limited by a closing base pair. We denote this base pair with (r_0, r'_0). In the case of internal loops, bulges and multi-loops, there exist base pairs (r_i, r'_i) with $r_0 < r_i < r'_i < r'_0$, which limit branching substructures.

Suppose a base pair (r_0, r'_0) which closes a loop. According to the used inside-to-outside scheme, the loop region is traversed from the left to the right, i.e. from $S_1[r_0 + 1]$ to $S[r'_0 - 1]$. If a base pair (r_i, r'_i) is found, this substructure is traversed first from inside to outside and the loop-walk is continued afterwards. For such base pairs (r_i, r'_i) we need an additional clustering operation which combines clusters inside base pair (r_i, r'_i) and clusters produced between $S_1[r_0 + 1]$ and $S_1[r_i - 1]$. If a position r'_i is processed during the algorithm, these clusters are already produced. Therefore we can simply add a clustering operation between the sets $\mathsf{CANDSET}_{r'_i}$ and $\mathsf{CANDSET}_{r_i - 1}$. We call this function `clusterSTEM` and it is invoked for all nucleotide positions r'_i, if $(r_i, r'_i) \in B$ is base pair in the described manner.

The function `clusterSTEM` combines clusters with clusters in a fixed manner. Each cluster from $\mathsf{CANDSET}_{r'_i}$ is combined with the best cluster, i.e. the largest cluster, from $\mathsf{CANDSET}_{r_i-1}$. The best cluster remains a candidate cluster. Clearly, two combined clusters have to fulfil the cluster constraints, i.e Non-Crossing and the distance threshold. With a clever ordering of the EPMs part of a cluster, the last added EPM is either the most *outside* or the last *before* EPM. Therefore this check can be done in $O(1)$ for each cluster from $\mathsf{CANDSET}_{r'_i}$.

Figure 5.6 below illustrates the loop walk. The positions for `clusterSTEM` are red marked. With this given scheme for loops, any loop closing base pair (r_0, r'_0) needs no special treatment. All EPMs in $\mathsf{EPMORDER}_{r'_0}$ are clustered with `clusterEPM`.

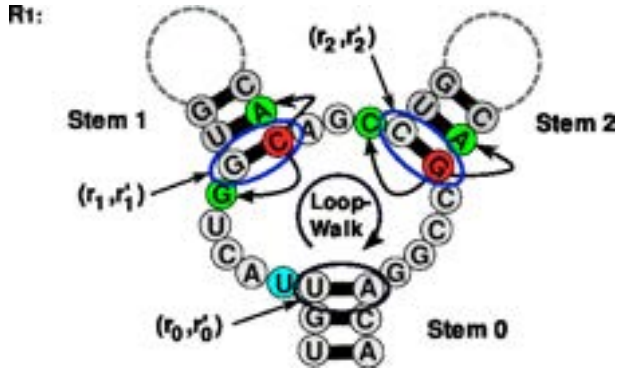

Figure 5.6: Clustering of a multi-loop closed by base pair (r_0, r'_0). The blue circled base pairs delimit stem 1 and stem 2. For the red marked nucleotides the `clusterSTEM` operation is applied. This function combines the green marked sets $\mathsf{CANDSET}_{r'_i-1}$ and $\mathsf{CANDSET}_{r_i-1}$. The loop-walk starts with an empty set $\mathsf{CANDSET}_{r_0+1} = \emptyset$ (blue nucleotide).

Complete Algorithm

Now we can give the complete algorithm. We separate it into two parts: (1) For each nucleotide position $1 \leq k \leq |S_1|$ the function `clusterEPM`(k) clusters all exact pattern matches found at position k, i.e in $\mathsf{EPMORDER}_k$. (2) The function `clusterSTEM`(r') handles base pairs (r, r') which delimit a substructure within a loop.

Note that RNA $\mathcal{R}_1$ is processed from inside to outside with loop positions first. Further, each loop starts with an empty set $\mathsf{CANDSET}_k = \emptyset$. Dangling ends of $\mathcal{R}_1$ are treated as loop positions as well. The pseudocode for `clusterEPM` is given in algorithm 5.2 below.

`clusterEPM`(k) // $1 \leq k \leq |S_1|$

1. For all EPMs $\mathcal{E} \in \mathsf{EPMORDER}_k$ determine set $\mathsf{CANDPOS}_{\mathcal{E}}$.
2. Find best cluster $\mathcal{C}^{\delta,\tau}_{max}$ for each $k' \in \mathsf{CANDPOS}_{\mathcal{E}}$, i.e. find best cluster from $\mathsf{CANDSET}_{k'}$, for which $\{\mathcal{C}^{\delta,\tau}_{max} \cup \{\mathcal{E}\}\}$ is a valid cluster in $\mathcal{R}_2$.
3. Build solution $\mathbf{S}_{\mathsf{MAX}}$ for each $\mathcal{E}$ with $\mathcal{C}^{\delta,\tau}_{max}$ according to chosen strategy.

4. Insert solution in $\mathsf{CANDSET}_k$ for each $\mathcal{E}$, i.e. $\mathcal{E}, \mathsf{CANDSET}_k \cup \mathbf{S}_{\mathsf{MAX}}$.

During a loop walk, the following function `clusterSTEM`(r') is applied to any position r', with $(r, r') \in B_1$ and (r, r') delimits a substructure. The additional condition for the right-outside-bound in step one is needed because all EPMs in $\mathsf{EPMORDER}_{r'}$ are already treated during the `clusterEPM` function.

`clusterSTEM`(r, r') // $(r, r') \in B_1$ is delimiting base pair

1. $\forall \mathcal{C}^{\delta,\tau} \in \mathsf{CANDSET}_{r'}$, $\forall \mathcal{E} \in \mathcal{C}^{\delta,\tau}$ that hold $\mathsf{RIGHT}^1_{\mathcal{E}} < r'$, find best cluster $\mathcal{C}^{\delta,\tau}_{max} \in \mathsf{CANDSET}_{r-1}$ for which $\{\mathcal{C}^{\delta,\tau} \cup \mathcal{C}^{\delta,\tau}_{max}\}$ is a valid cluster in $\mathcal{R}_2$.
2. Build solution $\mathbf{S}_{\mathsf{STEM}} = \left\{ \{\mathcal{C}^{\delta,\tau} \cup \ \mathcal{C}^{\delta,\tau}_{max}\} \right\}$ for each $\mathcal{C}^{\delta,\tau}$
3. Insert solution in $\mathsf{CANDSET}_{r'}$ for each $\mathcal{C}^{\delta,\tau}$, i.e. $\mathsf{CANDSET}_{r'} \cup \mathbf{S}_{\mathsf{STEM}}$

Note that in both functions the verification of a valid cluster in $\mathcal{R}_2$ can be reduced to a single Non-Crossing test with an appropriate data structure for the clusters itself. Additionally, for each position k, the set $\mathsf{CANDSET}_k$ has to be updated with all clusters from $\mathsf{CANDSET}_{k-1}$, which are within threshold τ for the chosen distance function δ. Further, if k is right paired and delimits a stem, the update is also necessary from the set $\mathsf{CANDSET}_{r-1}$, if (r, r') is the limiting bond in B_1. At the end we have to insert all current EPMs in $\mathsf{CANDSET}_k$ as seed cluster. Seed clusters as generated as follows: $\forall \mathcal{E} \in \mathsf{EPMORDER}_k$, $\mathcal{C}^{\delta,\tau}_{\mathsf{SEED}} = \{\mathcal{E}\}$. Then each seed cluster is inserted in CANDSET, i.e. $\mathsf{CANDSET}_k \cup \{\mathcal{C}^{\delta,\tau}_{\mathsf{SEED}}\}$.

With a traceback through the set CANDSET, clusters with different properties can be retrieved. If only the best cluster should be obtained, no traceback is needed. In the pseudocode below the largest cluster from CANDSET is denoted as $\mathcal{C}^{\delta,\tau}_{\mathsf{BEST}}$. The other clusters can be used for further analysis. For example, one can ask for the best cluster inside an arbitrary base pair (i, i'). Such requests are supported by the data structure CANDSET.

Next we give the pseudocode for the clustering algorithm. We omit the function `clusterSTEM` and give instead the main loop of the clustering algorithm.

Algorithm 5.1: `clusterAll (main loop)`

Output: cluster $\mathcal{C}^{\delta,\tau}_{\mathsf{BEST}}$, with $|\mathcal{C}^{\delta,\tau}_{\mathsf{BEST}}|$ is maximal in CANDSET

Preprocessing($\overline{\mathbf{E}}^{1,2}_{\gamma}$, EPMORDER);

foreach $1 \leq k \leq |S_1|$ *from inside to outside in* B_1 **do**
  $\mathsf{CANDSET}_k = \emptyset$;
  clusterEPM(k);
  if $\exists (r, r') \in B_1$ *with* $k = r'$ *and* (r, r') *limits a stem* **then**
    clusterStem(r, r');
    update $\mathsf{CANDSET}_k$ from $\mathsf{CANDSET}_{r-1}$;
  end
  if $STRUCT_1(k-1) \neq$ *left-paired* **then** update $\mathsf{CANDSET}_k$ from $\mathsf{CANDSET}_{k-1}$
end
Traceback(CANDSET)

Algorithm 5.2: `clusterEPM`

Procedure `clusterEPM(`k`)`
  forall $\mathcal{E}_j \in \mathsf{EPMORDER}_k$ **do**
    $\mathcal{C}^{\delta,\tau}_S := \{\mathcal{E}_j\}$; $\mathbf{S}_{\mathsf{MAX}} = \emptyset$;
    forall $k' \in \mathsf{CANDPOS}_{\mathcal{E}_j}$ **do**
      maxSize := 0;
      forall $\mathcal{C}^{\delta,\tau}_i \in \mathsf{CANDSET}_{k'}$ **do**
        if $\mathcal{C}^{\delta,\tau}_i \cup \{\mathcal{E}_j\}$ *is cluster in* $\mathcal{R}_2$ *and* maxSize $\leq |\mathcal{C}^{\delta,\tau}_i|$ **then**
          maxSize :=$|\mathcal{C}^{\delta,\tau}_i|$;
          $\mathcal{C}^{\delta,\tau}_{max} = \mathcal{C}^{\delta,\tau}_i$;
        end
      end
      $\mathcal{C}^{\delta,\tau}_S := \mathcal{C}^{\delta,\tau}_S \cup \mathcal{C}^{\delta,\tau}_{max}$;
      $\mathsf{CANDSET}_{k'} \setminus \mathcal{C}^{\delta,\tau}_{max}$;
      if *CLUSTER-MAX-2* **then** $\mathbf{S}_{\mathsf{MAX}} \cup \{\mathcal{C}^{\delta,\tau}_{max}\}$
    end
    $\mathbf{S}_{\mathsf{MAX}} \cup \{\mathcal{C}^{\delta,\tau}_S\}$; `/* CLUSTER-MAX-1 and CLUSTER-MAX-2 */`
    $\mathsf{CANDSET}_k := \mathsf{CANDSET}_k \cup \mathbf{S}_{\mathsf{MAX}}$;
    $\mathsf{CANDSET}_k := \mathsf{CANDSET}_k \cup \{\mathcal{C}^{\delta,\tau}_{\mathsf{SEED}_j}\}$; `/*` $\mathcal{C}^{\delta,\tau}_{\mathsf{SEED}_j}$ `is seed cluster for` $\mathcal{E}_j$ `*/`
    if $|\mathcal{C}^{\delta,\tau}_{\mathsf{BEST}}| \leq |\mathcal{C}^{\delta,\tau}_S|$ **then** $\mathcal{C}^{\delta,\tau}_{\mathsf{BEST}} := \mathcal{C}^{\delta,\tau}_S$;
  end

5.4.3 Complexity Analysis

In contrast to optimal algorithms, a worst case analysis is difficult for greedy techniques. Nevertheless, an estimation of the average number of performed operation is possible. Therefore we analyse the clustering strategy under specific parameters.

Due to the fact that the threshold value τ is fixed and determines how long a cluster is "active" in the set CANDSET, this provides a basis for an analysis. In the following we estimate the performed number of operations for the strategy CLUSTER-MAX-2 and distance function δ_{SEQ1}.

For the analysis we suppose two given RNAs with their sequence lengths $|S_1| = n$ and $|S_2| = m$. Further there are $p = |\mathbf{E}_\gamma^{1,2}|$ exact pattern matches given. Then we can assume on average

$$a_\mathcal{E} = \frac{p}{n} \tag{5.8}$$

exact pattern matches for each nucleotide k. This equals the average number of EPMs for each $\mathsf{EPMORDER}_k$. Starting with a set $\mathsf{CANDSET} = \emptyset$, there are at first $a_\mathcal{E}$ clusters and then for each following position the size increases with $2 \cdot a_\mathcal{E}$ maximal. After τ steps, $a_\mathcal{E}$ clusters are out of the threshold and then with each following step there are $2 \cdot a_\mathcal{E}$ clusters out of the threshold. Consequently there are

$$cand_k = 2 \cdot a_\mathcal{E} \cdot (\tau + 1) \cdot c \tag{5.9}$$

candidate clusters for each $\mathsf{CANDSET}_k$ with some constant c for the secondary structure. With an iteration over the first sequence, there are together $2 \cdot p \cdot (\tau + 1) \cdot c$ generated clusters. Hence we have to test on average $cand_k$ clusters to find for each exact pattern match the best cluster. This reveals

$$cand_k \cdot p = 2 \cdot a_\mathcal{E} \cdot p \cdot (\tau + 1) \cdot c = \frac{2 \cdot p^2 \cdot (\tau + 1) \cdot c}{n} \tag{5.10}$$

as an estimation of the needed overall operations. Note that each `clusterSTEM` operation produces some $c \cdot cand_k$ additional clusters for some positions. Further the update of each $\mathsf{CANDSET}_k$ needs on average $cand_k$ operations. We assume these factors included in the constant c. The test for Non-Crossing in $\mathcal{R}_2$ can be done in $O(1)$ for both clustering operations.

For real applications it is interesting to determine the input parameters against an expected average time complexity. For example, to perform not more than $O(nm)$ operations, we can estimate the size $|\mathbf{E}_\gamma^{1,2}|$ and threshold τ with

$$\hat{\tau} \leq \frac{n^2 m}{2 \cdot P^2 \cdot c} \qquad\qquad \hat{P} \leq \sqrt{\frac{n^2 \cdot m}{2 \cdot c \cdot (\tau + 1)}} \tag{5.11}$$

The correctness for the clustering algorithm follows from the fact that each cluster is build up from a seed cluster with a single EPM. Any extension is only done with an EPM lying *outside* or *after*, which satisfies the NON-CROSSING condition. This holds as well for the `clusterSTEM` operation. Further any EPM is clustered to a cluster from $\mathsf{CANDSET}_k$ which satisfies the distance condition. Thus, all clusters in CANDSET are clusters according to definition 5.1.1.

Summarizing, the clustering algorithm is able to find clusters according to definition 5.1.1. Experimental results for two pairs of RNAs are given in chapter 6.

Theorem 5.4.1 (Clustering Algorithm)
Given two nested RNAs $\mathcal{R}_1 = (S_1, B_1)$ *and* $\mathcal{R}_2 = (S_2, B_2)$ *with* $n = |S_1|$, $m = |S_2|, n \leq m$. *Further a set of exact pattern matches* $\mathbf{E}_\gamma^{1,2}$ *with* $p = |\mathbf{E}_\gamma^{1,2}|$. *Under a given distance threshold* τ *and a distance function* δ, *the proposed clustering algorithm determines correct a maximal cluster* $\mathcal{C}_{\mathsf{MAX}}^{\delta,\tau}$ *of exact pattern matches for a chosen clustering strategy. The number of performed operation can be estimated with* $(2 \cdot p^2 \cdot (\tau+1) \cdot c)/n$ *for some constant* c.

