Media Engineering and Technology Faculty
German University in Cairo

Accuracy-based identification of local RNA elements

Bachelor Thesis

Author:	Essam A. Moaty A. Hady
Supervisors:	Prof. Dr. Rolf Backofen
	Steffen Heyne
Reviewer:	Prof. Dr. Slim Abdennadher
Submission Date:	18 September, 2010

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Essam A. Moaty A. Hady
18 September, 2010

Abstract

In this thesis we try to tackle the problem of identifying local RNA elements in a genome-wide scale. We employ a fast sparse algorithm to predict maximum expected accuracy structures based on base-pairing/unpairing probabilities. Moreover, we introduce a new locality definition and present an accuracy function reflecting this locality.

Base-pairing and base-unpairing probabilities can be efficiently computed using `RNAplfold` [32] included in the Vienna package [17]. Based on these probabilities, we identify structured regions that have high probabilities of containing significant local RNA motifs.

After that, we introduce our new program `RNAMotid` together with other included features that enables it to scan genome-wide sequences for structured regions. Moreover, we discuss how several modules were integrated together in our program to allow flexibility and optionality of the analysis.

Finally, we evaluate the performance of `RNAMotid` in identifying local RNA motifs embedded in randomly shuffled context. Before that, we apply an overall parameter training followed by a family-based parameter training. Then we discuss the factors that affect the performance of `RNAMotid`.

Contents

Chapter 1

Introduction

1.1 Motivation

Ribonucleic acids (RNAs) are molecules of high biological importance in the living cell. Such molecules could be divided into functionally different categories, such as mRNAs (messenger RNAs), tRNAs (transfer RNAs) and microRNAs. A brief description of mRNA molecules' life cycle can be summarized as a cycle that begins with transcription from DNA molecules. This cycle ends with a final degradation of the mRNA molecule as shown in figure 1.1. mRNA molecules experience through such cycle a translation stage to complete the protein biosynthesis process. The final product depends on the gene being expressed.

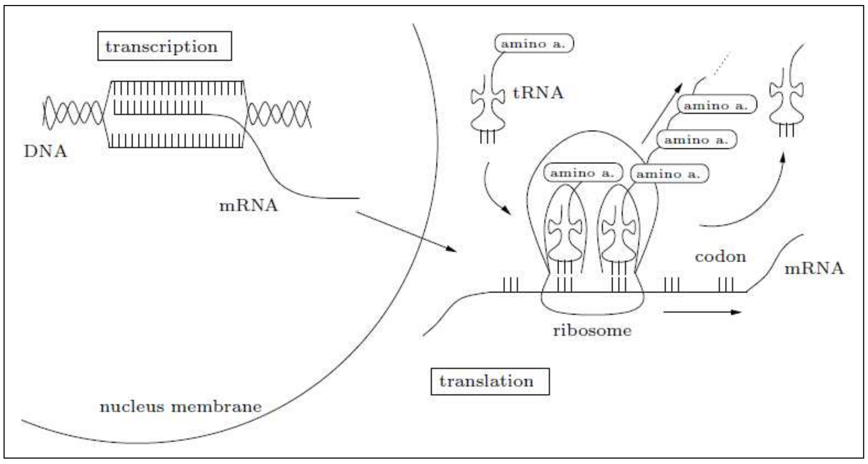


Figure 1.1: A schematic description of the transcription and translation processes inside a eukaryotic cell with a nucleus. Figure taken from [3]

In order to control the produced amount of proteins, mRNA molecules experience a gene expression regulation phase . The correct time point and expression level are crucial for correct functionality of the cell [11, 28]. The discovery of elements responsible for such regulation draw attention to the important role that some classes of RNA molecules can play in controlling the living cell [19]. Changes in the expression of a particular gene is often the effect of cis- or trans-regulatory elements. For example, AREs (AU-rich elements) - are regions in mRNAs that are functionally important for their degradation [20, 1]. A well known example of trans-regulatory elements are microRNAs.

Conserved RNA motifs play crucial role in the cell. Accordingly, it is an important task to provide computational tools [13, 23, 35] for the analysis of RNA sequences. Such tools include the ability to identify motifs and RNA functional elements. However, the application of these tools is still not possible for a genome-wide scale analysis. Huge amounts of sequence data become available every day. However, finding out structurally conserved and significant RNA elements in such large scale remains with limited progress.

The reasons behind these limits may become more clear by considering two questions. The first one is, how significant elements in a given RNA sequence can be recognized. And the second one is, if we have a group of related RNA sequences, how can we determine regions where common structurally conserved elements or at least similar ones reside with an allowed degree of mis-match.

The first question proposes prediction of the secondary structure, heading to an optimization goal of maximizing a given property that is exhibited by such motifs and other functional RNA elements. One possibility of such folding criteria is to look for the structure with the highest energy stability, i.e., folding with the optimization goal of finding out the MFE (minimum free energy) structure. This was previously suggested to be a property of functional RNA elements [24]. However, energy stability was found to be not of statistical significance for a reliable detection of functional RNA elements [12]. Moreover, this approach is considering only the MFE structure. In addition, the probability of the MFE structure will be very small for large sequences [4]. Therefore, the usage of the whole structural ensemble instead of fixed minimum free energy structure is often a better RNA secondary structure representation.

The second question proposes applying multiple structure alignment on two or more related input sequences. This approach exploits the fact that RNA secondary structure is more conserved than the sequence in genomes of evolutionarily related organisms [8]. The main limitation of this approach is the enormous computational cost of aligning the input sequences in a genomic-wide scale. For example, it was estimated to spend months processing human and mouse genomes for common regions [2] using `Dynalign` [10, 9] for example. Some faster approaches exist that depends on supplying a predicted minimum free energy structure to the multiple alignment module. However, such approaches depend on the quality of the structure prediction algorithms [8] and accordingly may not be a good choice for analyzing sequences in a genome-wide scale.

In this thesis, we try to tackle the problem of identifying local RNA elements in a genome-wide scale. We employ a fast sparse algorithm to predict maximum expected accuracy structures based on base-pairing/unpairing probabilities. Base-pairing/unpairing probabilities can be efficiently computed using `RNAplfold` [32] included in the Vienna RNA package [17]. Based on these probabilities we identify structured regions that have high probabilities of containing significant local RNA motifs.

1.2 Related Work

In order to identify functional RNA elements in long sequences, many tools and strategies are available to automate this process. These tools vary in their computational cost and biological accuracy according to the approach they adapt and the assumptions they make. Eventually, the user is left to make the decision between fast executing and high accuracy tools.

In the next subsections, we briefly categorize previous solutions to our problem according to the strategy they were based on and discuss advantages and drawbacks for each strategy.

1.2.1 Comparative-based solutions

Comparative approaches assume that the sequence of an RNA element and its functional structure are already known. Others make use of functionally related elements from the same RNA class in a set of genomes. This knowledge is then employed in training a dedicated computational model - a probabilistic one in most of the cases - to capture such information and use this trained model to process long RNA sequences. Comparative approaches consider the conservation of functional structures in evolutionarily related genomes as key feature for functional analysis [8].

One way of applying such strategy is training a covariance model[1] [36] using `Infernal` [14] with a multiple sequence alignment that ensures that functional structures are correctly aligned. Multiple alignments of ncRNAs for many RNA classes can be obtained from the `Rfam` database [33]. This approach assumes that a multiple sequence alignment exists and that it has been already checked for correctness of the alignment of functional structures.

The computational cost of applying this approach on an RNA sequence of length L using a covariance model trained on an alignment of length n is in $O(Ln^{1.3})$ time and in $O(n^2 \log n)$ space.

Even in the absence of an already computed sequence alignment, the application of this approach is still possible. In order to adapt to the new situation, a multiple sequence alignment is computed and then proceeding as previously mentioned with `Infernal`.

Other tools do not require a multiple sequence alignment as an input. `CMfinder` [38] accepts unaligned sequences and employs an algorithm that trains a covariance model. However, following a probabilistic model approach involves high dependency on other evolutionarily related sequences used during the training of the model. Moreover, covariance models restricts the extension of the tools from any further attempts for the inclusion of a pseudo-knotted modelling module because, they are SCFG (stochastic context free

[1]A probabilistic model that can flexibly describe the secondary structure of an RNA element together with the primary structure.

grammar) based. It can only be done by enduring the high cost of translational grammars.