Chapter 6

Results

On the basis of a predetermined set of exact pattern matches (EPMs), we have developed two methods for pairwise comparison of RNA secondary structures. EPMs describe exact sequential and structural similarities between two RNA secondary structures. However, such a precomputed set contains both overlapping and crossing exact pattern matches and approaches are needed to combine EPMs in a meaningful way.

The first method described in chapter 4 deals with the task to find the best global subset of EPMs, i.e. the subset which forms a longest common subsequence. We call this problem LCS-ERP and proposed an $O(n^2m^2)$ time and $O(nm)$ space dynamic programming algorithm to solve it. This approach can be used to describe a global similarity between two RNAs. The clustering strategy described in chapter 5 follows a more local approach to combine different EPMs. Although our proposed strategy does not ensure optimal clusters, it is flexible enough to find reasonable clusters in a reasonable time.

In the following we apply both methods to two Hepatitis C virus internal ribosomal entry site RNAs as well as to two bacterial 16S ribosomal RNAs to demonstrate their usability. For a validation of our results, we compare them to `RNA_align` and `RNAforester`. This chapter starts with a description of our implementation as basis for the later given experimental results.

6.1 Implementation of LCS-ERP and Clustering

In order to verify the functioning and usability, we implemented both the algorithm for finding the longest common subsequence of exact RNA pattern and the clustering strategy. This program was developed during this thesis and all below given results were obtained from this program. Our C++ implementation combines all necessary steps in a modular and object oriented way. The core of the program builds the object `EPMensemble` for the storage of all overlapping and crossing exact pattern matches (EPMs). The necessary EPMs are discovered according to the fast maximum common substructure algorithm from section 3.4 within the object `EPMfinding`. An implementation of this algorithm was obtained from [SB07]. The algorithm for the problem LCS-ERP is implemented in the object `EPMlcserp` and the clustering strategy is given in `EPMclustering`.

The input data can be provided in two ways. First, the two RNAs can be given with their sequences and structures in standard dot-bracket format. If no secondary structure is available, the program interacts with the Vienna RNA package library [HFS+94]

and calculates the minimum free energy (mfe) structure with `RNAfold` automatically [HFS+94]. The mfe-program uses the DP-algorithm from Zuker and Stiegler [ZS81] in combination with the equilibrium partition function to calculate base pair binding probabilities from McCaskill [McC90]. The thermodynamic parameters are taken from [MSZT99, WTK+94].

As it is an additional goal of this thesis to find a visualization for the exact pattern matches itself and of course for the resulting sequences of EPMs, we developed two ways to achieve this. First, it is possible to indicate a certain set of EPMs within a secondary structure. The plot for the structure is obtained as well from the Vienna package with the interface for the postscript output. An additional annotation script was implemented to highlight all EPMs in the structure plot with a different color. See for example figure 6.1 for such an output. Second, a dot plot is useful to illustrate pairwise EPMs within a single plot. Such an example was already given in figure 3.5 which shows a complete set of EPMs for two RNAs. For the ease of programming, we implemented this dot-plot view by means of a JAVA program. We choose an structured XML file to exchange all necessary data between the programs.

The LCS-ERP algorithm is completely implemented and the clustering strategy is able to calculate clusters with both clustering strategies CLUSTER-MAX-1 and CLUSTER-MAX-2. As distance function we have already implemented δ_{SEQ1} and δ_{EQL}. In order to determine the set $\mathbf{E}_{\gamma}^{1,2}$ with the given properties, one can provide for example the option -s# to indicate the minimal size γ.

6.2 Comparison to other Methods

For the comparison of the results we have chosen `RNA_align` and `RNAforester`. The first method computes sequence structure alignments according to the general edit distance algorithm described in section 2.4 [JLMZ02]. An implementation is available at the website `http://www.csd.uwo.ca/~bma/rna_align/`. For both applications we have computed the alignment with the scoring scheme $(\omega_m, \omega_d, \omega_{am}, \omega_b, \omega_a, \omega_r) = (1, 1, 1.8, 1.5, 1.75, 2)$.

Second, the `RNAforester` program from [HTGK03] is selected. This approach is build upon the tree alignment algorithm for ordered trees from [JWZ95] and extends it to calculate forest alignments. The time complexity is $O(|F_1| \cdot |F_2| \cdot deg(F_1) \cdot deg(F_2) \cdot (deg(F_1) + deg(F_2)))$ where $|F_i|$ is the number of nodes in forest F_i and $deg(F_i)$ is the degree of F_i. An implementation is available together with the Vienna RNA package. As input parameter the algorithm needs the scores for the different edit operations. We have chosen $b_m = 4$ (base-match), $b_r = -2$ (base-mismatch), $b_d = -2$ (base-deletion), $p_m = -4$ (base pair replacement) and $p_d = -4$ (base pair bond deletion) to compute the tree alignment for both applications.

It is important to note that a direct comparison with these methods is not possible due to the fact that our approach contains only exact matchings, whereas alignments contain additional gaps and mismatches. Nevertheless, a comparison is possible on the base of the exact matchings. After the computation of the alignments with the mentioned methods, we extract all positions with exact sequence structure matchings. The comparison is now

achieved via the intersections and differences with our approach. This enables further a similar visualization technique for the alignments to our results. Clearly, a validation of global alignments is only reasonable with the solution determined by LCS-ERP. Although the clustering approach yields more local solutions, a comparison to local alignments is not possible due to the nature of this approach. However, we can show that the obtained clusters are reasonable as well. The comparison can be found at the end of section 6.3.1 and 6.3.2.

6.3 Application of LCS-ERP and Clustering

In the following we analyse both methods with two test cases. We have chosen at first two RNAs with a length of around 400 nucleotides and the second pair comprise RNAs with about 1550 nucleotides. In detail, the analysis was carried out with the following RNAs:

- **App. 1:** Two RNAs from different Hepatitis C viruses, which belong both to the Rfam family HCV_IRES for internal ribosomal entry sites (IRES) [GJMM$^+$05]. IRES elements binds to the 40S ribosomal subunit and initiate the translation of the viral mRNA. Specific structural properties are necessary for their functioning. Between different IRES elements one could expect several similarities. The first RNA comprises bases 1-379 (GenBank code: AF165050) and the second comprises the bases 1-391 (GenBank code: D45172). The secondary structures were found via `RNAfold`.
- **App. 2:** Two 16S ribosomal RNAs from different organisms. The first RNA is from Eschericia coli and is 1541 bases long (GenBank code: J01859). The second RNA is from Dictyostelium discoideum and is 1551 bases long (GenBank code D16466). This organism is an eukaryotic slime mould, whereas E. coli belongs to the prokaryotic proteobacteria. This imply that both organisms evolved a long time separately. Nevertheless one could expect significant similarities between both RNAs due to the fact that the RNAs are ribosomal RNAs. The secondary structures were taken from the Comparative RNA Web (CRW) site [CSS$^+$02].

Note, that the first pair was chosen according to the paper from Siebert and Backofen [SB07]. All computations were carried out on a Pentium M with 1.4 GHz and 768Mb RAM.

6.3.1 Hepatitis C Virus IRES RNAs

For this case, the algorithm `EPMfinding` identified 3284 exact pattern matches in 0.44 seconds. We use the complete set $\mathbf{E}_{\gamma}^{1,2}$ as input for both pairwise comparison methods.

LCS-ERP applied to Hepatitis C Virus IRES RNA

Our LCS-ERP algorithm obtained a set of 26 exact pattern matches with a sequence of 175 nucleotides as longest common subsequence of exact RNA patterns. The LCS-ERP corresponds to a sequence coverage of about 45%. The calculation was performed in 0.53 seconds.

The structures with the indicated LCS-ERP can be seen in figure 6.1 below. Although the structures differ significantly in their shape, the algorithm detects several similar regions between both structures. The numbers mark the five largest EPMs from the set $\mathbf{E}_{\gamma}^{1,2}$. These are the same as the manually marked EPMs shown in figure C.1 in the appendix on page 109. The solution for LCS-ERP includes all of them automatically. An intersting detail is for example the included small blue hairpin in the left structure between number three and four. In the right RNA, this hairpin is opposite to the small yellow stem with number five, whereas in the left structure this stem is situated in another region. Nevertheless, all indicated EPMs are Non-Crossing in order to build the LCS-ERP.

Appl. 1	length	#EPMs	time
LCS-ERP	175	26	$0.53s$

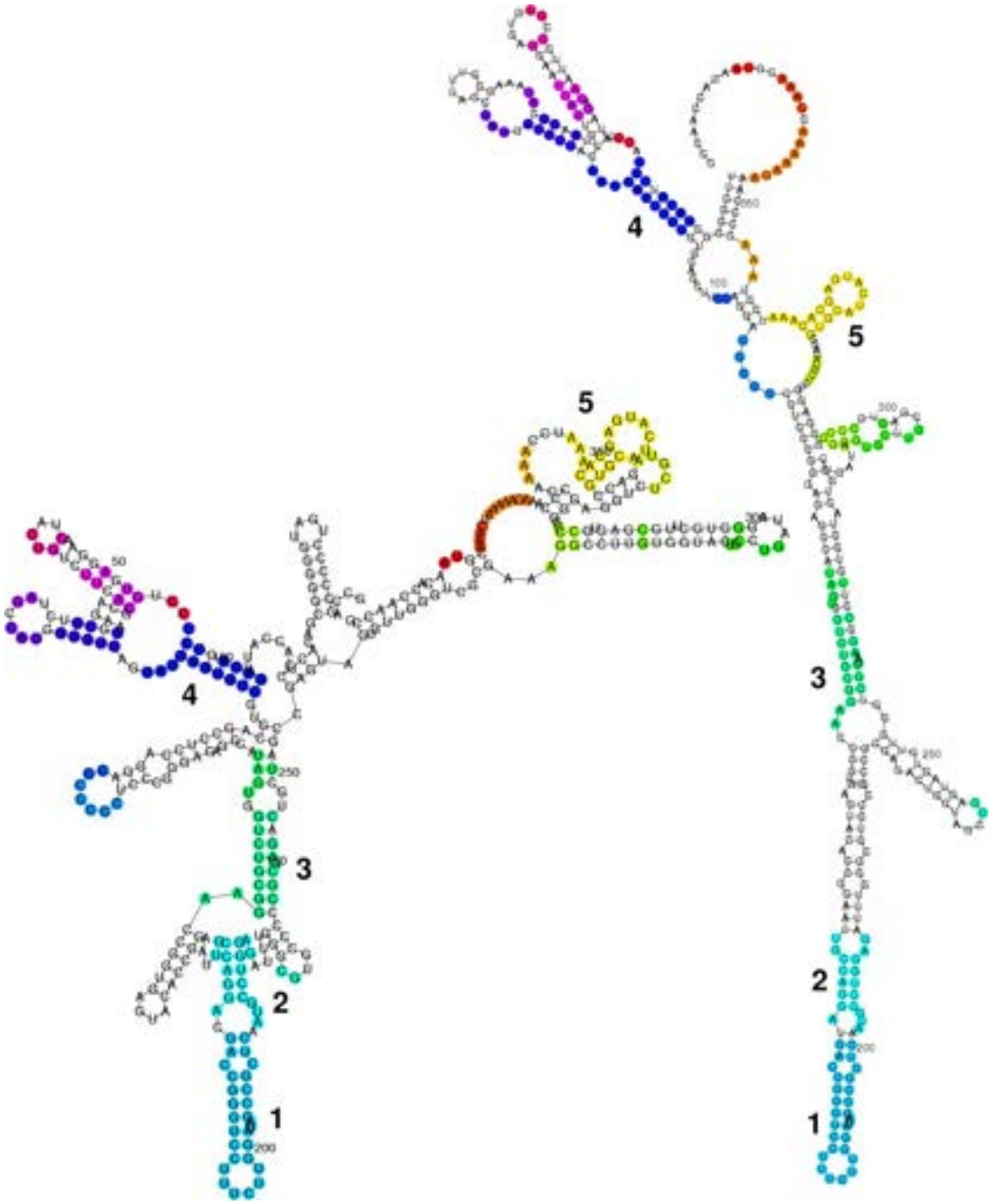

Figure 6.1: LCS-ERP approach applied to two Hepatitis C virus internal ribosomal entry sites (IRES) RNAs. The colored nucleotides represent the found LCS-ERP with an overall length of 175 bases. Each EPM is shown in a different color. The numbers indicate the five marked EPMs from figure C.1. GenBank codes: D45172 (left RNA), AF165050 (right RNA)

Clustering applied to Hepatitis C Virus IRES RNAs

Due to the different clustering strategies and distance functions, the solutions for this approach depends on the chosen parameters. Therefore we show for the first application the differences between the clustering strategies CLUSTER-MAX-1 and CLUSTER-MAX-2. For comparability, we focus always on the best cluster $\mathcal{C}_{max}^{\delta,\tau} \in$ CANDSET, i.e. the cluster with the largest size, in the analysis. Furthermore we change the threshold value τ for both strategies in the same way. The threshold determines the maximal allowed distance between exact pattern matches from a cluster. The used distance function is δ_{SEQ1}.

CLUSTER-MAX-1 vs. CLUSTER-MAX-2

To assess the general abilities as well as the differences of both strategies, we analyse this test case with a series of different threshold values τ. The results are summarized in table 6.1. The corresponding structures for CLUSTER-MAX-1 are shown in figure 6.2 and B.4. The structures for CLUSTER-MAX-2 can be found in figure 6.3 and B.5.

The data show that with an increasing search range by the threshold value τ the size of the cluster $\mathcal{C}_{max}^{\delta,\tau}$ increases as well. For larger values than $\tau = 80$ the largest clusters do not change anymore in this case. Further one can see that both strategies are able to find significant clusters. For example, the lower stem with the two large EPMs is matched for nearly all τ values. For small values of τ, the best cluster is a proper local cluster for both strategies, for larger τ the cluster spans nearly the whole structure.

By comparison of both clustering strategies differences in size and shape of the found clusters emerge. For larger values of τ clearly CLUSTER-MAX-2 yield larger and better clusters. Compare the pictures for $\tau = 80$ in the figures 6.2 and 6.3. The large green matching part of the best CLUSTER-MAX-2 solution is completely skipped in the best CLUSTER-MAX-1 solution. An notable difference to the global solution occurs for CLUSTER-MAX-2 with $\tau = 80$. The small pink hairpin is not part of the global solution. From figure 6.3 one can verify that this is an reasonable matching as well. A summary of all clustering parameters as well as remarks to the running time can be found at the end of this chapter in section 6.3.3.

threshold τ	size $\mathcal{C}_{MAX1}^{\delta,\tau}$	time in s	size $\mathcal{C}_{MAX2}^{\delta,\tau}$	time in s
5	48	0.08	48	0.49
10	48	0.38	63	0.53
20	72	0.41	89	0.58
40	73	0.49	112	0.63
60	74	0.55	116	0.71
80	74	0.59	119	0.73

Table 6.1: Comparison of the clustering strategies CLUSTER-MAX-1 and CLUSTER-MAX-2 for two Hepatitis C virus IRES RNAs. For larger thresholds τ, the cluster size increases for both strategies. CLUSTER-MAX-2 reveal larger clusters in general. The running times are comparable of both strategies.

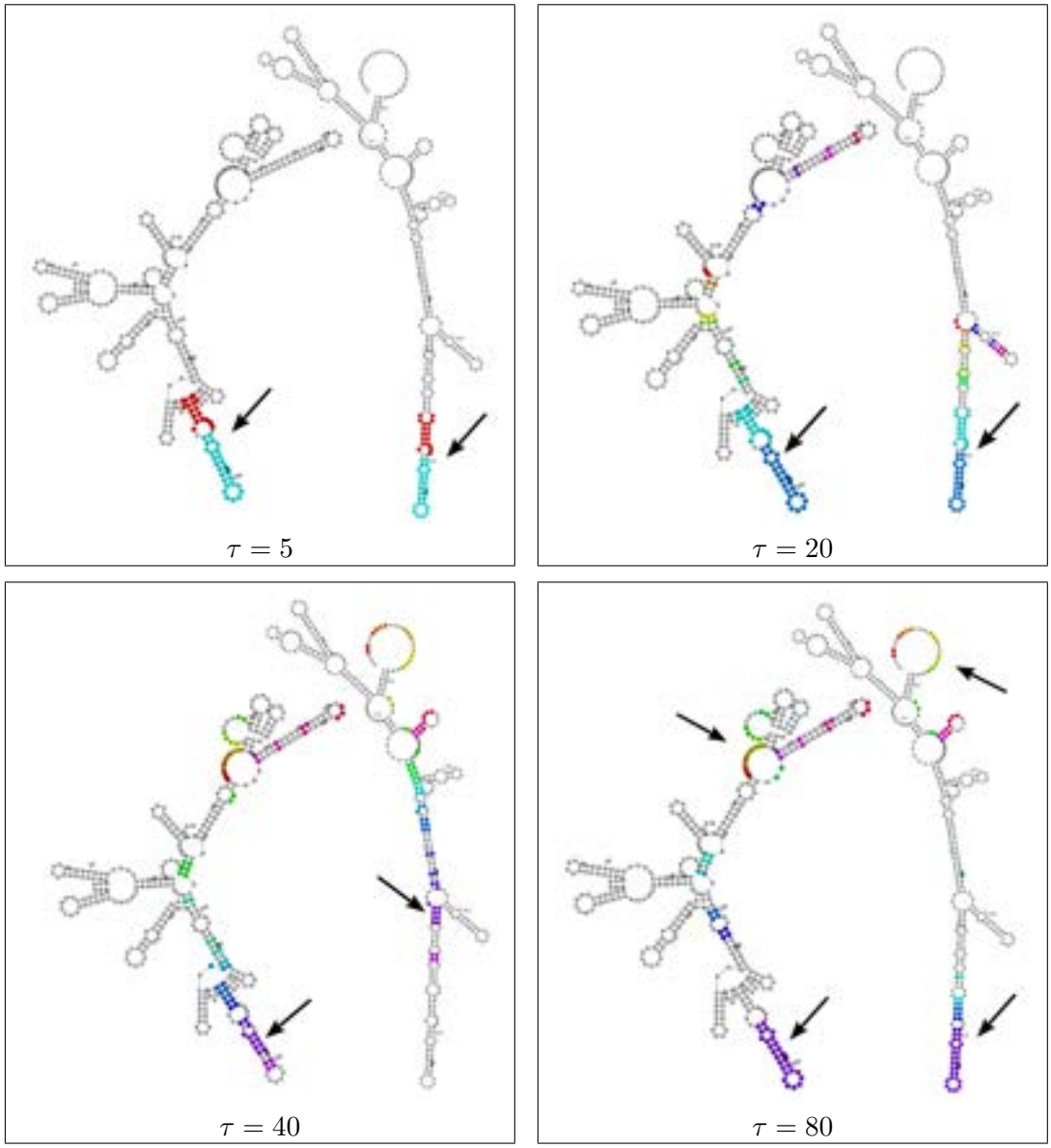

Figure 6.2: Analysis for two Hepatitis C virus IRES RNAs with clustering strategy CLUSTER-MAX-1. The pictures show the largest found cluster for the given distance threshold τ value. Each EPM is shown in a different color. The arrows indicate regions with large matchings. For example look at the figure for $\tau = 40$. The arrow indicates a large EPM which is not part of the other solutions.

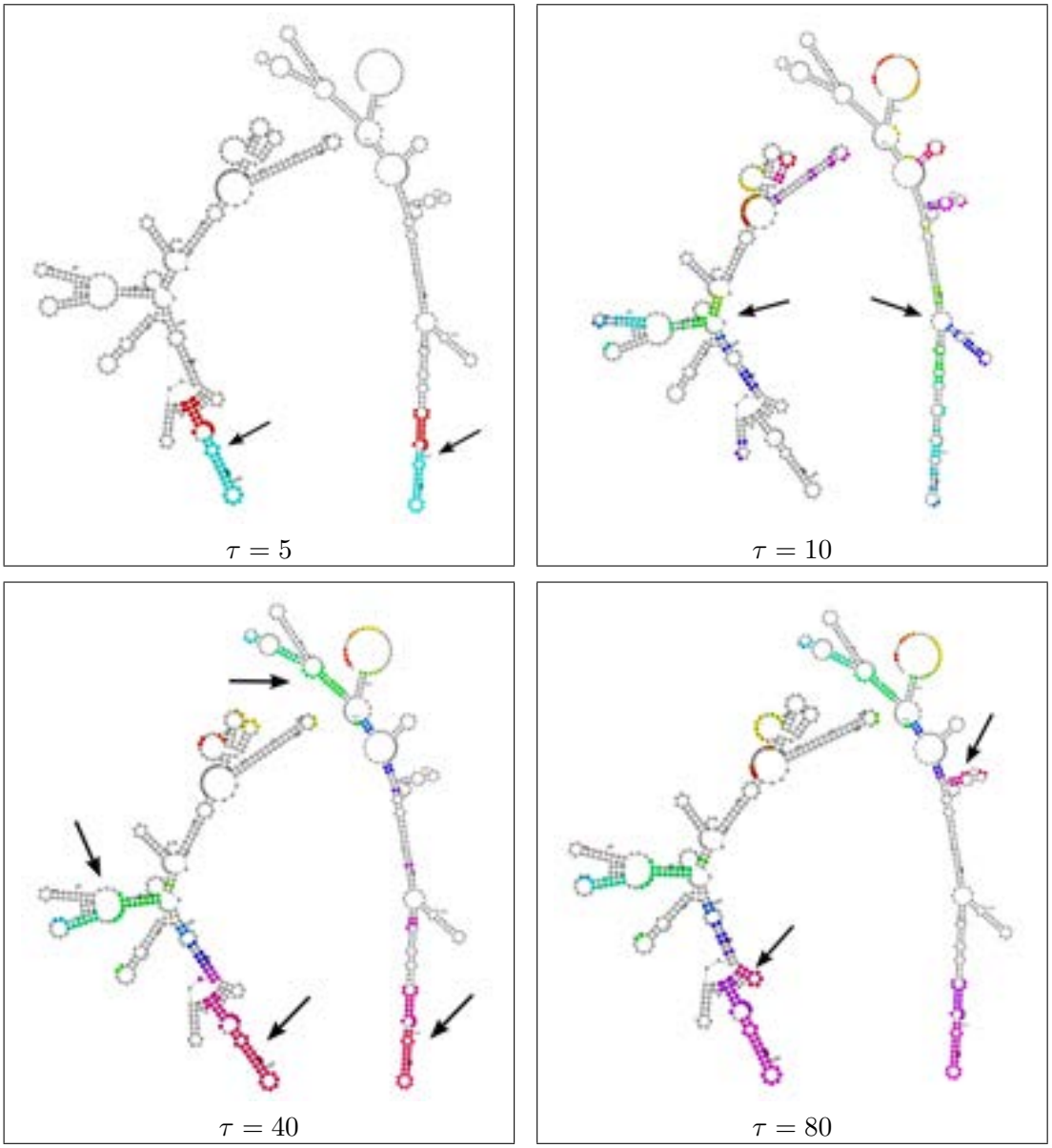

Figure 6.3: Analysis for two Hepatitis C virus IRES RNAs with clustering strategy CLUSTER-MAX-2. The pictures show the largest found cluster for the given distance threshold τ value. Each EPM is shown in a different color. The arrows indicate regions with interesting matchings. For example, look at the figure for $\tau = 10$. The multi-loop with the green and blue EPMs is matched completely different to the other solutions. Second, look at the figure for $\tau = 80$. The small red stem-loop is neither part of other clusters nor part of the LCS-ERP solution.

Comparison with RNAforester and RNA_align

For the comparison of our results to RNA_align and RNAforester, we have first computed the alignments for both Hepatitis C virus IRES RNAs. The obtained alignments for the Hepatitis C virus IRES RNAs can be found in the appendix in figure A.1 for RNA_align and in figure A.3 for RNAforester. Next, we have extracted from these alignments all positions with exact sequence structure matchings.

RNA_align has found an alignment with 192 exact matchings in which 159 matchings intersect with the solution from LCS-ERP. This means that about 9% of our matchings are different to RNA_align and the LCS-ERP solution covers 82.8% of the exact matchings from RNA_align. Figure 6.4 below shows the structural comparison between RNA_align and LCS-ERP. One can see that all important regions are blue colored which means identical exact matchings. Further, there are several light blue colored nucleotides. These positions are part of exact matchings in both methods, but the positions in the other RNA are different. With a sharp view one can see that these are mainly shifted matchings in a loop region. The red colored nucleotides indicate additional exact matchings found by RNA_align. These are mainly single nucleotides or regions which have to be excluded in LCS-ERP due to the Non-Crossing constraint.

The RNAforester approach obtained an alignment with 128 exact matchings. In comparison to our LCS-ERP approach, we cover the included exact matchings with a similar rate of 80.5%. However, our method finds much more exact matches in general. The structural comparsion of our approach with the alignment found by RNAforester is shown in figure A.2 in the appendix. One can see that our approach additionally matches one stem region and several loop regions (green colored positions). Further there are only a few different matches found by RNAforester. These are mainly single nucleotides, which cannot be part of our solution.

The following table 6.2 summarizes the comparison. The comparison of the running times yields good results for our approach as well. Please note that the running time for LCS-ERP includes the time additionally needed to determine all exact pattern matches.

exact matches	RNA_align	RNAforester	LCS-ERP	time
RNA_align	192	-	-	$62s$
RNAforester	-	128	-	$5.4s$
LCS-ERP	159 (82.8%)	103 (80.5%)	175	$0.97s$

Table 6.2: Comparison of the number of found exact matchings by LCS-ERP and two alignment methods. For this application, the LCS-ERP approach yields good results in comparison to RNA_align. Further, the LCS-ERP approach finds significantly more exact matchings than RNAforester. The rate of about 80% identical matchings supports the significance of the LCS-ERP method. The running time is faster as well.

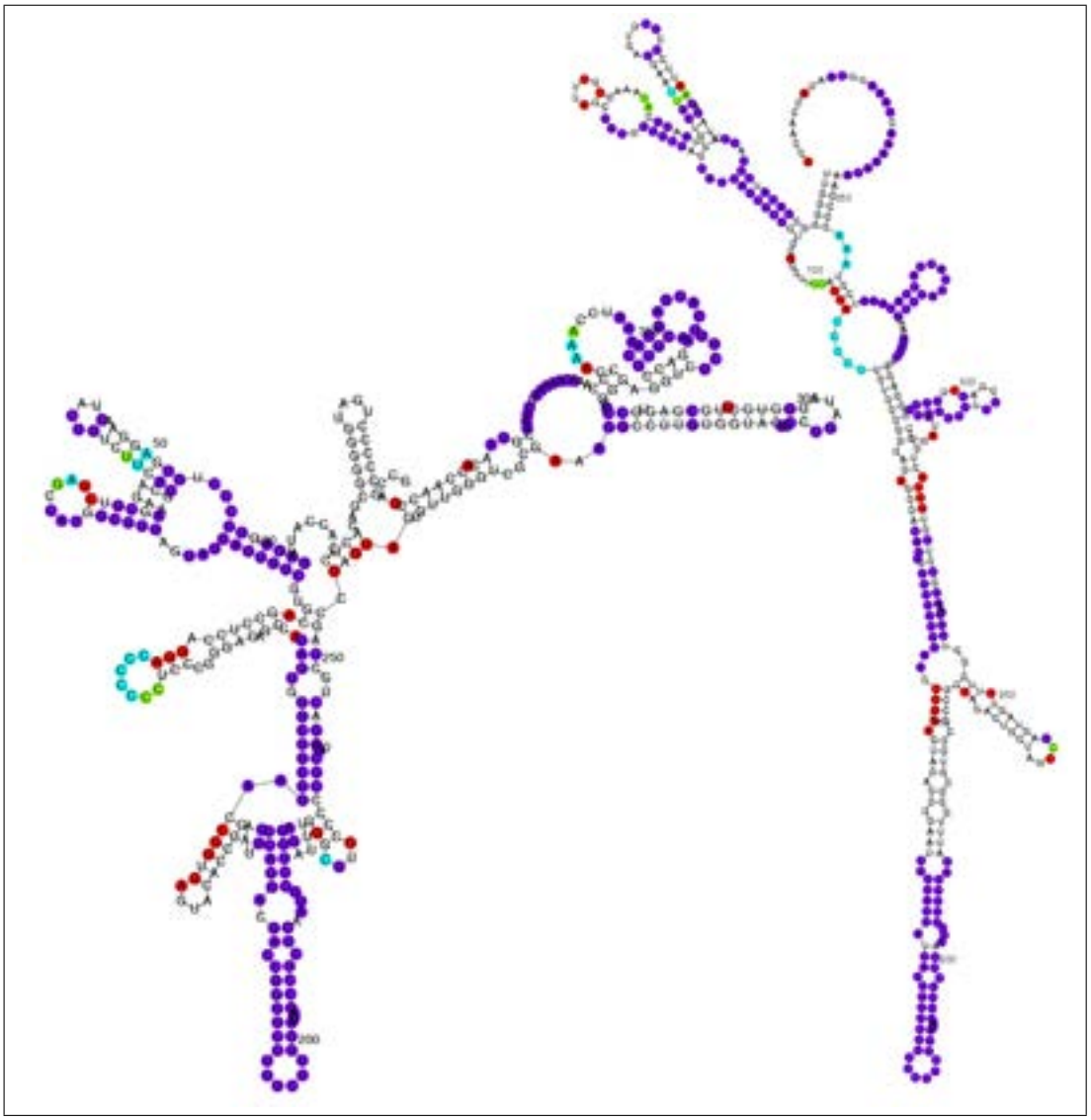

Figure 6.4: Comparison between LCS-ERP and `RNA_align` for two Hepatitis C virus IRES RNAs. The blue colored nucleotides denote exact matches in both methods. The light blue nucleotides denote matched nucleotides in both methods, but with different positions in the other RNA. The green colored nucleotides occur only in LCS-ERP and the red colored nucleotides occur only in `RNA_align`. The picture shows that the LCS-ERP solution is comparable to the solution from `RNA_align`. GenBank codes: D45172 (left RNA), AF165050 (right RNA)

6.3.2 16S Ribosomal RNAs

For this case, the algorithm `EPMfinding` identified 50322 exact pattern matches in 1.21 seconds. We use the complete set $\mathbf{E}_{\gamma}^{1,2}$ as input for both pairwise comparison methods.

LCS-ERP applied to two 16S ribosomal RNAs

The LCS-ERP algorithm obtained a set of 159 exact pattern matches with a sequence of 875 nucleotides as longest common subsequence of exact RNA patterns. This corresponds to a sequence coverage of about 57%. The calculation was performed in 15.7 seconds.

The structures with the indicated LCS-ERP can be seen in figure 6.5. One could see that our approach is capable to identify a large number of exact similarities between both RNAs. See especially the high number of matched hairpins and internal loops. Due to the fact that ribosomes are one of the best conserved functional units in living organisms, the overall shape of both secondary structures is very similar. However, in detail one could recognize differences in stem and loop regions. The matched regions could therefore reflect conserved functional units. Unbound nucleotides within loops can clearly easier interact with other molecules than paired nucleotides.