The usage of previously mentioned tools assumes that other evolutionarily and functionally related sequences are available to train the probabilistic model to capture sequential and/or structural characteristics. However, this assumption may not hold in some cases. Other tools that operate comparatively can do without the need for other evolutionarily related aligned sequences. `RSEARCH` [31] can be considered as a good representative for this kind of tools. However, `RSEARCH` is also SCFG based and thus comprehends the same previously mentioned drawbacks of SCFGs.

When using `RSEARCH` for scanning a long sequence of length n with a shorter sequence of length L to find structural and sequential similarities, the running time complexity is in $O(nL^3)$ and the space complexity is in $O(L^3)$.

The high computational cost of `RSEARCH` makes it inapplicable for a genome-wide RNA sequences analysis. Moreover, only one RNA sequence is used as text to look for similarities. This makes the sensitivity of the results not as good as other comparative based approaches.

Up to now a previous knowledge of both the sequence and the structure of the motif is assumed. However, it is not always the case. Tools trying to deal with the scenario when the structure is not known proceed by supposing a structure for a pattern of RNA element sequence - usually the thermodynamic stable structure is assumed, i.e., the MFE (Minimum Free Energy) structure. Then a sequence-structure alignment is applied to a target RNA sequence. Tools of such behaviour also assumes a conservation of functional structures in related genomes by evolution.

Examples of these tools include `FOLDALIGN` [22, 21] and `Dynalign` [10, 9]. They employ simultaneous alignment and folding - Sankoff-style alignment - to improve the quality of structure prediction. However, they are computationally expensive. Both of them minimize the combined free energy of the alignment. They assume Zuker's energy model as proposed in [24]. An assumption that was proven to be of non statistical significance in [12]. Moreover, the very high computational cost makes them inapplicable for genome-wide analysis of functional RNA elements. Also `RNAforester` [27, 26, 25] can be considered as an example. However, it proceeds by first predicting MFE structures then aligning fixed secondary structures.

1.2.2 Prediction-based solutions

Doing without comparative-based solutions when searching for functional RNA elements seems to be a good solution. Especially, when dealing with genome-wide scale analysis. This is due to the enormous computational cost required by comparative approaches. However, choosing a prediction-based strategy for functional RNA elements detection is still considered a challenging alternative that depends on the quality of the folding algorithm.

Programs predicting functional RNA elements are usually MFE based. For example, `RNALfold` [18] from the Vienna RNA package [17] which calculates locally stable MFE secondary structures of RNAs. However, considering only a fixed MFE structure may not be the best representation.

1.2.3 Family-specialized solutions

The categorization of RNAs into different classes or families paved the way to employ such division in devising other approaches that are RNA class specific. Solutions exploiting such categorization include `miRseeker` that's specialized in microRNAs detection and `tRNAscan` for tRNAs (transfer RNAs) detection. `RNAmicro` also is a tool that exploits this categorization and is designed for the purpose of dealing with genome-wide scale data.

1.3 Contribution

Typically, in large-scale analysis of genomic sequences for functional motifs, neither the functional structure of the RNA element is known nor its boundaries [8]. Moreover, comparative approaches have a high computational cost that makes them inapplicable for extended genome-wide scale analysis as discussed before in section 1.2.1. Accordingly, approaches based on multiple sequence-structure alignment seem not to be the optimal choice for large-scale analysis.

Furthermore, the assumption of a global folding model applies only to trans-acting RNAs and RNA sequences up to ∼1000 nucleotides. For long RNA sequences it was shown that base-pairs with a large span are disfavored kinetically [5]. Hence, a local folding model is more appropriate for the analysis of long RNA sequences like mRNAs, untranslated regions (UTRs) and long non-coding RNAs (lncRNAs).

Base-pairing probabilities for base-pairs restricted to a span of L together with local unpairing probabilities can be efficiently calculated by `RNAplfold` [32]. `RNAplfold` considers the ensembles of all secondary structures in thermodynamic equilibrium. It is implemented in an efficient way that requires a running time in $O(nL^2)$ and a space in $O(n + L^2)$ to compute base-pairing and base-unpairing probabilities. It uses a sliding window fashion of fixed length L while processing the RNA sequence of length n.

The use of local base-pairing/unpairing probabilities within long RNA transcripts has the advantage that the complete structural ensemble of the RNA is considered instead of the fixed minimum free energy structure. In this thesis we investigate the potentials of employing `RNAplfold` [32] in the detection of local structured cis-acting and trans-acting RNA elements.

The difference to existing approaches is a new definition for significant local RNA elements. Local accuracies are weighted by a normalization function, which has an inverse proportionality to the length of the motif.

Predictions from scanning algorithms are biased towards large motifs in case of large folding windows [8]. However, using an accuracy function with this nature avoids such bias and favours only motifs according to the final weighted accuracy value.

Recently, a fast sparse folding algorithm was proposed in [37]. We apply such algorithm to compute the maximum expected accuracy structure based on precomputed base-pairing and base-unpairing probabilities. This allows for an efficient identification of branched RNA elements instead of only hairpin structures.

In order to be able to analyze long RNA sequences in genomic scale, a scanning fashion is used for calculating base-pairing/unpairing probabilities and for applying the sparse folding algorithm. This window scanning approach has the computational advantage of keeping memory requirements manageable. Also, biologically it maintains the importance of being local in structure prediction while having no dependency on the chosen folding window. This approach takes into consideration that the final output is not affected by such windowing by considering transitional overlapping sections between successive windows in computing probabilities and in folding as well.

1.4 Overview

Chapter 2 introduces the necessary definitions, notations and conventions used throughout the thesis. In chapter 3, a full description of the implemented algorithms is presented. Finally, time and space analysis of the program are given. Chapter 4 discusses the implementation details and the features included in the program. In chapter 5, we present the evaluation of the program and discuss the results. In chapter 6 we conclude this thesis with an outlook on possible future work.

Chapter 2

Preliminaries

In this chapter we introduce fundamental biological definitions and other concepts and conventions related to the algorithmic aspects of our contribution. We are going to follow these conventions throughout the rest of the thesis.

2.1 Molecular Biology: DNA, RNA and Motifs

DNAs (Deoxyribonucleic acid) and RNAs (Ribonucleic acid) are two of the most important types of molecules in the living cell. The fact that DNA is responsible for storing the heredity information explains the reason behind its high importance. Furthermore, DNA is the place from which the complex process of protein biosynthesis start.

One of the characteristics that constitute differences between DNA and RNA molecules is their structure. DNA molecules are constructed from a double chain of *nucleotides* which we will call *bases.* Complementary bases from opposite chains pair together and form what we call *a base pair.* Typically, a DNA molecule contains four bases which are *adenine*, *cytosine* , *guanine* and *thymine*. These bases are designated as *A, C, G* and *T* respectively. The pairing between complementary bases from different chains is said to follow a *Watson-Crick pair*[1] scheme.

In contrast to the DNA molecular structure and bases composition, a typical RNA molecule is a single stranded one. However, bases along that strand can pair together allowing the RNA molecule to fold forming many structural levels. Theses levels differ in their degree of freedom that we are going to discuss later. Moreover, an RNA molecule's bases composition is not exactly the same as in DNA molecules. In RNA, a thymine type nucleotide - designated as T - is always replaced with a *uracil* type nucleotide designated as U.

[1]James Watson and Francis Crick, they were the first to come up with the double helix model for the molecular structure of DNA molecules.

2.1.1 RNA Structures

An RNA molecule is able to fold into a structure by base-pairs. In order to describe the formed structure in detail, we introduce different levels of abstraction. Figure 2.1 depicts those different structural levels. Next, we briefly explore each of those structures and introduce some definitions.

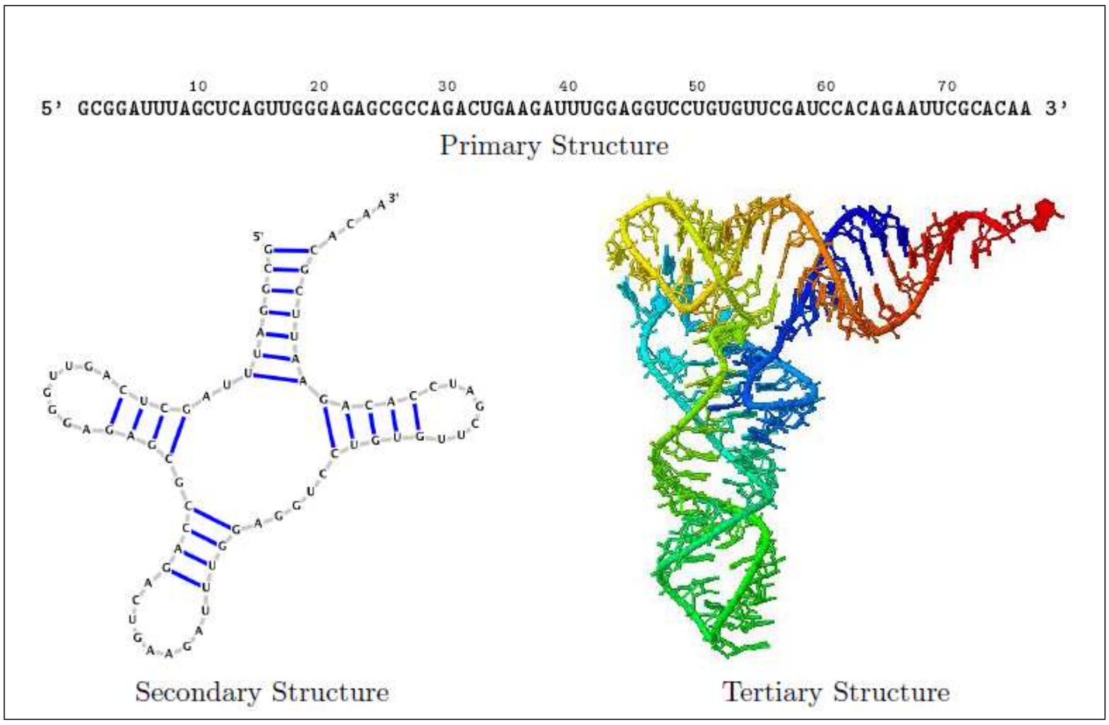

10 20 30 40 50 60 70
5' GCGGAUUUAGCUCAGUUGGGAGAGCGCCAGACUGAAGAUUUGGAGGUCCUGUGUUCGAUCCACAGAAUUCGCACAA 3'
Primary Structure
5'
3'
Secondary Structure
Tertiary Structure

Figure 2.1: Primary, secondary and tertiary structures of a sample RNA molecule. Figure taken from [34]

Primary Structure

This structure describes the order of nucleotides in an RNA molecule. It models the molecule on the sequence level as a string of symbols. These symbols correspond to the bases forming the RNA molecule's strand. It is the simplest level of abstraction for describing an RNA molecule for further required analysis.

Throughout this thesis we refer to the alphabet Σ as an alphabet over the set of symbols $\{A, C, G, U\}$.

Definition 2.1.1 (Primary Structure)**.** *For an RNA sequence, a primary structure $P = p_0 \ldots p_{n-1}$ is the ordered finite string of concatenation of its symbols over the alphabet Σ. The length of P is the count of its symbols and is designated as $|P|$. By p_i we denote the i^{th} symbol and by $P_{i,j}$ we denote the subsequence starting at and including p_i which ends at p_j including it.*

Alternatively, a primary structure can be called a primary sequence due to its sequential nature. Furthermore, we mean by subsequence here the continuous substring.

The determination of the nucleotide sequence of a given RNA molecule can be done by laboratory experimental means and is not considered as a challenging problem in the world of bioinformatics.

Secondary Structure

The secondary structure exploits an additional degree of freedom. In addition to the 1D-modeling of the RNA molecules' composition, the secondary structure describes how the molecule is folded in a 2D level. This is done by describing the pairings between molecule bases. Typically, the secondary structure is more conserved in evolution than the primary structure between related organisms. This is one reason for the increasing interest to find those conserved structures.

The complexity of a secondary structure varies according to the complexity of the base-pairs. A *plain* secondary structure has no base-pairs and is equivalent to a primary structure. However, a *branched* secondary structure introduces base-pairs with constraints on the presence of such base-pairs. These constraints guarantee that no nesting or even crossing base-pairs are allowed. A *nested* secondary structure keeps the constraint of being non-crossing but allows a nesting in the base-pairings. Finally, a crossing structure allows that two base-pairs may cross each other. An RNA molecule whose secondary structure contains a crossing base-pair is said to be having a pseudoknot. A summary of the different complexity levels of RNA secondary structures is depicted in figure 2.2. Throughout this work we will not deal with crossing secondary structures. We proceed by defining an RNA secondary structure.

Definition 2.1.2 (Secondary Structure)**.** *Let $P = p_0 \dots p_{n-1}$ be a primary structure. A secondary structure of an RNA molecule is the set of pairs $S = \{(i,j) \mid 0 \leq i < j < n\}$ where p_i is paired with p_j. Moreover, for a given index k s.t. $0 \leq k < n$, p_k occurs at most once in both components of the pairs in S, i.e., $\forall ((i,j),(i',j)' \in S) : i = i' \Leftrightarrow j = j'$ and $i \neq j$. By $S_{i,j}$ we denote the substructure for the subsequence $P_{i,j}$. The number of base-pairs in a given substructure is denoted by $|S_{i,j}|$ or alternatively $BP(S_{i,j})$.*

Definition 2.1.3 (Minimum pairing length)**.** *Let S be a secondary structure for an RNA primary sequence P. By μ we denote the minimum allowed length that a base-pair in S can span. Formally, $j - i \geq \mu \ \forall \ (p_i, p_j) \in S$.*

Definition 2.1.4 (Maximum pairing length)**.** *Let S be a secondary structure for an RNA primary sequence P. By δ we denote the maximum allowed length that a base-pair in S can span. Formally, $j - i \leq \delta \ \forall \ (p_i, p_j) \in S$.*

This definition ensures that the maximum degree of pairing is one for any base, i.e., each base can pair with at most only one base.

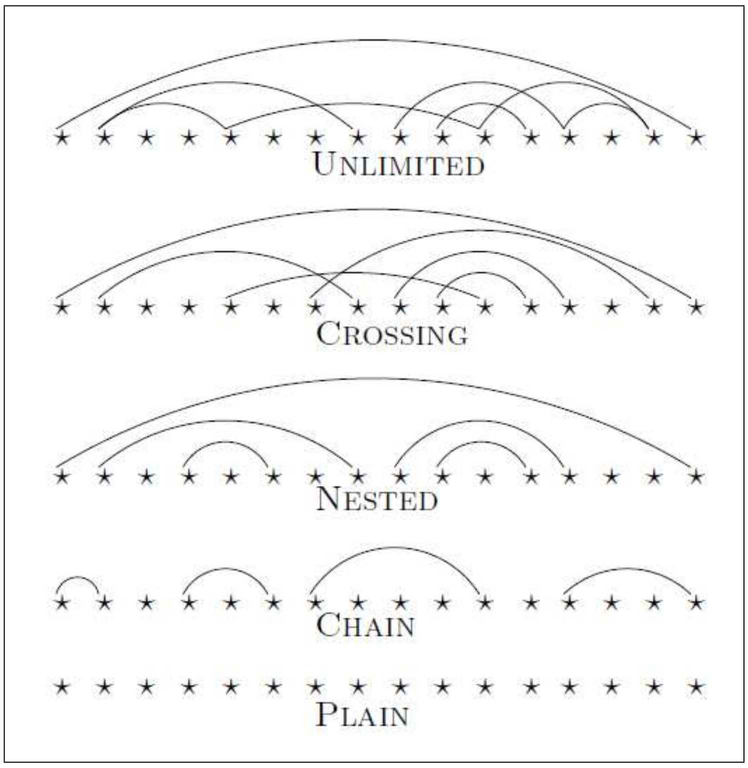


Figure 2.2: Different categories of RNA secondary structures. Figure taken from [3]

Tertiary Structure

By RNA tertiary structure we mean the 3D folding of an RNA molecule. Typically, the determination or the prediction of such structure is an expensive and complex task. We will not go further with this type as it is irrelevant to our work in this thesis.