Appl. 2	length	#EPMs	time
LCS-ERP	875	159	15.7s

Clustering applied to 16S ribosomal RNAs

According to the first application, we show at first differences between the clustering strategies CLUSTER-MAX-1 and CLUSTER-MAX-2. Further we focus here on differences between the distance functions δ_{SEQ1} and δ_{EQL}. For comparability, we use always the best cluster $\mathcal{C}_{\mathsf{MAX}}^{\delta,\tau}$ in the analysis. Due to the larger structures, we show in some cases only cutouts to illustrate properties of the found clusters.

Clustering Strategies CLUSTER-MAX-1 and CLUSTER-MAX-2

First we have applied both clustering strategies in combination with distance function δ_{SEQ1} for different distance threshold values τ. For the differences between both strategies we focus on the case with $\tau = 50$. The according structures with the highlighted cluster is shown in figure B.1 for CLUSTER-MAX-1 and in B.2 for CLUSTER-MAX-2 in appendix B. Table 6.3 below summarizes the results.

First, one can see from figures B.1 and B.2 that both strategies are able to identify significant similarities between both RNAs. Further the data exhibit similar differences between the strategies like for the first application. CLUSTER-MAX-2 produces larger and better clusters than CLUSTER-MAX-1. Besides their size, the cluster from CLUSTER-MAX-2 also contains more exact pattern matches in a specific region. Compare for example the right branched stem region for both strategies in figure B.1 and B.2. The EPMs are

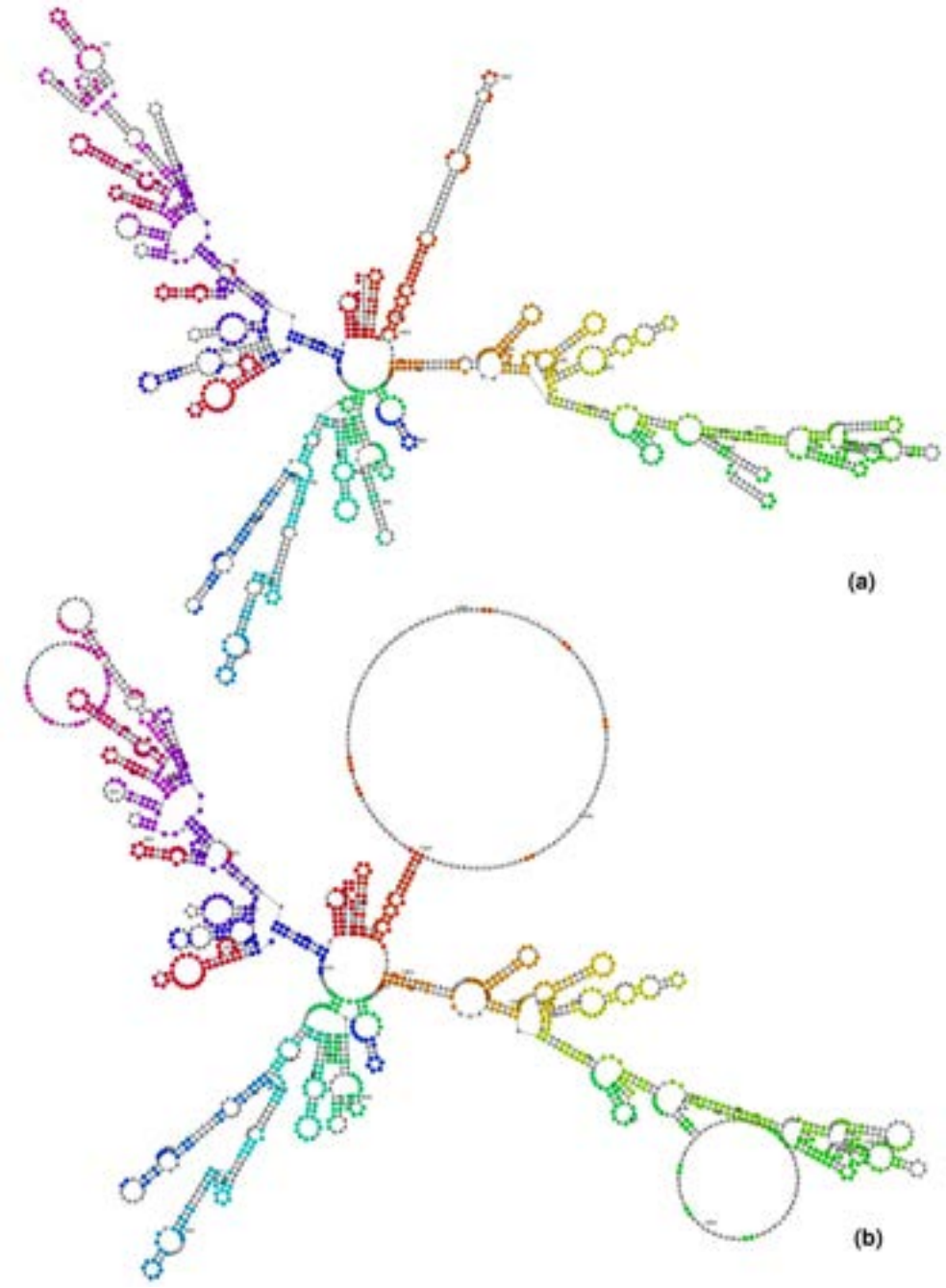

Figure 6.5: LCS-ERP approach applied two 16S ribosomal RNAs. The colored nucleotides represent the found LCS-ERP with an overall length of 875 bases. Each exact pattern match is shown in a different color. One could recognize several regions of similarity. Although the structures vary, nearly all stems and hairpins can be found with significant matches in both RNA. (a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

more dense and the cluster for CLUSTER-MAX-2 contains more matched hairpins in this region. CLUSTER-MAX-1 tends to match base pairs only in the main branch and ignore small branched stems.

One can see from the table that higher distance thresholds τ not necessarily increase the size of the largest found cluster. Further the table shows that the running time increases with higher distance thresholds τ, but the running time of CLUSTER-MAX-1 scales better than CLUSTER-MAX-2. For small τ values the differences are negligible which supports the choice of CLUSTER-MAX-2 as the better strategy. Moreover, this strategy finds reasonable clusters for small distance thresholds with a fast running time. See the already large cluster obtained for $\tau = 10$ shown in figure B.3.

threshold τ	size $\mathcal{C}^{\delta,\tau}_{\mathsf{CMAX1}}$	time in s	size $\mathcal{C}^{\delta,\tau}_{\mathsf{CMAX2}}$	time in s
10	155	4.22	403	4.92
50	183	9.24	555	12.64
100	286	18.63	483	29.41

Table 6.3: Comparison of the clustering strategies CLUSTER-MAX-1 and CLUSTER-MAX-2 for two 16S rRNAs. CLUSTER-MAX-2 revels significant larger clusters than CLUSTER-MAX-1. The running time of CLUSTER-MAX-2 is little slower than CLUSTER-MAX-1.

Distance Functions δ_{SEQ1} and δ_{EQL}

In section 5.2, we introduce the distance function δ_{EQL} as modification of δ_{SEQ1} in order to obtain more similar clusters. This function needs in addition to the distance threshold τ the second parameter Δ_{DT} as the threshold of the allowed distance differences for two EPMs in the considered RNAs.

The analysis is carried out with clustering strategy CLUSTER-MAX-2 and a fixed distance threshold $\tau = 10$. The largest cluster with the unmodified distance function δ_{SEQ1} is shown in figure B.3 in the appendix. The cluster has a size of 403 matched nucleotides. One can see that this cluster exhibits already a better locality in comparison to the cluster obtained with $\tau = 50$ shown in figure B.2. Further the cluster includes significant matches like the green colored stem.

The application of distance function δ_{EQL} to this case reveals some differences. Figure 6.6 shows the results for two different values Δ_{DT}. In both cases one can see some effects. Especially for $\Delta_{\mathsf{DT}} = 2$ the best found cluster decreases in size with the result of a more local cluster. However, one can see from the picture that already for $\Delta_{\mathsf{DT}} = 5$ the effect decreases with higher thresholds.

The best cluster for $\Delta_{\mathsf{DT}} = 2$ has a size of 172 nucleotides and the cluster for $\Delta_{\mathsf{DT}} = 5$ comprises 368 nucleotides. This is a clear reduction in comparison to the best cluster for δ_{SEQ1} with $\tau = 10$ and a size of 403 nucleotides.

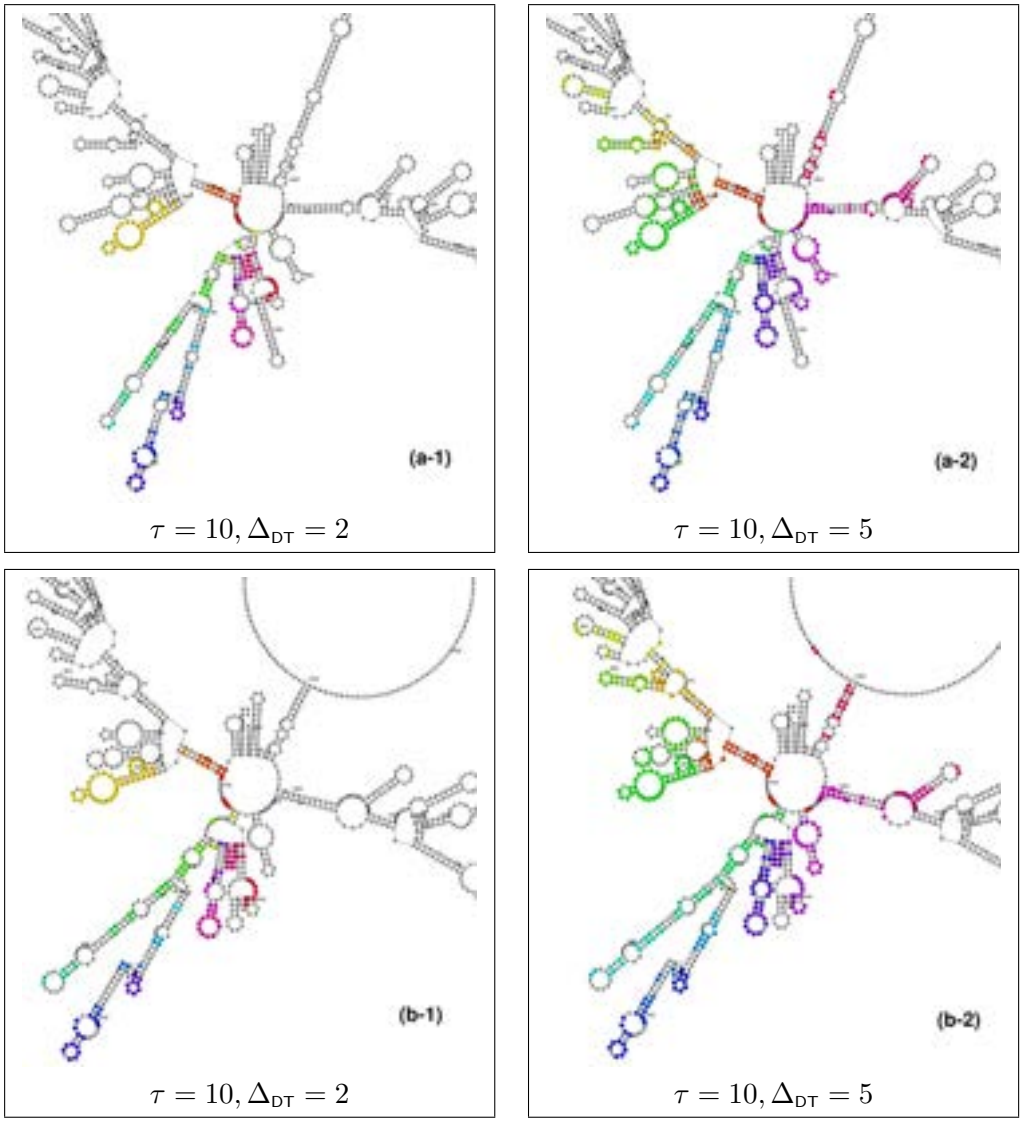

Figure 6.6: Clustering with distance function δ_{EQL}. The pictures show the largest found cluster for the given distance threshold τ value. Δ_{DT} is the allowed difference threshold. In comparison to δ_{SEQ1}, the clusters are more local. For smaller values of Δ_{DT} the effect is higher than for larger values. (a-1,2) E. coli 16S rRNA (J01859), (b-1,2) D. discoideum 16S rRNA

Δ_{DT}	size $\mathcal{C}^{\delta,\tau}_{CMAX2}$	time in s
2	172	4.2
5	368	4.8

Table 6.4: Results for two 16S rRNAs with distance function δ_{EQL} for $\tau = 10$ and two different values for Δ_{DT}. The best found clusters for $\Delta_{DT} = 2$ is smaller, but exhibits a better locality in comparison to $\Delta_{DT} = 5$ and to δ_{SEQ1} with $\tau = 10$.

Comparison with RNAforester and RNA_align

Similar to the first application, we have first computed the alignments for both 16S rRNAs. The obtained alignment for RNA_align can be found in figure A.5 and for RNAforester in figure A.6 in the appendix. Next we have extracted from these alignments all positions with exact sequence structure matchings.

The alignment from RNA_align contains 861 exact sequence structure matchings. In comparison, our approach obtained 875 exact matchings and of these are 688 or 79.9% equal to the alignment. In contrast to the first application, our method finds in fact some more matchings than RNA_align. The structural comparison of both methods is shown in figure A.4 in the appendix. The blue colored nucleotides show that our method conforms very well with the alignment. In addition, there are about 70 nucleotides in both structures which are covered by both methods, but mapped only to different positions.

The RNAforester approach obtained an alignment with 847 exact matchings. In comparison to our LCS-ERP approach, we cover the included exact matchings with a rate of 82.6% (700). Similar to the first application, our method finds several more exact matches in general. The structural comparsion of our approach with the alignment found by RNAforester is shown in figure 6.7. It shows that large parts are matched similar by both methods (blue colored nucleotides). Further there are 61 nucleotides in the E. coli RNA and 49 nucleotides in the D. discoideum RNA which are matched to different positions. These are often only shifted matchings. The number of unique matchings for the LCS-ERP methods is as well higher than for RNAforester.

The following table 6.5 summarizes the comparison. For the used 16S rRNA we achieved a slight better result with RNAforester. The comparison of the running times yields really good results for our approach as well and emphasizes its application to large RNAs. Please note that the running time for LCS-ERP includes the time to determine all exact pattern matches. Due to the high memory usage of the RNA_align implementation for long RNA sequences, we used a compute server for this computation. Nevertheless, the computation needs more than 1.5 hours.

exact matches	RNA_align	RNAforester	LCS-ERP	time
RNA_align	861	-	-	$1h\,35m^*$
RNAforester	-	847	-	$7m\,25s$
LCS-ERP	688 (79.9%)	700 (82.6%)	875	$16.9s$

Table 6.5: Comparison of the number of exact matches found by LCS-ERP and two alignment methods. For this application, the LCS-ERP approach yields good results in comparison to both methods. Further, the LCS-ERP approach finds the most number of exact matchings. The rate of about 80% identical matchings supports the significance of the LCS-ERP method. The running time is remarkable faster than both methods. *AMD Opteron 275/875, 2.2 GHz, 20 Gb RAM

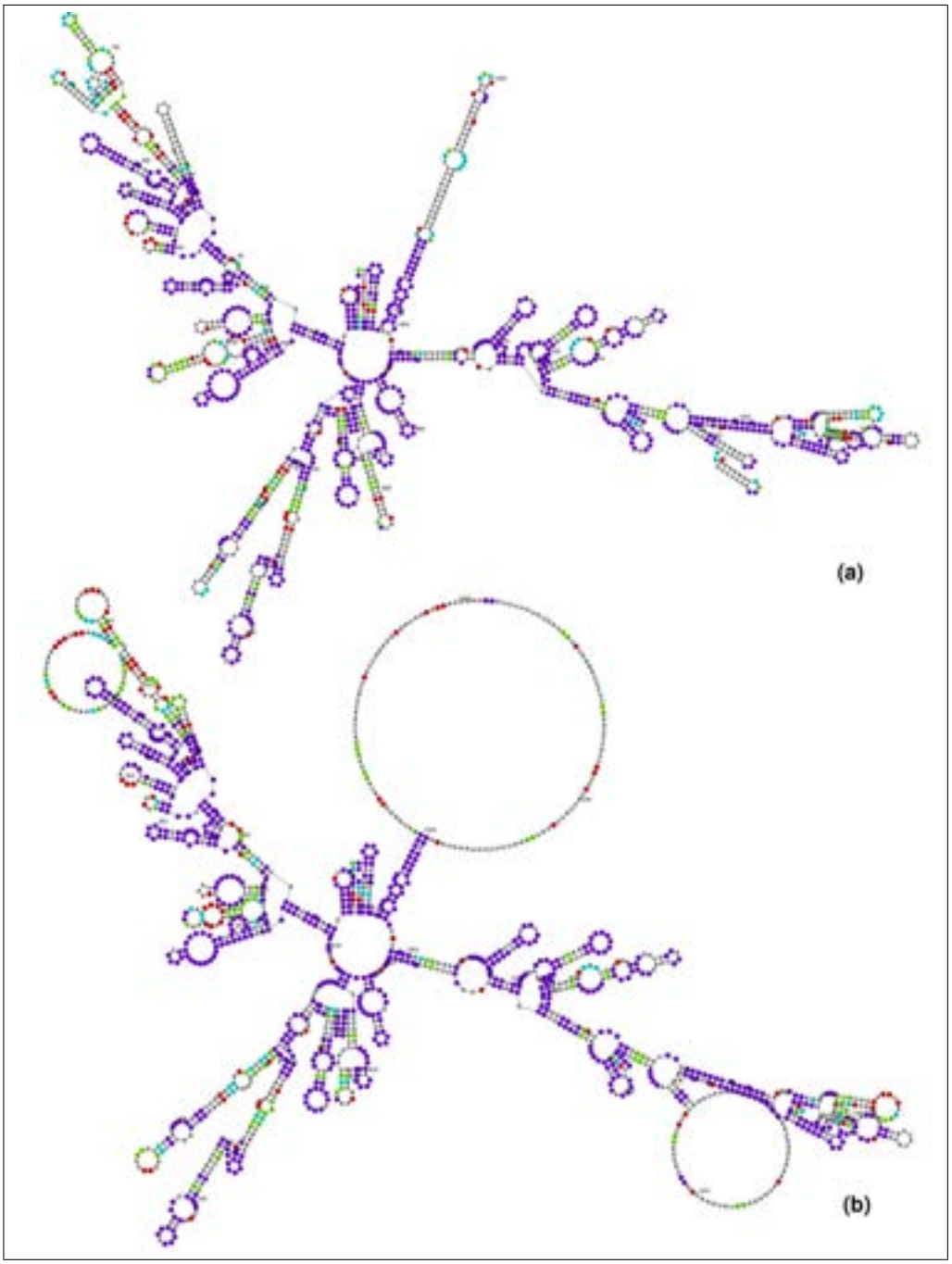

Figure 6.7: Comparison between LCS-ERP and `RNAforester` for two 16S rRNAs. The blue colored nucleotides (700) are exact matches in both methods. The light blue nucleotides denote matched nucleotides by both methods but with different positions in the other RNA (61 in (a), 49 in (b)). The green colored nucleotides occur only in LCS-ERP and the red colored nucleotides occur only in `RNAforester`. The picture shows that the LCS-ERP solution is comparable to the solution from `RNAforester`.(a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

6.3.3 Summary for the Clustering Parameters

From both applications arise some aspects which parameters are useful or not for the clustering approach. First, if one is interested in a high number of matches, clearly the CLUSTER-MAX-2 strategy reveals the larger clusters. Furthermore, these clusters include more significant matches of stem and loop regions. The CLUSTER-MAX-2 strategy enables that an already produced cluster can be extended in different ways. This advantage leads at the end to larger clusters.

Second, the choice of the distance threshold τ has a great influence on the resulting clusters as well. With small values one can achieve more local clusters, whereas clusters from higher values tend to span large parts of the secondary structure. These clusters are more similar to the solution from the global LCS-ERP approach. Moreover, the data show that too high distance thresholds are nearly useless. Over a certain value, the overall size of the largest cluster is not increasing. Depending on the RNA secondary structure, the size can also decrease again.

The choice of the distance function clearly depends on the application. Normally, we recommend the standard function δ_{SEQ1}. The data show that this function is already capable to detect large and significant clusters of exact pattern matches. Further, we suggest the usage of function δ_{EQL} for the search of specific formed clusters.

The running time with strategy CLUSTER-MAX-1 is slightly better than for CLUSTER-MAX-2. This is obvious, as this strategy produces fewer candidate clusters. The highest influence on the running time is given by the choice of the distance threshold τ. Therefore we propose the choice of a sufficiently small τ value with the balance between running time and cluster size. Nevertheless, the data show that already small values of τ produce good and large clusters with a fast running time even for large RNAs.

6.4 Discussion of Results

In this thesis we have developed the LCS-ERP approach and the clustering approach for pairwise sequence structure comparison of RNAs on the basis of exact pattern matches. In the previous sections we have applied both methods to two different pairs of RNAs and presented the corresponding results. This section discusses the achieved results and gives pros and cons for both developed methods.

6.4.1 The LCS-ERP Approach

This approach was developed to utilize exact pattern matches for the detection of global similarities between two RNAs. In order to show the potential of this method for pairwise comparison, we have applied the presented dynamic programming algorithm from chapter 4 to two Hepatitis C virus IRES RNAs and two 16S ribosomal RNAs.

According to the shown results for the LCS-ERP approach, one can state for both applications that the found solutions express global similarities due to the high number of included exact matchings. Although the secondary structures differ, the found subsets

of EPMs unveil similar regions for both pairs of RNAs (see figure 6.1 on page 77 and figure 6.5 on page 84). The high significance is achieved due to the matched loop and stem regions in both applications. These regions are presumable necessary for the correct functioning of the considered RNAs. In the case of the two Hepatitis C virus IRES RNAs, they could represent probable binding sites for the 40S ribosomal subunit. For the two 16S ribosomal RNAs the matchings could represent conserved structural elements in the ribosome during evolution. The high coverage of 57% of our solution in this case is also explained by the fact that both rRNAs are experimental verified. The high significance of the solution for the two Hepatitis C virus IRES RNAs is supported by the fact that the five largest exact pattern matches are included in the LCS-ERP solution.

The comparison with the standard alignment approaches `RNA_align` and `RNAforester` confirms the results. Counting only the equivalent matchings, the data show that for both applications our solution has an accuracy of about 80% in comparsion to the used alignment methods. This is a high rate, if one takes into consideration that all single nucleotide matchings are not part of our solution. Further, there are in both applications a high number of matchings which differ only in one RNA. This occurs especially in loop regions with similar consecutive bases and therefore the matchings can be easily shifted.

Comparing the overall number of exact matched nucleotides, the LCS-ERP approach achieves for both applications better results than the `RNAforester` algorithm. In the case of the LCS-ERP method applied to two Hepatitis C virus IRES RNAs our approach finds more similar regions than the `RNAforester` algorithm. For both applications the number of exact matched nucleotides is comparable to the `RNA_align` algorithm. This is definitely a good result, because the general edit distance algorithm from [JLMZ02] is the most general method for pairwise sequence structure comparison of RNAs. Clearly, the alignments can be possibly improved with a different scoring scheme, but we have already chosen a scheme with the focus on exact matches.

Discussing the running time, the LCS-ERP algorithm is fast for both applications. Even for long RNAs the global solution was found within seconds. In particular, one has to compare the running times between the LCS-ERP method and the alignment methods for the two long 16S rRNAs. The data emphasize the preference of our approach especially for long RNA sequences to alignment methods. Our approach is able to give much faster reasonable assertions about the global similarity for two RNAs.

Nevertheless, there are some drawbacks of this method as well as the proposed algorithm. First, besides the theoretical complexity of $O(n^2m^2)$ for the LCS-ERP algorithm, the precomputed input comprises only about 50000 exact pattern matches for a real scenario with long RNAs like in the second application. This is much lower than the theoretical maximal number of $|S_1| \cdot |S_2|$ and the number of holes is about the same as the number of EPMs. Therefore it is more realistic to estimate the running time with $O(H\,n\,m)$, whereby H denotes the number of holes. This corresponds better to the fast running times as well. Moreover, the presented formal description of the LCS-ERP algorithm is not based on exact pattern matches. The recursion formula can be changed to a version which really operates on EPMs instead of sequence positions. Such an algorithm would really benefit from the reduction of complexity given with the set of exact pattern matches. Third, the approach was not tested with unrelated RNAs. For this case our

approach would also find a solution but without a high biologically meaning. However, this is a general drawback of global comparison methods.

6.4.2 The Clustering Approach

This approach was mainly developed under the aspect that the detection of all exact sequence structure similarities between RNAs is possible in $O(nm)$. In order to profit from the *fast* algorithm according to the method from [SB07], our approach uses a greedy technique to reduce the overall complexity. In contrast to our global method, a goal was to detect pairwise similarities with more local properties in form of clusters of EPMs. The usage of a fast greedy strategy in favour of an optimal method is also supported by the fact that no optimal algorithm in known which detects the best cluster with a distance threshold τ. In addition, such an optimal algorithm would have had presumably at least the complexity of the optimal global algorithm. Further a greedy technique is flexible and its adoption to a different questioning is often easier.

Similar to the LCS-ERP approach, we have analyzed the performance of the clustering algorithm presented in chapter 5 with two Hepatitis C virus IRES RNAs and two 16S ribosomal RNAs. The data show that this method is able to identify significant similarities between two RNAs. The found clusters exhibit similarities between the treated RNAs. For example see the found cluster in figure 6.3 with $\tau = 80$ as well as the cluster in figure B.2.

In order to make the best of the greedy strategy, we proposed two different clustering strategies. With the focus on the cluster size as well as the significance of the matches, clearly the CLUSTER-MAX-2 strategy reveals the better clusters. The CLUSTER-MAX-2 strategy enables that an already produced cluster can be extended in different ways. This advantage compared to the CLUSTER-MAX-1 strategy leads at the end to more significant clusters.

The flexibility of the clustering approach is mainly achieved due to the different distance functions as well as the choice of the distance threshold τ. This threshold can be considered as the decisive parameter for the clustering approach. With a sufficiently small threshold the found clusters represent local similarties. For example see figure 6.3 on page 80 for $\tau = 5$ and figure B.3 on page 105. With higher thresholds, the clusters span over a large part of the structure and the solutions are more similar to the LCS-ERP approach. However, the higher the distance threshold the less changes the found cluster. Therefore, the exact choice of the distance threshold τ is important and depends on the application. The range of interesting τ values again depend on the RNAs itself. For the search of local clusters this can be assumed as disadvantage.

As an alternative distance function we proposed δ_{EQL}. This function needs a second threshold which determines the allowed differences between the distances for two EPMs. From figure 6.6 one can see that the locality is improved, but the overall cluster size is decreased as well. Similar results may be obtained with smaller distance thresholds. As a structural distance function we have proposed DISTANCE-STRUCTURE-SHORTESTPATH, but the

path finding problem needs further investigation to use this distance function. The proposed distance functions can be considered as a starting point for more sophisticated distance functions which improve both the locality and the significance of the clusters.

Concerning the running time, the data show that the algorithm finds clusters in a fast way. Especially for small distance thresholds τ the solution is found within seconds for both applications. For higher thresholds τ the running time increases and the outcomes differ sparsely. The reason is that higher distance thresholds cause that a cluster is updated to more positions in the data structure $\mathsf{CANDSET}_k$. A different data structure could avoid the large overhead of update operations. This aspect supports the choice of a small threshold in favour of a high threshold with a higher running time. Especially for large RNAs the usage of the clustering approach is therefore promising.

A comparison to other methods is pending. However, a comparison to this approach with the results obtained by the LCS-ERP approach shows that the clusters are feasible and the LCS-ERP solution can be considered as a global cluster of exact pattern matches. Further, a comparison to local alignment methods is not possible, because it is not guaranteed that the largest cluster and the local alignment cover a similar region in the RNA.

Due to the structure of the algorithm there are two general drawbacks. First, it iterates in loop regions from the left to the right which limits the structure of the distance functions. The distance in δ_{SEQ} is determined only to the right outside bound of the second EPM in order to maintain the invariant of each set of candidate clusters $\mathsf{CANDSET}_k$. The incorporation of the left outside bound can improve the results, but changes the structure of the algorithm. Second, the algorithm operates only on the first RNA. Clearly, one would select the smaller RNA as the first, but the results can be different if the order is changed.

Chapter 7

Discussion

7.1 Conclusion

In this thesis we have investigated two approaches in order to detect similarities between RNAs given with their primary and secondary structures. According to recent discoveries, RNAs accomplish with their wide range of biological functions a major role for living organisms. Similar functions are often associated with similar sequential and structural properties of the considered RNAs.

In contrast to other pairwise comparison methods, the developed methods in this thesis base solely on exact sequence-structure properties between the given RNAs. These exact substructures were obtained from the fast maximum common substructure (MCS) algorithm from [SB07] in form of exact pattern matches (EPMs). The identified EPMs comprise both a sequential and structural similarity. In biology, EPMs can represent necessary substructures of important functional motives like SECIS elements. Thus, it is interesting to know, if two RNAs have several motives in common.

Although the MCS algorithm is able to compute *all* exact pattern matches, the matchings itself overlap and cross each other. Approaches are needed which find meaningful subsets of EPMs representing pairwise similarities between the considered RNAs. Therefore, we have introduced the Non-Crossing notion as the general condition for exact pattern matches in nested RNAs. With regard to other pairwise sequence-structure comparison methods, a Non-Crossing subset of EPMs is both a plain mapping and an arc-preserving subsequence. The fact that each EPM comprises at least two nucleotides encourages their usage for a motif-based comparison.

Both methods were developed in order to show the potential of such motif based pairwise comparison methods. Second, due to the fact that the MCS algorithm detects all EPMs in $O(nm)$, the developed methods should benefit from this fast algorithm. These goals are achieved with the discussed drawbacks. The first method can be used to detect global similarities between two RNAs. Besides its theoretical complexity, the LCS-ERP algorithm is fast for real scenarios with long RNAs. The second method follows a different algorithmic approach. The clustering method is flexible enough for different applications. With the right choice of the parameters, the found clusters represent more local or more global similarities of the considered RNAs.

In general one can conclude that it seems a promising approach especially for large RNAs to precompute certain types of relationships between the considered RNAs. Such a precomputing can be used to reduce the complexity of the following comparison method. This opens the application of our approach for multiple comparison methods.

7.2 Open Problems and Future Work

Concerning the global LCS-ERP approach, the algorithm could be changed to a version which operates solely on exact pattern matches. This could improve the running time for real applications. The accuracy of the LCS-ERP approach can be increased with the incorporation of single nucleotide matchings. This could result in a third algorithmic step which aligns the regions between two consecutive exact pattern matches. Further, there are open questions in relation to the LAPCS problem. First, is it possible that algorithms for the general NP-hard LAPCS(Nested, Nested) problem can benefit from a fast precomputed set of exact pattern matches and their incorporation according to our LCS-ERP algorithm. Second, is it possible to use the LCS-ERP solution as an approximation of the LAPCS solution.