2.1.2 Functional RNA elements

The DNA encodes all the necessary information for the processes in the cell. The cell makes use of this information by the process of *transcription*. This process copies the information from a specific DNA region called *gene* to an RNA molecule. Typically, the copying process is catalysed by some enzymes. For a long period, the whole picture of an RNA molecule was limited to a molecule that carries the required blueprint for protein biosynthesis from the gene to the protein factories, called *ribosomes*. The encoding for a protein is represented by mRNAs (messenger RNAs).

Protein biosynthesis contains one more process that takes place outside the nucleus after the transcription process. This process is called *translation*. In this process the construction of proteins is done by connecting smaller building blocks together in an ordered fashion. These building blocks are called *amino acids*. Outside the nucleus, ribosomes scan the mRNA and simultaneously connect the corresponding amino acids to form the required protein. Required amino acids are transferred by tRNAs (transfer RNAs). A schematic view of protein biosynthesis process is depicted in 1.1.

A *non-coding region* of a gene is responsible for the production of *non-coding RNAs*. Those non-coding RNAs are functionally important for the living cell, such as tRNAs. The next subsections categorize these functional elements into two groups and explain briefly each of them.

cis-regulatory motifs

These are regions of the mRNA molecule itself. They are responsible for regulating the process of gene expression. This type of control is required to end up with a suitable level and an appropriate time for gene expression. Only a part of the mRNA molecule carries the required information for producing the required protein. The rest of the molecule plays an important regulatory role throughout the whole life cycle of mRNAs. The parts of the mRNA molecule that are responsible for such regulation are called *cis-regulatory motifs*. Some of these motifs are responsible even for the degradation of mRNA molecules, such as AREs (AU-rich elements).

trans-regulatory RNAs

Other kind of regulatory RNAs are called *trans-regulatory RNAs*. They differ from cis-regulatory motifs in the place where they reside and from which they can carry out their regulatory duties. An example of such motifs is microRNAs. They are typically 20-25 nucleotides long single-stranded molecules that function when binding to mRNAs for quick degradation or for protein translation inhibition [6, 15].

2.2 RNA Secondary Structure Prediction

Different approaches exist to predict the secondary structure of an RNA molecule given its primary sequence. Especially dynamic programming algorithms were designed for the problem. However, the prediction of the most realistic one is always a challenge. In the next subsections, we explore such attempts and analyze their worst case running times and space complexities. Moreover, we explore the capabilities of each algorithm when utilized for an extended analysis in a genome-wide scale.

2.2.1 Maximum number of base-pairs

The first approach is a dynamic programming algorithm, with an optimization goal of maximizing the number of base-pairs in the folded secondary structure. This algorithm is named *Nussinov's algorithm* [30] after the name of its designer. The goal of Nussinov algorithm is maximizing the number of base-pairs in the computed secondary structure without taking into account the nature of the resulting substructures.

Definition 2.2.1 (Maximum base-pairs structure)**.** *Recall that we referred to the number of base-pairs in a given substructure by $BP(S_{i,j})$. By $\Psi_{i,j}$ we denote the secondary substructure with the maximum number of base-pairs for a given RNA primary subsequence $P_{i,j}$.*

Definition 2.2.2 (N-matrix)**.** *For an RNA primary sequence $P_{o,n-1}$, a Nusinov's matrix is an $(n \times n)$-matrix with entries $N[i,j]$ such that $N[i,j] = BP(\Psi_{i,j})$ $\forall$ $0 \leq i,j < n$.*

The algorithm of Nussinov tries to find $BP(\Psi_{0,n-1})$ for a given primary sequence P, i.e., tries to maximize the value $BP(S_{i,j})$ for all $0 \leq i < j < n$ by filling in the N-matrix corresponding to the given RNA primary sequence P.

A naive implementation of Nussinov algorithm together with its underlying recurrence are presented in algorithm 1. This implementation fills the upper right triangular part of the N-matrix. After initializing the main diagonal, the algorithm proceeds by filling other shorter diagonals starting from the upper left end of each diagonal. By filling the N-matrix entry $N[0, n-1]$, the algorithm has computed the maximum number of base-pairs that could be found in a global secondary structure folding for the input primary sequence.

Algorithm 1 The Algorithm of Nussinov

```
Input: a primary sequence $P = p_0 \dots p_{n-1}$.
1. Initialization
  for $i = 1$ to $n-1$ do
    $N[i, i-1] = 0$;
  end for
  for $i = 0$ to $n-1$ do
    $N[i, i] = 0$;
  end for
2. Recurrence
  for $l = 1$ to $n-1$ do
    for $i = 0$ to $n-l-1$ do
      $j = i + l$;
      $N[i,j] = max \begin{cases} N[i-1, j-1] + 1, \; p_i \text{ and } p_j \text{ are complementary} & (i) \\ N[i, j-1] & (ii) \\ N[i+1, j] & (iii) \\ \max_{k, i<k<j-1}\{N[i,k] + N[k+1, j]\} & (iv) \end{cases}$
    end for
  end for
Output: $N[0, n-1]$
```

The recurrence cases used in Nussinov's algorithm includes the possibilities of (i) base-pairing, (ii, iii) a base being unpaired and (iv) branching into two substructures. These decompositions are depicted in Figure 2.3.

Filling the upper right triangle of an $(n \times n)$-matrix requires $O(n^2)$ steps to finish. According to the fourth case of Nussinov's recurrence presented in Algorithm 1, the number of possible branching points to be checked is in $O(n)$. This gives an overall running time in $O(n^3)$. The space complexity is in $O(n^2)$ since only one $(n \times n)$-matrix is used.

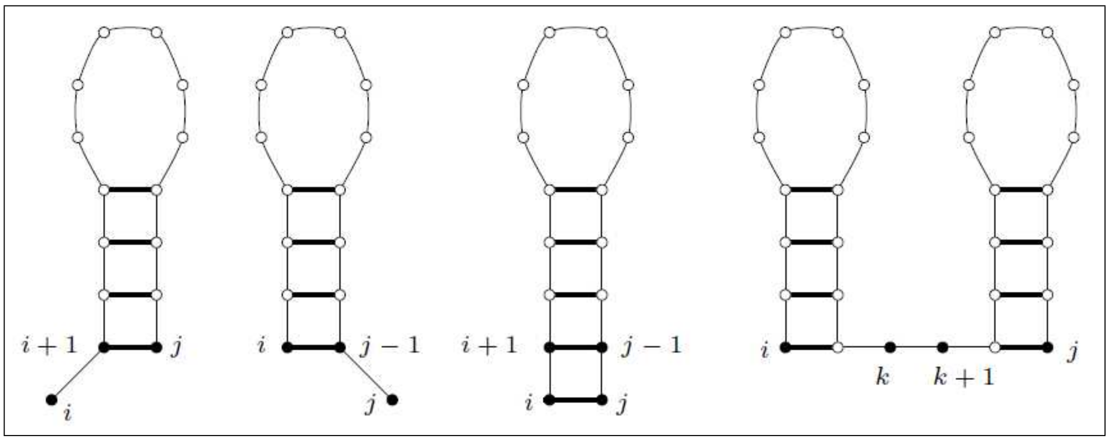


Figure 2.3: Possible sequence decompositions of Nussinov's algorithm. Figure taken from [3]

2.2.2 Minimum Free Energy

Nussinov's algorithm does not consider any preferences for the type of the formed substructures. Another algorithm was proposed by Zuker [29] that takes into account stabilizing and destabilizing effects of the kind of the formed substructures and tries to predict the minimum free energy (MFE). This is done by further decomposition of the formed substructures when enclosed by a given base-pair (i, j). Hairpin loop, stacking loop, interior loop and a branching are the four loop decomposition cases considered by Zucker's algorithm. This algorithm is named *Zuker's Algorithm* also after the name of its designer.

Definition 2.2.3 (MFE structure). *For a given RNA primary subsequence $P_{i,j}$, $\Omega_{i,j}$ denotes the secondary substructure with the minimum free energy. By $FE(S_{i,j})$ we denote the value of free energy of the substructure $S_{i,j}$.*

Definition 2.2.4 (Z-matrix). *For an RNA primary sequence $P_{0,n-1}$, a Z-matrix is an $(n \times n)$-matrix with entries $Z[i,j]$ such that $Z[i,j] = FE(\Omega_{i,j})\ \forall\ 0 \leq i,j < n$ and $S_{i,j}$ is a general secondary substructure of the RNA primary sequence P.*

As we mentioned above, a given substructure enclosed by a base-pair needs further decomposition to analyze its energy components. This leads us to the next definition of the V-matrix.