The clustering approach can be used for the improvement of local sequence structure alignment methods. For example, a cluster can be used as starting or anchor point for methods like the LSSA algorithm from [BW04]. A good "hint" can reduce its running time. Concerning the problem of local clusters in general, a first advancement is the formulation of an optimal local algorithm. The problem for this case is the definition of an optimal local common subsequence which consists only of exact pattern matches. One possibility is the search of optimal clusters according to our definition of clusters. A second approach would be the usage of scores instead of distances. This enables the incorporation of normalized scores according to [BHLW05] or other arbitrary scoring schemes [BHLW06].

Appendix A

Comparative Alignments

A.1 RNA_align applied to two Hepatitis C Virus IRES RNAs

Figure A.1: Alignment of two Hepatitis C virus IRES RNAs with RNA_align. The alignment contains 192 exact sequence structure matches, which 168 are in common with the solution of LCS-ERP.

A.2 RNAforester applied to two Hepatitis C Virus IRES RNAs

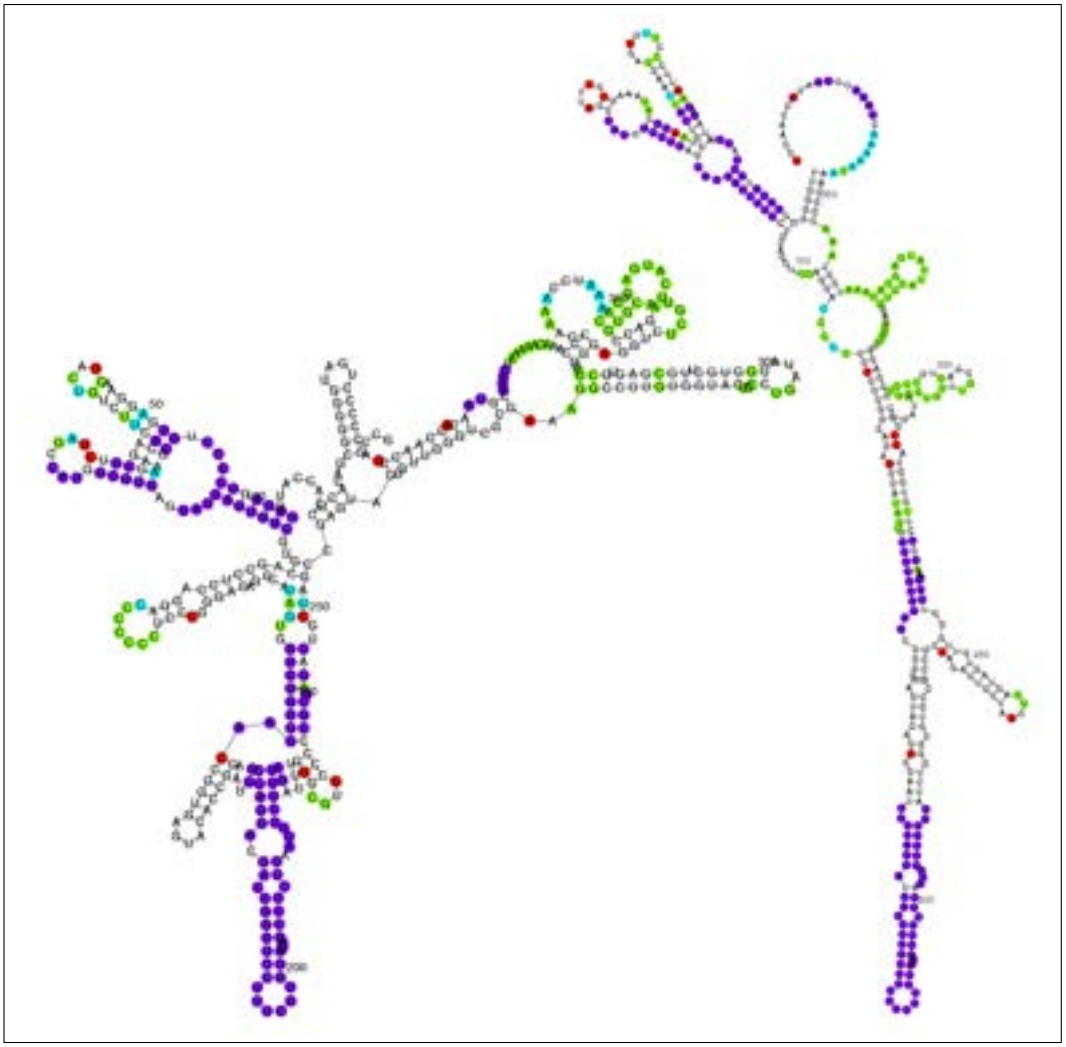

Figure A.2: Comparison between LCS-ERP and RNAforester for two Hepatitis C virus IRES RNAs. The blue colored nucleotides are exact matches in both methods. The light blue nucleotides denote matched nucleotides in both methods but with different positions in the other RNA. The green colored nucleotides occur only in LCS-ERP and the red colored nucleotides occur only in RNAforester. The LCS-ERP solution has 103 exact sequence structure matchings in common with RNAforester. The green colored nucleotides show that the LCS-ERP solution include additional stem and loop regions. (D45172 left RNA, AF165050 right RNA)

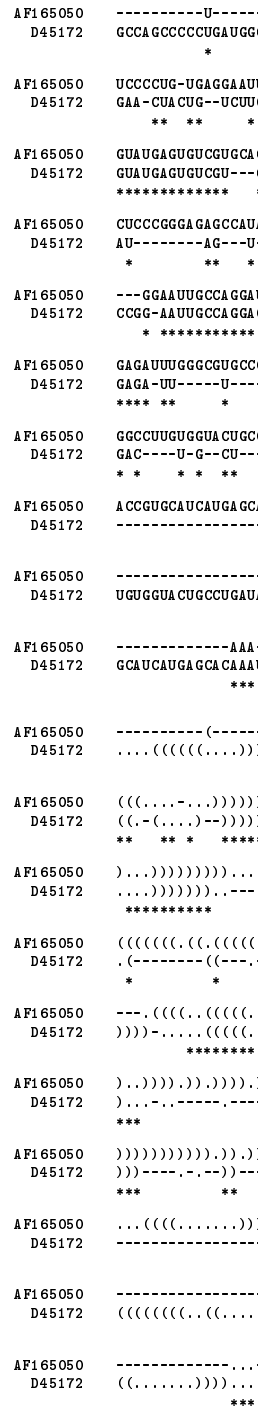

Figure A.3: Alignment of two Hepatitis C virus IRES RNAs with RNAforester. The alignment contains 128 exact sequence structure matches, which 113 are in common with the solution of LCS-ERP.

A.3 RNA_align applied to two 16S rRNAs

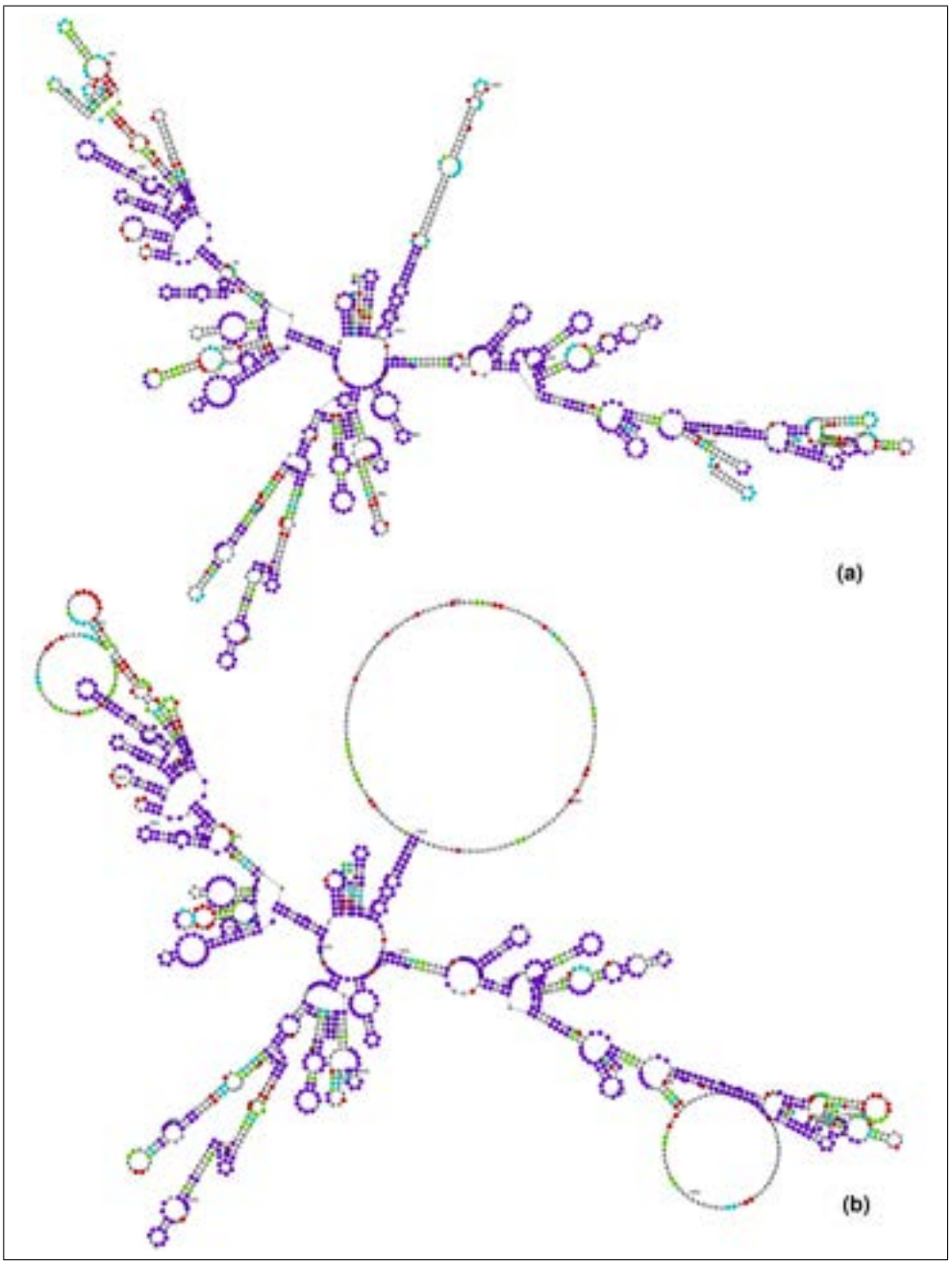

Figure A.4: Comparison between LCS-ERP and RNA_align for two 16S rRNAs. The blue colored nucleotides are exact matches in both methods. The light blue nucleotides denote matched nucleotides in both methods but with different positions in the other RNA. The green colored nucleotides occur only in LCS-ERP and the red colored nucleotides occur only in RNA_align. The LCS-ERP solution has 688 exact sequence structure matchings in common with RNA_align. (a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

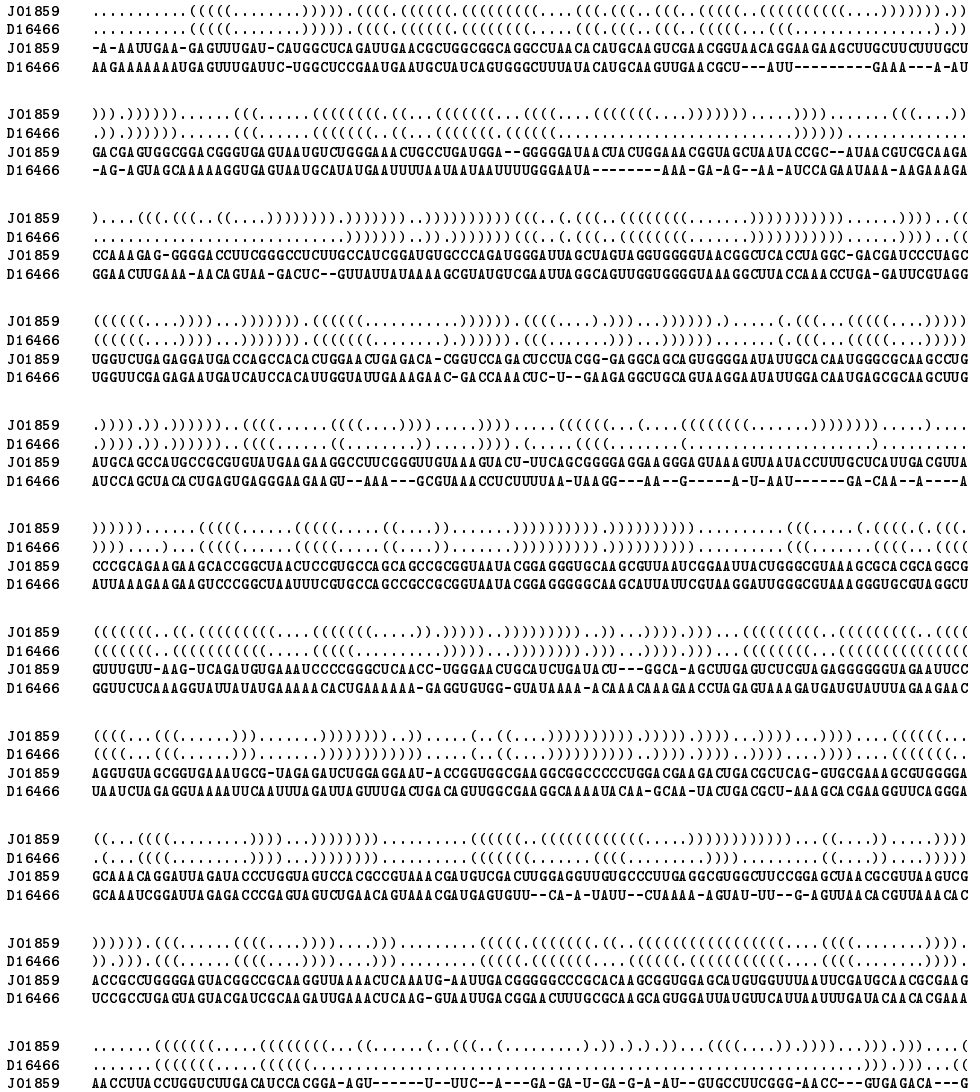

Figure A.5: Alignment of two 16S rRNAs with `RNA_align`. The alignment contains 861 exact sequence structure matches and of these are 688 equal to the solution of LCS-ERP. (continued next page)

Figure A.5: (continued) Alignment of two 16S rRNAs with `RNA_align`. The alignment contains 861 exact sequence structure matches and of these are 688 equal to the solution of LCS-ERP.