Definition 2.2.5 (V-matrix). *For an RNA primary sequence $P_{0,n-1}$, a V-matrix is an $(n \times n)$ matrix with entries $V[i,j]$ such that $V[i,j] = FE(\Omega_{i,j})\ \forall\ 0 \leq i,j < n$ and $S_{i,j}$ is a secondary substructure of an RNA primary sequence P subject to the constraint that p_i and p_j are paired. Formally, $(p_i, p_j) \in S$.*

The recurrence equations for Zucker's Algorithm are presented in equations (2.1), (2.2) and (2.3). The consideration of the nature of the formed loops by the algorithm is obvious in the recurrence equation (2.2). $es(i,j)$ is the energy released by the substructure $S_{i,j}$ due to the stacking of the base-pair (i,j). It has a stabilizing effect, i.e., negative value. $eh(i,j)$ is the energy due to the formation of a hairpin loop enclosed by the base-pair

(i, j) with $i - j - 1$ unpaired bases. Moreover, by $eb(i, i', j, j')$ we denote the energy of an interior loop or alternatively a bulge enclosed by the pairing of (i, j). $i' = i + k + 1$ is accessible from i. And k is the number of unpaired bases in the bulge formed at i. Similarly, for j', k' is the number of unpaired bases in the bulge at j. Finally, a is a constant penalty for branching. These values are determined experimentally and are used to establish preferences for different types of loops.

Definition 2.2.6 (M-matrix)**.** *For an RNA primary sequence $P_{0,n-1}$, an M-matrix is an $(n \times n)$ with entries $M[i, j]$ such that $M[i, j] = FE(\Omega_{i,j})$ $\forall$ $0 \leq i, j < n$ and $S_{i,j}$ is a secondary substructure of an RNA primary sequence P subject to the constraint that $S_{i,j}$ is a non-empty structure, i.e., it contains at least one structural element and is a part of a multi-loop.*

Note that, for all matrices involved in Zuker's algorithm, the boundary conditions are $MAT[i, j] = +\infty$ $\forall$ $j - i < \mu$, where MAT is a wildcard for Zuker's matrices and $\mu > 0$.

In the V-matrix recurrence, equation (2.2), the multi-loop branch case of the V-matrix recurrence ensures that it will not recurse to structures other that multi-loops.

Note the difference between the N-matrix recurrence in algorithm 1 and the Z-matrix main recurrence equation (2.1) considering the optimization goals. In the N-matrix recurrence presented in algorithm 1, the goal is obtaining an optimum structure by *maximizing* the values of matrix entries. However, in the Z-matrix case, the goal is *minimizing* the values.

$$Z[i, j] = min \begin{cases} V[i-1, j-1] & (i) \\ Z[i, j-1] & (ii) \\ Z[i+1, j] & (iii) \\ \min_{k, i<k<j-1}\{Z[i, k] + Z[k+1, j]\} & (iv) \end{cases} \tag{2.1}$$

$$V[i, j] = min \begin{cases} eh(i, j) & (i) \\ es(i, j) + V[i+1, j-1] & (ii) \\ \min_{i<i'<j'<j}\{eb(i, i', j, j') + V[i', j']\} & (iii) \\ \min_{k, i<k<j-1}\{M[i+1, k] + M[k+1, j-1] + a\} & (iv) \end{cases} \tag{2.2}$$

$$M[i, j] = min \begin{cases} M[i, j-1] + b & (i) \\ M[i+1, j] + b & (ii) \\ V[i+1.j-1] + c & (iii) \\ \min_{k, i<k<j-1}\{M[i, k] + M[k+1, j]\} & (iv) \end{cases} \tag{2.3}$$

We begin by exploring the Z-matrix recurrence equation (2.1), it proceeds by filling an $(n \times n)$-matrix, which essentially requires $O(n^2)$ steps to finish. However, case (iv) in the recurrence may require $O(n)$ steps in the worst case to cover all branching positions. This leads to an overall running time in $O(n^3)$.

For the V-matrix recurrence presented in equation (2.2), obviously, the interior-loop - case (*iii*) - has the highest complexity of $O(n^2)$. By bounding the size of the interior loop, the overall run time complexity to fill of the V-matrix is bounded by the multi-loop case - case *iv* - which has a worst case running time in $O(n^3)$ as well.

The M-matrix recurrence can finish in the worst case after $O(n^3)$ steps. This leads to an overall running time complexity of Zuker algorithm in $O(n^3)$. Three matrices are used, the size of each is in $O(n^2)$ which is also the space complexity of the algorithm. A straight forward implementation of Zuker's algorithm resembles that of Nusinov's algorithm presented in algorithm 1.

2.2.3 Maximum Expected Accuracy

The maximum expected accuracy folding (MEA) was proposed to be an alternative to the minimum free energy folding in [7, 16]. The probability of the MFE structure $Pr(\Omega)$ is the highest in thermodynamic equilibrium when predicting single structures. However, it is very low in case of long sequences [4].

The MEA folding proposal is to look for the structure with the highest number of *correct* base-pairs. In our work, we consider the base-pairing probability as the measure for its correctness. Moreover, we consider another contribution to the correctness of a given secondary structure. We consider the probabilities of single bases being unpaired as well. This leads to the next definitions.

Definition 2.2.7 (Structure Accuracy). *Let $Pr_{i,j}$ be the probability that a pairing between the bases p_i and p_j will occur in the thermodynamic equilibrium. Also, let Pr_i be the probability that a given base p_i will be unpaired. The structure accuracy of a given secondary structure $Acc(S, P)$ is the summation of probabilities of both its paired and unpaired bases in the secondary structure S and is denoted by*

$$Acc(S, P) = \sum_{(i,j)\in S} (Pr_{i,j} - Pr_i - Pr_j) + \sum_{i\in P} Pr_i \tag{2.4}$$

Definition 2.2.8 (MEA Structure). *For a given RNA primary subsequence $P_{i,j}$, $\Upsilon_{i,j}$ denotes the secondary substructure with maximum structure accuracy.*

The advantage of the MEA folding is that it separates the process of predicting a secondary structure into two models of computation. The first one assumes an already calculated set of probabilities as an input to the prediction algorithms. However, the underlying model for probabilities calculation considers the resulted structural elements in thermodynamic equilibrium. It deals differently with different types of loops. At that lower level, considering other physics related variables to calculate such probabilities is possible. This modularity and separation between a physics-based model and a physics-free one makes it more usable by using a lower level model that captures the preferences of a secondary structure and then using it as an input to fast higher level algorithms.

Accordingly, if those probabilities can be computed efficiently, a Nussinov-style algorithm can be employed to maximize the accuracy. Fortunately, the average base-pairing probability for any pair (p_i, p_j) together with the unpairing probability for any base p_i can be efficiently calculated by `RNAplfold` [32] by considering the ensembles of all secondary structures in thermodynamic equilibrium. It is implemented in an efficient way that requires a running time in $O(nL^2)$ and a space in $O(n + L^2)$ to compute base-pairing probability matrices and base-unpairing probabilities using a sliding window fashion of fixed length L while processing the RNA sequence of length n.

Definition 2.2.9 (A-matrix)**.** *For an RNA primary sequence $P_{0,n-1}$, an Accuracy matrix is an $(n \times n)$-matrix with entries $A[i,j]$ such that $A[i,j] = Acc(\Upsilon_{i,j})\ \forall\ 0 \leq i, j < n$. The boundary condition of the matrix is $A[i,i] = Pr_i\ \forall\ 0 \leq i < n$.*

In algorithm 2, we present a Nussinov-style algorithm to calculate the $MEA(\Upsilon)$ given a primary RNA sequence P. The recurrence is the same as in Nussinov's algorithm. Also, the loop architecture is similar. This algorithm requires $O(n^3)$ steps in the worst case to finish. It consumes only one $n \times n$ matrix which leads to a space complexity of the whole algorithm in $O(n^2)$.