A.4 `RNAforester` applied to two 16S rRNAs

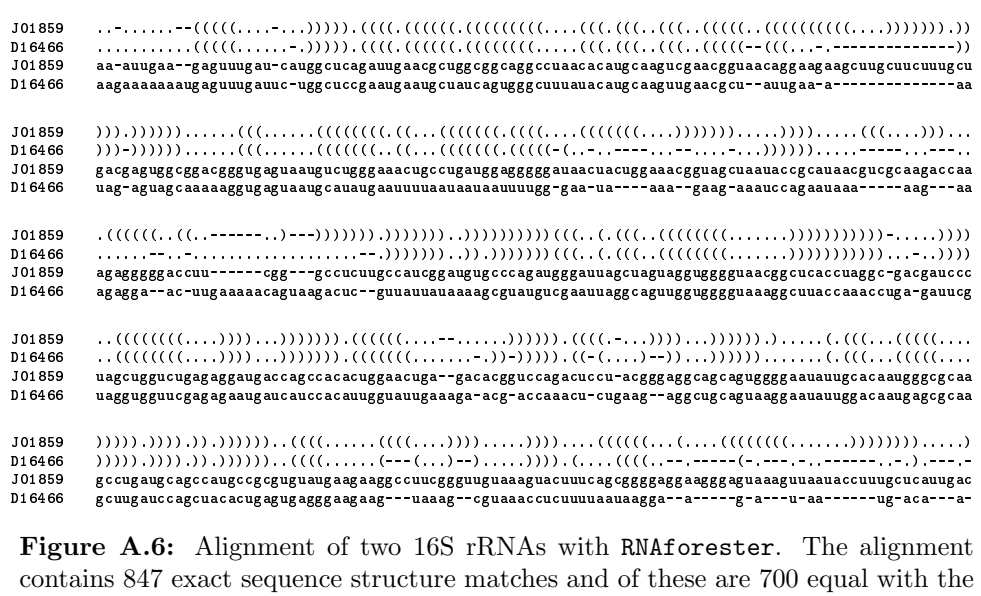

Figure A.6: Alignment of two 16S rRNAs with `RNAforester`. The alignment contains 847 exact sequence structure matches and of these are 700 equal with the solution of LCS-ERP. (continued next page)

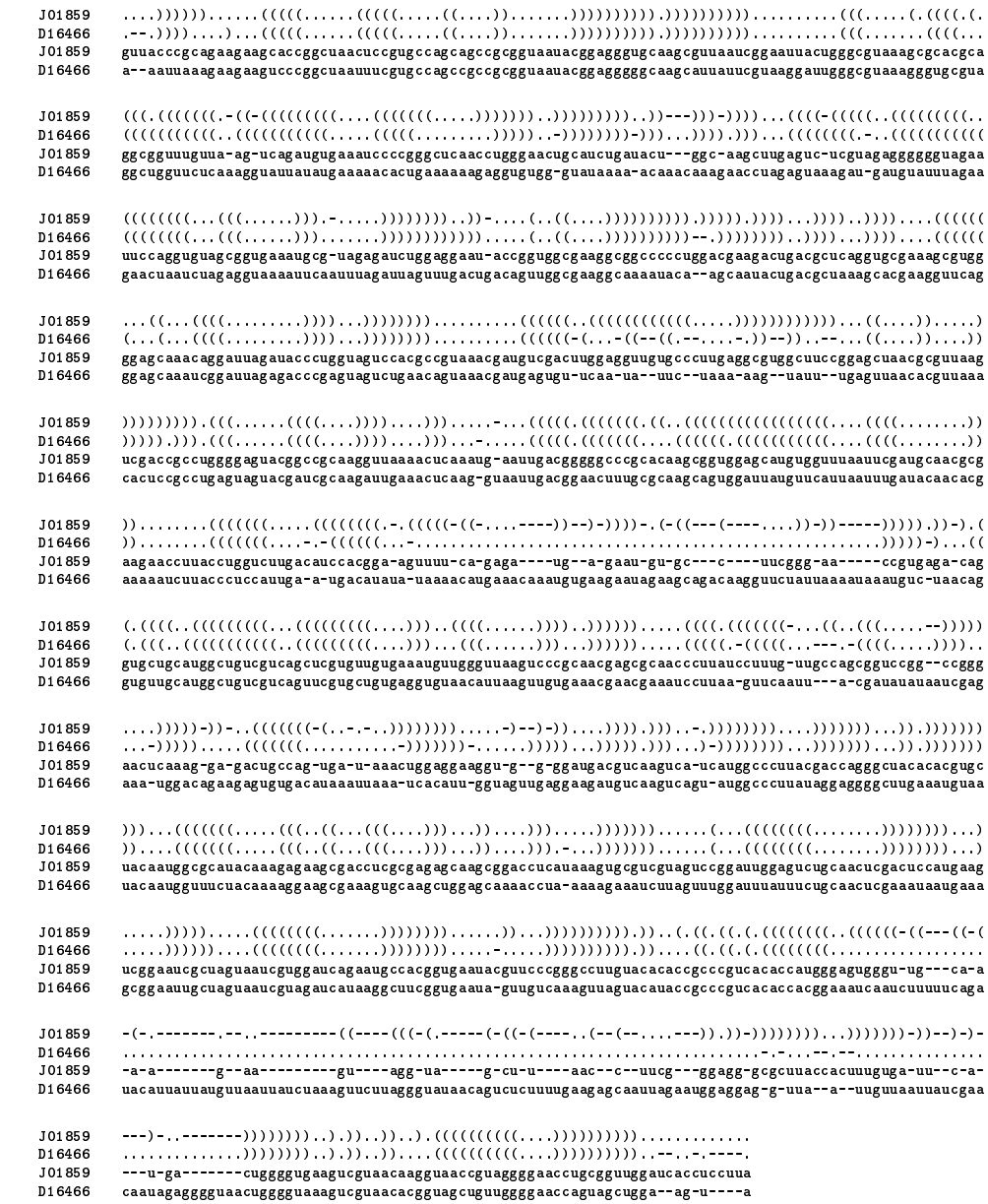

Figure A.6: (continued) Alignment of two 16S rRNAs with `RNAforester`. The alignment contains 847 exact sequence structure matches and of these are 700 equal with the solution of LCS-ERP.

Appendix B

Additional Results

B.1 Clustering applied to two 16S rRNA

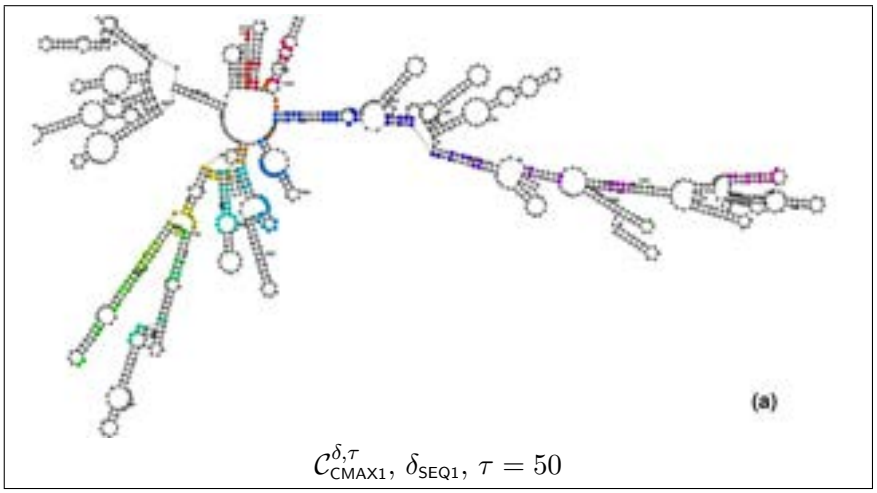

Figure B.1: CLUSTER-MAX-1 strategy applied to two 16S rRNA. The figure shows the largest found clusters for the given parameters. Please see figure B.2 in comparison. (a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

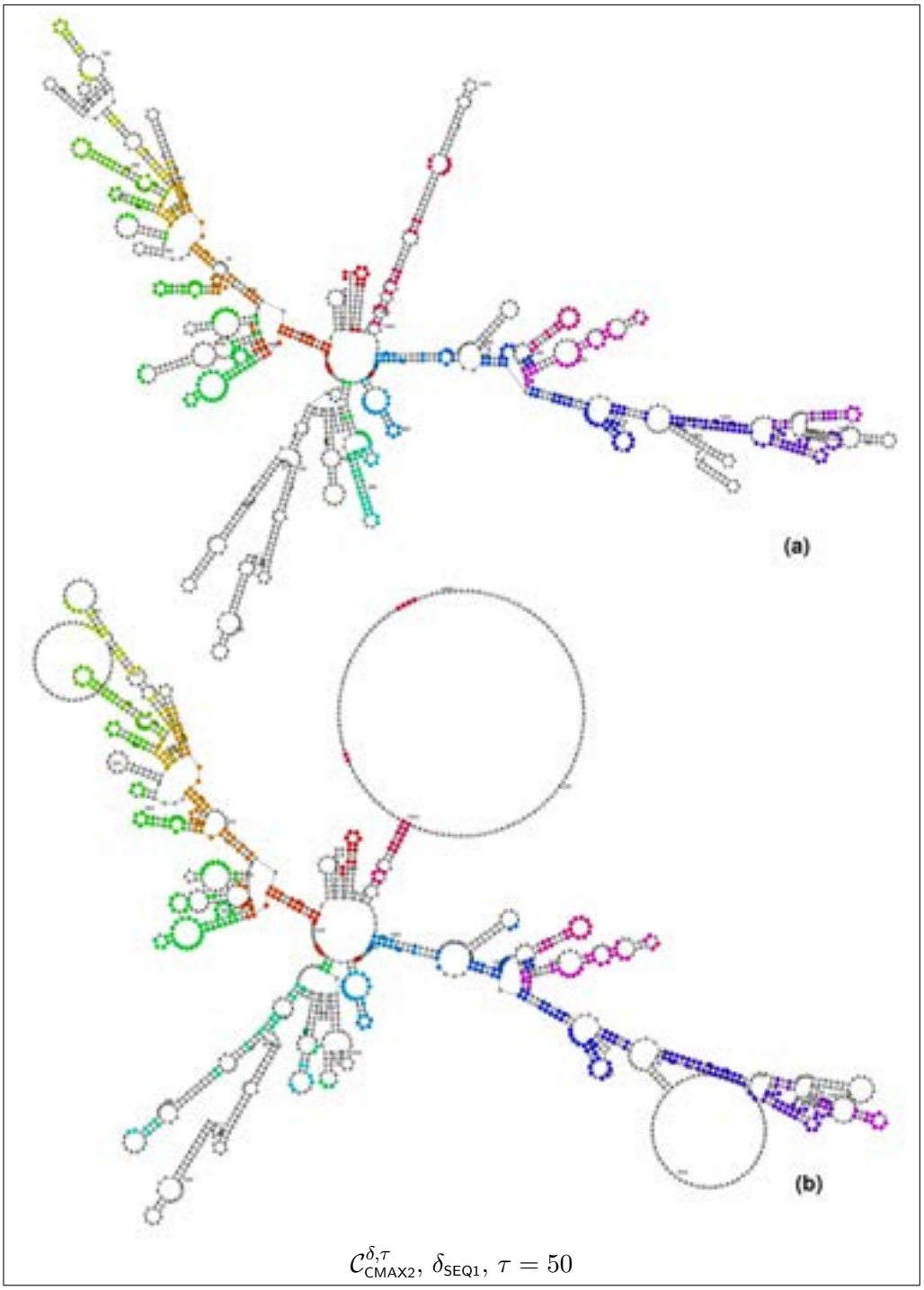

Figure B.2: CLUSTER-MAX-2 strategy applied to two 16S rRNA. The figure shows the largest found clusters for the given parameters. Please see figure B.1 in comparison. (a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

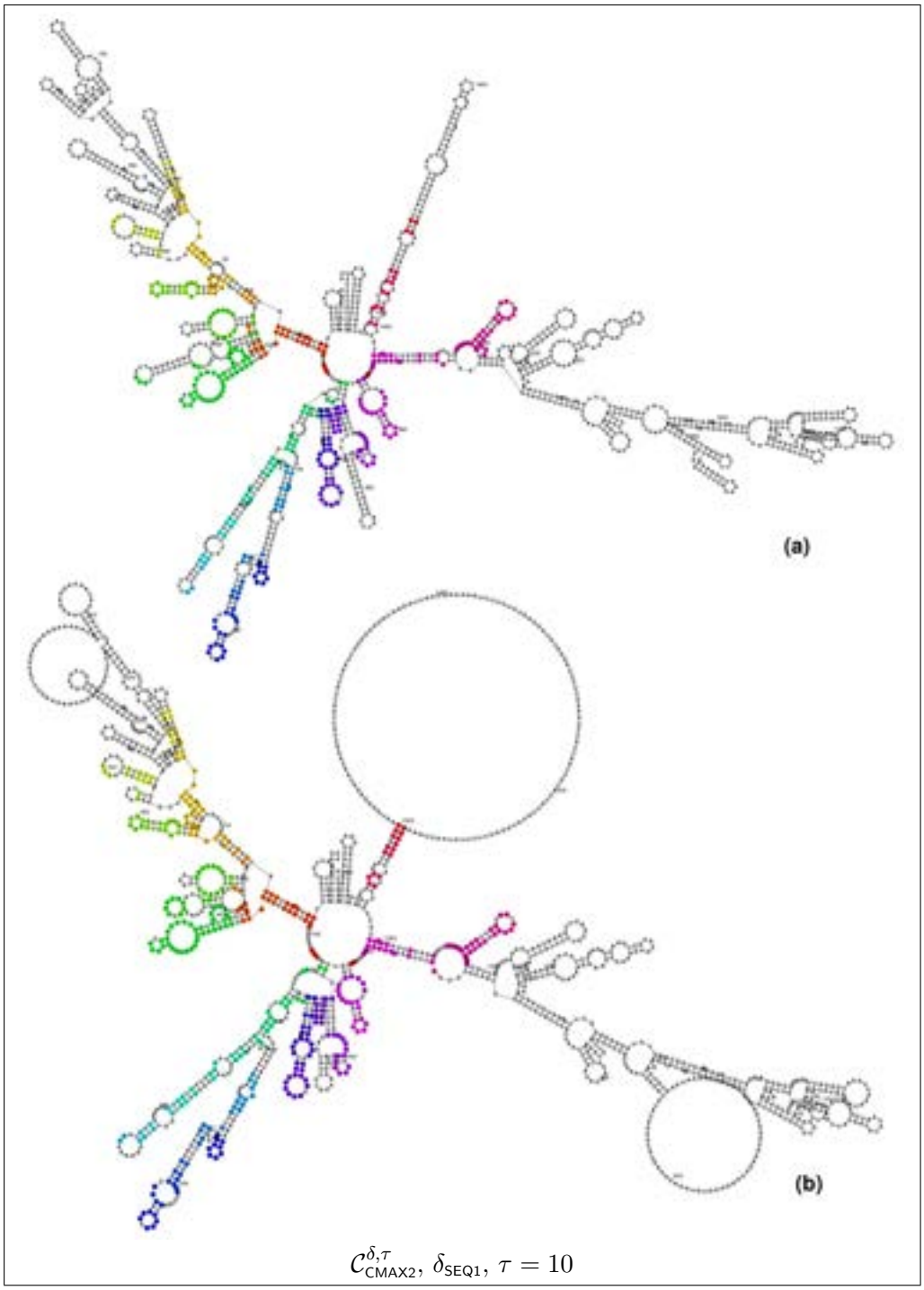

Figure B.3: CLUSTER-MAX-2 strategy applied to two 16S rRNA. The figure shows the largest found clusters for the given parameters. (a) E. coli 16S rRNA (J01859), (b) D. discoideum 16S rRNA (D16466)

B.2 Clustering applied to two Hepatitis C virus IRES RNAs

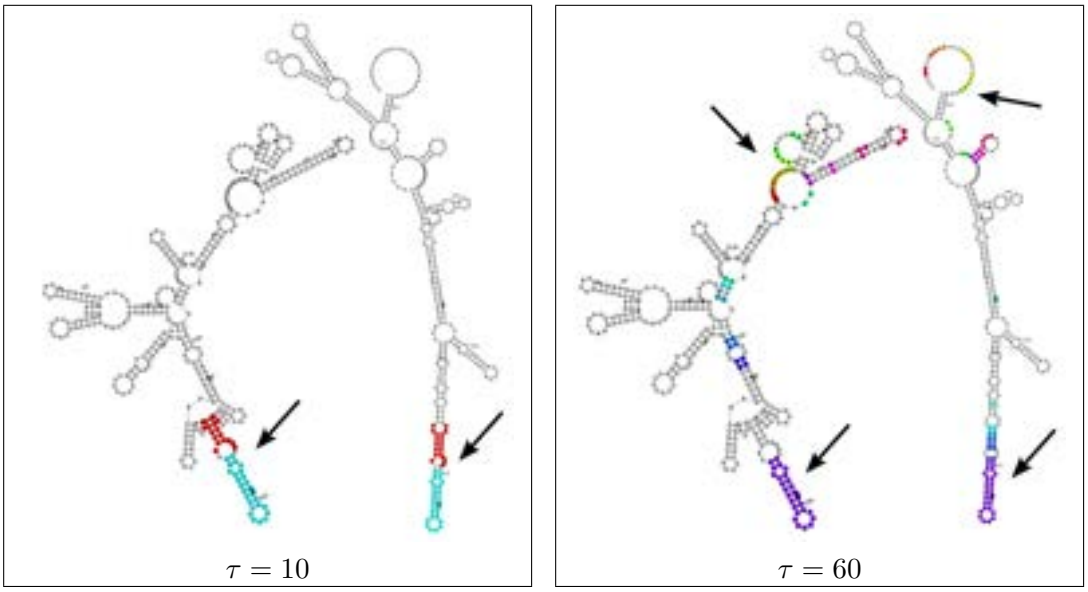

Figure B.4: Additional clusters not shown in figure 6.2. Analysis for Hepatitis C virus IRES RNAs with clustering strategy CLUSTER-MAX-1. The pictures show the largest found cluster for the given distance threshold τ value. Each EPM is shown in a different color. The arrows indicate regions with large matchings.

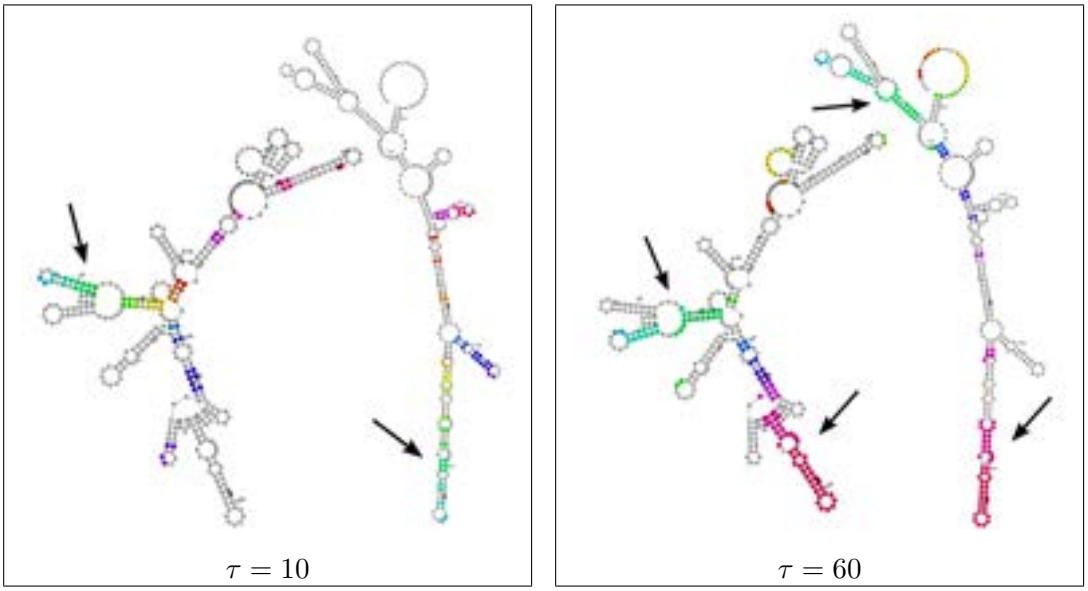

Figure B.5: Additional clusters not shown in figure 6.3. Analysis for two Hepatitis C virus IRES RNAs with clustering strategy CLUSTER-MAX-2. The pictures show the largest found cluster for the given distance threshold τ value. Each EPM is shown in a different color. The arrows indicate regions with large matchings.

Appendix C

MCS algorithm

C.1 Pseudocode for the MCS-algorithm

Algorithm C.1: loop-walking

```
1  Function loop-Walking(i, j)
2  Init: l_{i_1} = 1 ; // global position i
3  Init: l_{j_1} = 1 ; // global position j
4  for k = l_{i_1} to l_{i_size} do
5      for l = l_{j_1} to l_{j_size} do
6          r = 0;
7          if (k, l) not yet considered then
8              while k + r < l_{i_size} ∧ l + r < l_{j_size}
9                      S_1[k + r] = S_2[l + r] ∧
10                     STRUCT_1[k + r] = STRUCT_2[l + r]
11             do r = r + 1;
12             M^{loop}(pos(l), pos(k)) = maxMatching(k,l,r);
```

Algorithm C.2: `max-matching`

```
1  Function maxMatching(i, j, r)
2  Init: size = 0, m = 0;
3  while m ≤ r and S1[i + m] = S2(j + m) do
4      if STRUCT1(i + m) = ss and STRUCT2(j + m) = ss then
5          size = size + 1;
6      else if STRUCT1(i + m) = lp and STRUCT2(j + m) = lp then
7          if S1[i + m + 1] = S2[j + m + 1] then
8              size = size + M^eb(pos(i + m), pos(j + m));
9              m = m + 1;
10         else return size + M^nb(pos(i + m), pos(j + m));
11     else if STRUCT1(i + m) = rp and STRUCT2(j + m) = rp then
12         size = size + M^nb(pos(i + m), pos(j + m));
13     m = m + 1
14 return size;
```

C.2 MCS-algorithm applied to two Hepatitis C virus IRES RNAs

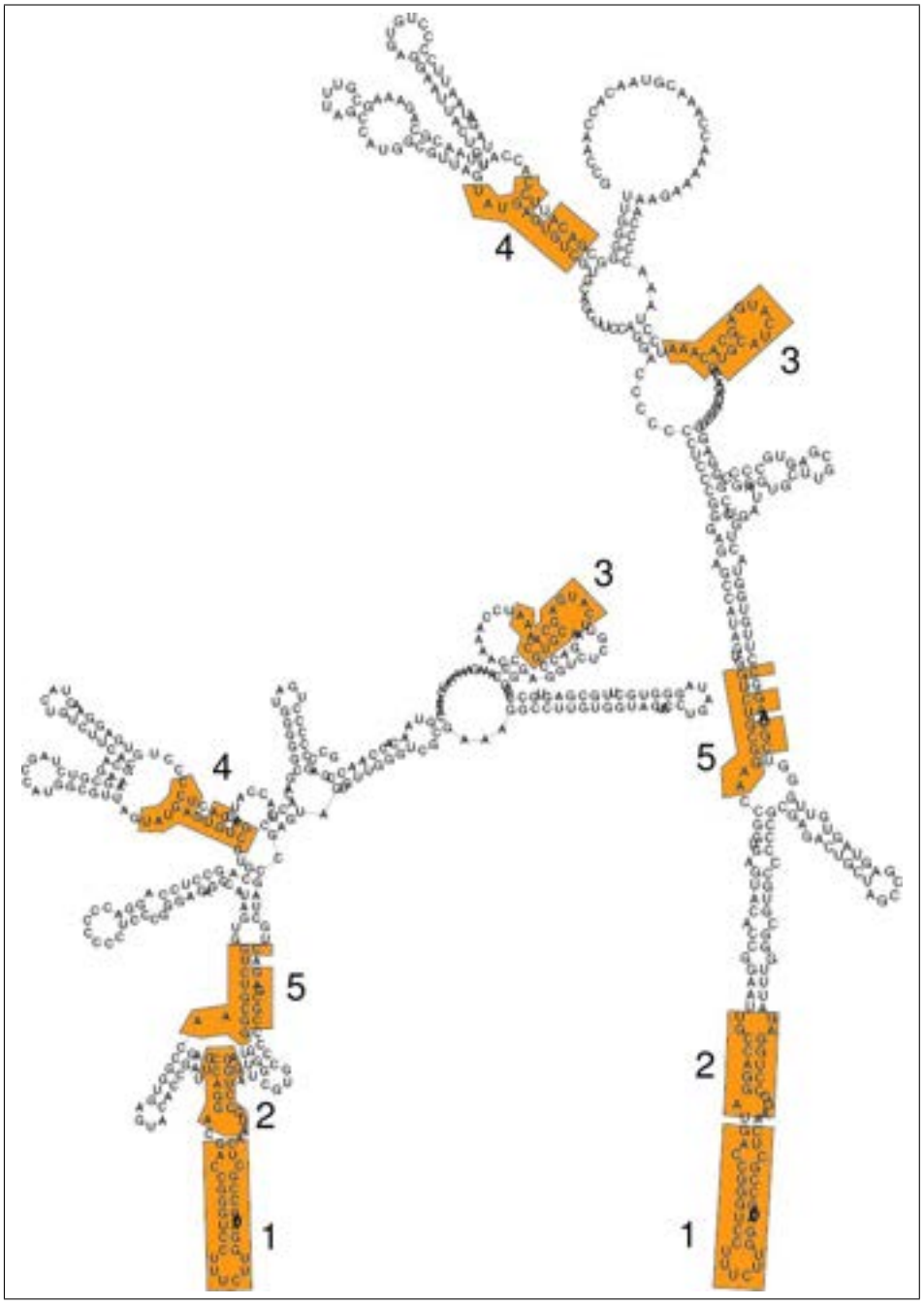

Figure C.1: Two Hepatitis C virus IRES RNAs. The five largest exact pattern matches are highlighted. EPM 1 is the largest found substructure which comprises 30 nucleotides. All marked EPMs hold the Non-Crossing condition. Figure taken from [SB07]. GenBank codes: AF165050 (right RNA), D45172 (left RNA)

Bibliography

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–10, 1990.

[BFRS03] Guillaume Blin, Guillaume Fertin, Irena Rusu, and Christine Sinoquet. RNA sequences and the EDIT(NESTED,NESTED) problem. Technical Report RR-IRIN-03.07, IRIN, Université de Nantes, 2003.

[BHLW05] Rolf Backofen, Danny Hermelin, Gad M. Landau, and Oren Weimann. Normalized similarity of RNA sequences. In *Proc. 12th Symposium on String Processing and Information Retrieval (SPIRE 2005)*, volume 3772 of *Lecture Notes in Computer Science*, pages 360–369. Springer-Verlag, 2005.

[BHLW06] Rolf Backofen, Danny Hermelin, Gad M. Landau, and Oren Weimann. Local alignment of RNA sequences with arbitrary scoring schemes. In *Proc. 17th Symp. Combinatorial Pattern Matching*, volume 4009 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2006.

[BMR95] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In *Proc. 6th Symp. Combinatorial Pattern Matching*, pages 1–16, 1995.

[BS04] Rolf Backofen and Sven Siebert. Fast detection of common sequence structure patterns in RNAs. In *Symposium on String Processing and Information Retrieval 2004 (SPIRE 2004)*, pages 79–92, 2004.

[BW04] Rolf Backofen and Sebastian Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2(4):681–698, 2004.

[BWF+00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–42, 2000.

[CB00] Peter Clote and Rolf Backofen. *Computational Molecular Biology: An Introduction*. Mathematical and Computational Biology. Jon Wiley & Sons, Chichester, August 2000. series editor S. Levin. 290 pages.

[Cor01] Thomas H. Cormen. *Introduction to Algorithms*. MIT Press, 2. edition, 2001.

[Cou02] Jennifer Couzin. Breakthrough of the year. Small RNAs make big splash. *Science*, 298(5602):2296–7, 2002.

[CSS+02] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Muller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs: Correction. *BMC Bioinformatics*, 3(1):15, 2002.

[Edd02] S. R. Eddy. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3(1):18, 2002.

[Eva99] Patricia Anne Evans. *Algorithms and Complexity for Annotated Sequence Analysis*. PhD thesis, University of Alberta, 1999.

[FCDK01] J. E. Fletcher, P. R. Copeland, D. M. Driscoll, and A. Krol. The selenocysteine incorporation machinery: interactions between the SECIS RNA and the SECIS-binding protein SBP2. *RNA*, 7(10):1442–53, 2001.

[GCA06] Raymond F. Gesteland, Thomas R. Cech, and John F. Atkins. *The RNA World*. Cold Spring Harbor Laboratory Press, 3. edition, 2006.

[GGN02] J. Gramm, J. Guo, and R. Niedermeier. Pattern matching for arc-annotated sequences. In *Proc. of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2002)*, 2002.

[GJMM+05] Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean R. Eddy, and Alex Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, 33:D121–D124, 2005.

[GL98] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Protein Science*, 7(2):445–56, 1998.

[Got82] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[GSS01] J. Gorodkin, S. L. Stricklin, and G. D. Stormo. Discovering common stem-loop motifs in unaligned RNA sequences. *Nucleic Acids Research*, 29(10):2135–44, 2001.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

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

[HFS+94] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.

[Hir77] Daniel S. Hirschberg. Complexity of common subsequence problems. In *FCT*, pages 393–398, 1977.

[HTGK03] Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structures. In *Proceedings of Computational Systems Bioinformatics (CSB 2003)*, 2003.

[JLMZ00] T. Jiang, G.-H. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, volume 1848 of *Lecture Notes in Computer Science*, pages 154–165. Springer-Verlag, Berlin, 2000.

[JLMZ02] Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.

[JWZ95] T. Jiang, J. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[KCN+03] Gregory V. Kryukov, Sergi Castellano, Sergey V. Novoselov, Alexey V. Lobanov, Omid Zehtab, Roderic Guigo, and Vadim N. Gladyshev. Characterization of mammalian selenoproteomes. *Science*, 300(5624):1439–43, 2003.

[LCJW02] Guohui Lin, Zhi-Zhong Chen, Tao Jiang, and Jianjun Wen. The longest common subsequence problem for sequences with nested arc annotations. *J. Comput. Syst. Sci.*, 65(3):465–480, 2002.

[LCWI01] Giuseppe Lancia, Robert Carr, Brian Walenz, and Sorin Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proc. of the Fifth Annual International Conferences on Compututational Molecular Biology (RECOMB01)*. ACM Press, 2001.

[LRV98] H.P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. In *Proc. of the Second Annual International Conferences on Compututational Molecular Biology (RECOMB98)*, volume 5, pages 517–30. ACM Press, 1998.

[McC90] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–19, 1990.

[MSZT99] D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288(5):911–40, 1999.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–53, 1970.

[San85] David Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.

[SB05] David W. Staple and Samuel E. Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biology*, 3(6):e213, 2005.

[SB07] Sven Siebert and Rolf Backofen. A dynamic programming approach for finding common patterns in RNAs. *Journal of Computational Biology*, 14(1):34–45, 2007.

[Sie06] Sven Siebert. *Common Sequence Structure Properties and Stable Regions in RNA Secondary Structures.* PhD thesis, Albert-Ludwigs-University Freiburg, Institute of Computer Science, 2006.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[WG06] Kay C. Wiese and Edward Glen. jviz.RNA - an interactive graphical tool for visualizing RNA secondary structure including pseudoknots. In *Proceedings of the 19th International Symposium on Computer Based Medical Systems (IEEE/CBMS-2006)*, pages 659–664, 2006.

[WSPB97] R. Wilting, S. Schorling, B. C. Persson, and A. Böck. Selenoprotein synthesis in archaea: Identification of an mRNA element of Methanococcus jannaschii probably directing selenocysteine insertion. *Journal of Molecular Biology*, 266(4):637–41, 1997.

[WTK+94] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Muller, D. H. Mathews, and M. Zuker. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl. Acad. Sci. USA*, 91(20):9218–22, 1994.

[ZS81] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–48, 1981.

[ZS89] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[ZWM00] Kaizhong Zhang, Lusheng Wang, and Bin Ma. Computing similarity between RNA structures. unpublished, 2000.

List of Figures

List of Tables

List of Algorithms

Selbständigkeitserklärung

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

Jena, den

Unterschrift