Algorithm 2 Maximum Expected Accuracy Calculation

```
Input:
   1. base-pairing probabilities $Pr_{i,j}\ \forall\ 0 \leq i < j < n$.
   2. base unpairing probabilities $Pr_i\ \forall\ 0 \leq i < n$.
1. Initialization
   for $i = 1$ to $n - 1$ do
      $A[i, i-1] = 0$;
   end for
   for $i = 0$ to $n - 1$ do
      $A[i, i] = Pr_i$;
   end for
2. Recurrence
   for $l = 1$ to $n - 1$ do
      for $i = 0$ to $n - l - 1$ do
         $j = i + l$;
         $A[i,j] = max \begin{cases} A[i-1, j-1] + Pr_{i,j} & (i) \\ A[i, j-1] + Pr_j & (ii) \\ A[i+1, j] + Pr_i & (iii) \\ \max_{k, i<k<j-1}\{A[i,k] + A[k+1,j]\} & (iv) \end{cases}$
      end for
   end for
Output: $A[0, n-1]$
```

Chapter 3

Algorithmic aspects

In this chapter, we present the algorithms used in our work and ideas behind these algorithms. We start by explaining the idea of sparsification and how the triangle inequality can be used to speed up dynamic programming algorithms. We explain how this speed up can be achieved. Finally, we employ such speed up to implement a MEA algorithm presented in algorithm 3. This algorithm will be the engine for our program `RNAMotid`.

3.1 Sparsification

Typically, folding algorithms that consider splitting the structure into two substructures at a given branching point require in their worst case $O(n^3)$ steps to finish. To fold an RNA sequence of length n, they essentially fill an $n \times n$ matrix in $O(n^2)$. Moreover, they need $O(n)$ steps to consider all possible branching points. That leads to the overall running time of $O(n^3)$.

Obviously, the bottleneck for such algorithms is the branching case. Sparsification deals with pruning this case by considering only a constant number of candidate points at which branching into two substructures may only be useful.

A MFE folding algorithm was presented in [37] which considers only a limited number of branching points. The idea behind this implementation is based on the triangle inequality rule. The restriction to certain branching points leads to an overall running time to fold an RNA sequence of length n in $O(n^2\psi(n))$, where $\psi(n)$ was shown to be constant on average. Another algorithm was also designed that exploits the same underlying idea of maximizing the number of base-pairs in the folded secondary structure.

In the next section, we present a new algorithm based on the same idea presented in [37] to find $Acc(\Upsilon_{0,n-1})$, given an RNA primary sequence P. After that, we analyze the running time and the space complexities of this algorithm.

3.2 The new algorithm

The main recurrence equation, upon which the previously presented implementation of the MEA algorithm 2 was based, is presented in equation (3.1).

The previously used recurrence, presented in equation (3.1), has four cases to which it can recurse. However, the unpairing cases can be included as a special branching case where one of the substructures contains only a single base. The recurrence can accommodate this inclusion easily because of the boundary conditions of the matrix which states that $A[i,i] = Pr_i \; \forall \; 0 \leq i < n$.

$$A[i,j] = max \begin{cases} A[i-1,j-1] + Pr_{i,j} \\ A[i,j-1] + Pr_j \\ A[i+1,j] + Pr_i \\ \max_{k, i<k<j-1}\{A[i,k] + A[k+1,j]\} \end{cases} \tag{3.1}$$

We can prove the correctness of transforming a part of equation (3.1) into equation (3.2) by induction. However, we just consider the intuition behind this correctness for simplicity. The term $A[i+1,j] + Pr_i$ can be obtained when substituting for $k = i$ in the maximization term leading to $A[i,i] + A[i+1,j]$. Recall that the boundary conditions state that $A[i,i] = Pr_i$. This leads to a final result from the maximization term of $A[i+1,j] + Pr_i$. Similarly, when substituting $k = j-1$ in the maximization term leading to a final result of $A[i,j-1] + Pr_j$.

Definition 3.2.1 (Component)**.** *Given an RNA secondary structure S, a component C is* either *a substructure $S_{i,j}$ in which the base p_i is paired with p_j, i.e., $(p_i, p_j) \in S$* or *a single unpaired base, where $i = j$.*

Definition 3.2.2 (MEA Component)**.** *For a given RNA primary subsequence $P_{i,j}$, $\Phi_{i,j}$ denotes a component with the maximum structure accuracy.*

Definition 3.2.3 (C-matrix)**.** *Given an RNA primary sequence $P_{0,n-1}$, a C-matrix is an $(n \times n)$-matrix with entries $C[i,j] = Acc(\Phi_{i,j}) \; \forall \; 0 \leq i,j < n$. The boundary condition of the matrix is $C[i,i] = Pr_i \; \forall \; 0 \leq i < n$.*

$$A[i,j] = max \begin{cases} C[i,j] \\ \max_{k, i\leq k<j}\{A[i,k] + A[k+1,j]\} \end{cases} \tag{3.2}$$

$$C[i,j] = max \begin{cases} Pr_i & \text{, if } i = j \\ A[i-1,j-1] + Pr_{i,j} & \text{, if } Pr_{i,j} > 0 \\ 0 & \text{, Otherwise} \end{cases} \tag{3.3}$$

Definition 3.2.4 (A'-matrix)**.** *Given an RNA primary sequence $P_{0,n-1}$, an A'-matrix is an $(n \times n)$-matrix with entries $A'[i,j] = -A[i,j]\ \forall\ 0 \leq i,j < n$. The boundary condition of the matrix is $A'[i,i] = 0\ \forall\ 0 \leq i < n$.*

Definition 3.2.5 (C'-matrix)**.** *Given an RNA primary sequence $P_{0,n-1}$, a C'-matrix is an $(n \times n)$-matrix with entries $C'[i,j] = -C[i,j]\ \forall\ 0 \leq i,j < n$.*

Introducing the A'-matrix and the C'-matrix will enable us to re-formulate equations (3.2) and (3.3) to equations (3.4) and (3.5) so that equation (3.4) follows the *triangle inequality* rule. The inclusion of the unpairing cases in the branching case is still valid, although we change the boundary conditions of the A'-matrix. The boundary conditions are still valid for the C'-matrix and any entry of the form $A[i,i]$ will definitely recurse to the non-branching case. This is because there is no branching point that can satisfy the condition $i \leq k < j$ for such entry. According to the above definitions, the following recurrence equations corresponding to the A'-matrix and the C'-matrix are presented in equations (3.4) and (3.5).

By dividing equation (3.1) into two recurrence equations, we achieve two goals. By introducing equation (3.5), we separated the recurrences responsible for structural decomposition from the other terms contributing to the value being calculated. This separation will be used later in our arguments. Next, we introduce the triangle inequality rule.

$$A'[i,j] = min \begin{cases} C'[i,j] \\ \min_{k, i \leq k < j}\{A'[i,k] + A'[k+1,j]\} \end{cases} \tag{3.4}$$

$$C'[i,j] = min \begin{cases} -Pr_i & \text{, if } i = j \\ A'[i-1,j-1] - Pr_{i,j} & \text{, if } Pr_{i,j} > 0 \\ 0 & \text{, Otherwise} \end{cases} \tag{3.5}$$

Definition 3.2.6. *A matrix M is said to follow the triangle inequality rule iff*

$$\forall\ i < k < j \qquad M[i,j] \leq M[i,k] + M[k+1,j]$$

From the above definition, it's easy to deduce that the A'-matrix, as computed from recurrence equation (3.4), follows the triangle inequality rule. This is obvious from the re-formulation of the recurrence due to the fact that a given entry in the A'-matrix is computed recursively via a minimization term. This term involves two competing values - $\min_{k,i \leq k<j}\{A'[i,k]+A'[k+1,j]\}$ and $C'[i,j]$. The first of these values - $\min_{k,i \leq k<j}\{A'[i,k]+ A'[k+1,j]\}$ - is the branching case which is the right side of the triangle inequality rule. So, the final value for a given entry will eventually come from either the branching or from a higher C'-matrix value of a component. This is the intuition behind the " $\leq$ " operator.

The whole intuition behind using this idea is explained by proofing the following theorem. This theorem claims that any given entry in the A'-matrix whose value was computed by recursing into two sub-problems, i.e., the branching case was chosen from the minimization term, can be reformulated as a summation of the solutions of another or the same two sub-problems where the first sub-problem's optimal solution is computed via considering it as a component, i.e., the first part of the structure is a component. The proof will be done by contradiction.

Theorem 1.

$$\forall\ i,j\ ((\exists\ i \leq k < j \wedge A'[i,j] = A'[i,k] + A'[k+1,j]) \Rightarrow \\ (\exists\ k' \leq k \wedge A'[i,j] = A'[i,k'] + A'[k'+1,j] \wedge A'[i,k'] = C'[i,k']))$$

Proof. Let k be the smallest index at which the statement $A'[i,j] = A'[i,k] + A'[k+1,j]$ would be true. Now, we will assume that $A'[i,k] < C'[i,k]$ and then obtain a contradiction. According to equation (3.4), $A'[i,k] = A'[i,k'] + A'[k'+1,k]$ for some $i \leq k' < k$. But the A'-matrix follows the triangle inequality rule, which lead to the following inequality

$$A'[i,j] = A'[i,k'] + \underline{A'[k'+1,k] + A'[k+1,j]} \geq A'[i,k'] + \underline{A'[k'+1,j]}$$

which contradicts the hypothesis that k is the smallest index at which branching is possible. □

The intuition behind the above proof by contradiction is by arguing that if the first sub-problem - $A'[i,k]$ - of the main two subproblems - $A'[i,k]$ and $A'[k+1,j]$ - did not recurse to the component case in the main recurrence equation - equation (3.4) - then it should have recursed using the branching case into another two sub-problems, leaving a total of three subproblem namely $A'[i,k''], A'[k''+1,k']$ and $A'[k'+1,j]$ instead of only two. However, according to this new combination of sub-problems, there is a possibility that the last two mentioned sub-problems can - *combined together* - constitute a component whose value may be the optimal. This additional potential to form a component by combining the old sub-problem - $A'[k'+1,j]$ - with the new subproblem - $A'[k''+1,k']$ - is the intuition behind our proof.

$$A'[i,j] = min \begin{cases} C'[i,j] \\ \min_{k, i \leq k < j}\{C'[i,k] + A'[k+1,j]\} \end{cases} \tag{3.6}$$

According to theorem 1, the recurrence in equation (3.4) can be represented as in equation (3.6).

Even with a direct application of the recurrence in equation 3.6, $O(n)$ comparisons should be carried out to find out the optimum branching point. But by taking a closer look we can observe that some redundancies are taking place in the computation of the optimum score. By Investigating these comparisons, a dominance behaviour was observed between them.

Theorem 2.

$$\begin{aligned}\forall\ i,j\ ((\exists\ i<k<j \wedge C'[i,j] \geq C'[i,k]+A'[k+1,j]) \Rightarrow \\ (\forall\ j' \geq j \wedge C'[i,j]+A'[j+1,j'] \geq C'[i,k]+A'[k+1,j']))\end{aligned}$$

This dominance relation is formalized in theorem 2. The theorem limits the number of branching points that needs to be checked by exploiting this dominance relation. It shows that considering a given point k for branching comparisons in a given sequence $P_{i,j}$ can only be beneficial if the score of considering the prefix $P_{i,k}$ *as a component* is better than considering it as a combination of two smaller subproblems. This means that we are only interested in branching points that lead to scores that can *not be dominated* by others according to definition 3.2.7. This criteria is formalized in definition 3.2.7. The proof of this theorem can also be done by contradiction in the same way as theorem 1 using the triangle inequality.

Definition 3.2.7. *An index j is considered as a branching candidate iff*

$$\forall\ i<k\leq j \qquad C'[i,j] < A'[i,k]+A'[k+1,j]$$

To make use of this dominance relation, a different order of filling the A' matrix will be employed. An order that keeps track whether the score of a given *prefix* is computed via considering it as a component or as a combination of two subproblems. According to the requirements and the knowledge that we want to acquire before processing a given subsequence, the matrix is filled, in the order of increasing the length of the prefixes of the suffixes of the sequence. The rows of the matrix are filled bottom first then top. In each row, the columns are filled from left to right. The filling order of the A' matrix is visualized in figure 3.1. Moreover, the corresponding order of processing the input RNA sequence is visualised in figure 3.2.

An implementation of the MEA sparse version is presented in algorithm 3. We will start by explaining how the *candidate list*[1] is filled up. Then, we will discuss the running time and space complexities.

Essentially, at the beginning of filling each row i, the candidate list is empty. Accordingly, the flow in execution will reach the 9th line of the iteration step, at the conditional statement. In the beginning of processing each row, the condition will always evaluate to true. This fact is ensured by the initialization step. In the initialization, each entry $C'[i,i]$ stores the negative of the unpairing probability Pr_i, which is guaranteed to be smaller that 0 that is stored in $A'[i,i]$ during the initialization step.

The index of the single unpaired base $P_{j=i}$ is added to the candidate list at the beginning of filling each row because, it satisfies the criteria we stated in definition 3.2.7. By this,

[1]The list containing the points that fulfill the criteria of being split points as stated in definition 3.2.7.

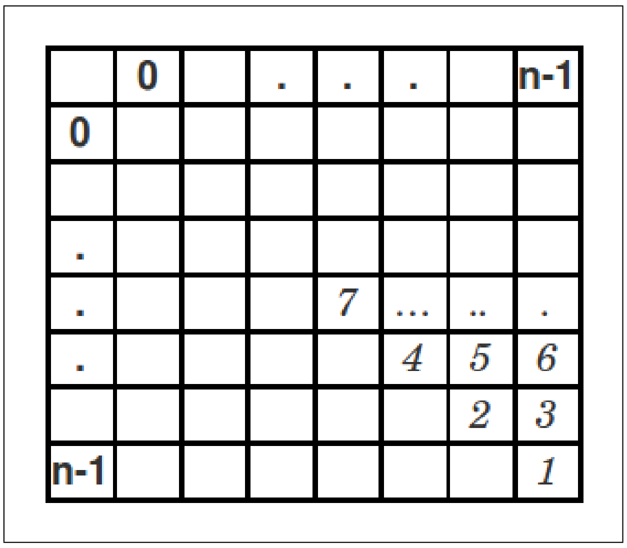


Figure 3.1: The filling order of the A' matrix.

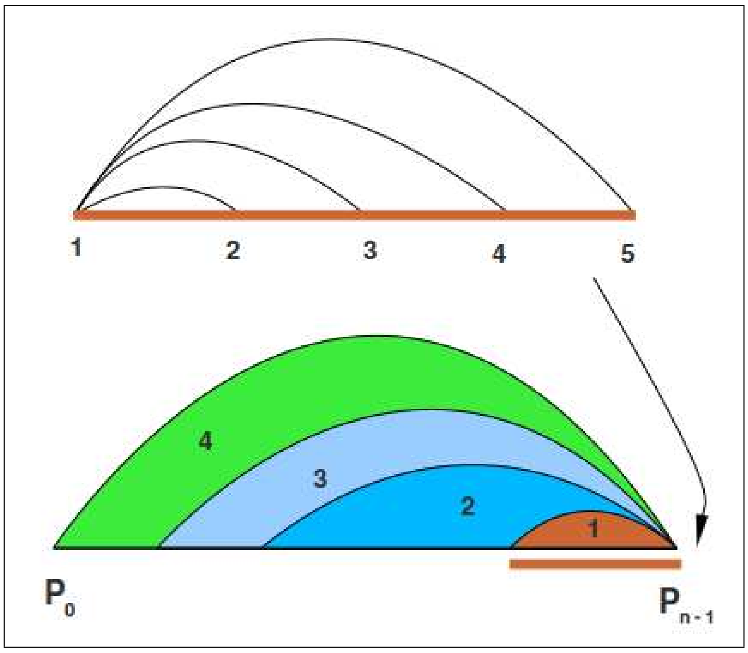


Figure 3.2: The processing order of the input RNA sequence.

we mean that this single unpaired base can be considered as a candidate split point for longer sequences $P_{i...j...k}$. That's because, considering $P_{i...j}$ as a component during its processing was better than considering it as a combination of two subproblems.

After that, for any entry j in a given row i, the matrix entry $A'[i,j]$ is filled with the optimum score considering all splitting points k previously added to the candidate list during the processing of smaller sequences up to $P_{i...k...j-1}$. These previous checks makes sure that when the flow reaches the conditional statement the condition is evaluated to true and accordingly a new entry is added to the candidate list only if this new score of this new candidate point j was not dominated by the score of any previously added points in the candidate list. The iteration step continues till the end of the current suffix being processed then the algorithm moves on to the next longer suffix and so on as illustrated in figure 3.2.

Complexity Analysis

In [37], the expected growth of the candidate list size with respect to increasing the input sequence size - where the candidate list is filled in the same fashion described here - was shown to be converging to a constant factor $O(\psi(n))$. This leads to an overall running time of the algorithm in $O(n^2\psi(n))$ on average. However, in our case, an additional case in the recurrence was added that counts for base unpairing. Now, we are left to prove that this additional case we introduced in equation (3.5) will not affect the overall running time of the algorithm.

Eventually, an additional element will be essentially added to the candidate list counting for the additional recurrence case at which $i = j$ for each row j. Since for each row there exists only a single entry for which $i = j$ is true, then this is the whole additional contribution to the candidate list considering the unpairing case. Asymptotically, this means that a constant factor c is added to $O(\psi(n))$ giving an overall running time in $O(n^2(\psi(n)+c))$. Which still leads to an overall running time of the algorithm in $O(n^2\psi(n))$.

In the algorithm two matrices were used each of maximum size in $O(n^2)$ leading to an overall space complexity of the algorithm in $O(n^2)$.

Accuracy Function

Algorithm 3 calculates the maximum accuracy as defined in subsection 2.2.3. However, the model we introduce here should consider also a kind of locality of the predicted RNA elements. This idea of locality imposes reformulation of our previous definition of accuracy. A new function should be devised that ensures the locality of the predicted RNA elements. For example, a cis-acting RNA element is considered to be local relative to the mRNA molecule. Also, a trans-acting RNA element is relatively local from the perspective of the whole genome.

We introduce here the formula of an accuracy function. A function that fulfills our criteria of locality of the predicted RNA elements. This function is introduced in equation 3.7.

Algorithm 3 Maximum Expected Accuracy Calculation - Sparse Version

Input:

1. base-pairing probabilities $Pr_{i,j}$ $\forall\ 0 \leq i < j < n$.
2. base unpairing probabilities Pr_i $\forall\ 0 \leq i < n$.

I. Initialization

```
 1: for $i = 1$ to $n - 1$ do
 2:    $A'[i, i-1] = 0$;
 3:    $C'[i, i-1] = 0$;
 4: end for
 5: for $i = 0$ to $n - 1$ do
 6:    $A'[i, i] = 0$;
 7:    $C'[i, i] = -Pr_i$;
 8:    for $j = i$ to $n - 1$ do
 9:       if $Pr_{i,j} > 0$ then
10:          $C'[i, j] = -Pr_{i,j}$;
11:       end if
12:    end for
13: end for
```

II. Iteration

```
 1: for $i = n - 1$ to $0$ do
 2:    candidate-list ← NULL
 3:    for $j = i$ to $n - 1$ do
 4:       for all $k$ such that $k \in$ candidate-list do
 5:          if $A'[i, j] < C'[i, k] + A'[k + 1, j]$ then
 6:             $A'[i, j] = C'[i, k] + A'[k + 1, j]$
 7:          end if
 8:       end for
 9:       if $C'[i, j] < A'[i, j]$ then
10:          $A'[i, j] = C'[i, j]$
11:          Append $j$ to candidate-list
12:       end if
13:    end for
14: end for
```

Output: $-A'[0, n - 1]$

$$Accuracy = \frac{Base\ pairing\ probabilities + Base\ unpairing\ probabilities}{Start\ value + Subsequence\ length^{degression}} \tag{3.7}$$

With a function of this form we can make sure that small RNA elements are predicted as well as long RNA elements. Two more variables - namely the start value and the degression - are introduced. By altering the values of these variables the variance of the function output with the length can be modified. This makes us able to alter the behaviour by which the function reacts to the change in the length parameter. Moreover, by training such parameters we can alter the performance of our program according to different input nature or family as we are going to see in chapter 5.

The curves depicted in figures 3.3 and 3.4 demonstrate the effect of introducing such variables and the change in the behaviour of the function when altering such variables.

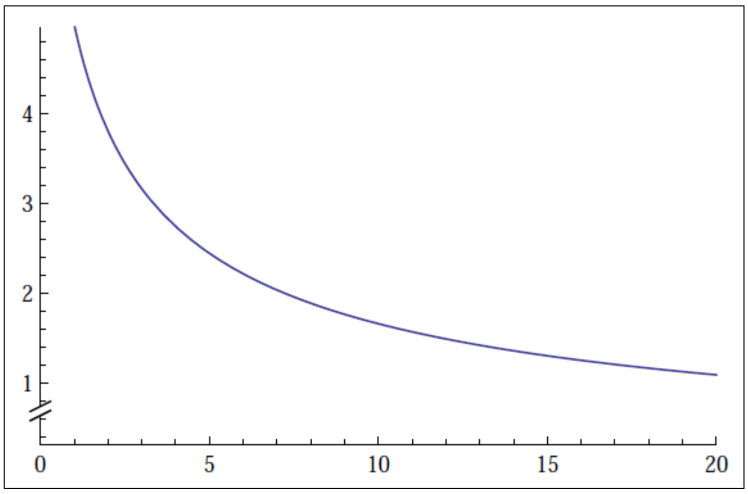


Figure 3.3: The curve of the accuracy function with *degression* = 0.7 and *start* = 1 for a domain from 0 to 20

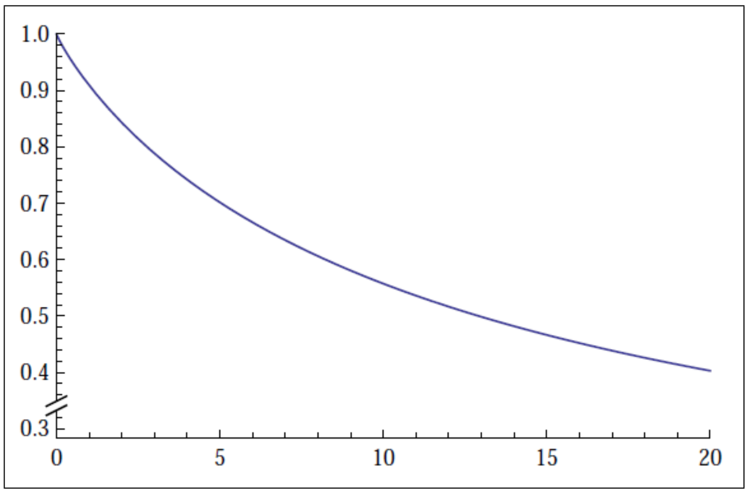


Figure 3.4: The curve of the accuracy function with *degression* = 0.9 and *start* = 10 for a domain from 0 to 20

Chapter 4

Implementation details

In this chapter we explore the implementation details of `RNAMotid` and present the features included in the program. Moreover, we introduce each module of the program and explain the implementation details of it. Also, we introduce the mechanism of the windowing approach and discuss the ability of the program to scan long sequences efficiently. Then we look at the big picture of the program and explain how each module fits into this picture.

4.1 RNAMotid in Modules

4.1.1 Probability Calculation Module

This module is responsible for preparing base-pairing and base-unpairing probabilities of the input RNA sequence to be used by next modules. Base-pairing and base-unpairing probabilities can be obtained via two possible ways.

The first way is through the functions provided by the Vienna RNA library [17]. The `pfl_fold()` function provided by the Vienna RNA library can be used to calculate base-pairing and base-unpairing probabilities given an RNA input sequence and other relevant parameters. Such parameters include the maximum span, a cutoff and the unpairing length. The maximum span specifies the maximum length for a valid base-pairing. A cutoff acts as a lower limit for the calculated base-pairing probabilities. Moreover, the unpairing length determines an upper limit for the region length up to which the mean probabilities of being unpaired are computed. Finally, a folding window size is provided.

The other approach to obtain base-pairing and base-unpairing probabilities for a given sequence is to rely on a pre-computed probabilities stored in a PostScript dot-plot file. This file can be generated by `RNAplfold` program [32] included in the Vienna RNA package [17]. By using this PostScript file, the probability calculation module becomes capable of obtaining the base-pairing probabilities. For the base-unpairing probabilities, another file can be generated by the `RNAplfold` program [32] that contains the base-unpairing probabilities. The module is capable of parsing such file and storing the base-unpairing probabilities as well.

By finishing the execution of this module, `RNAMotid` is done preparing the base-pairing and base-unpairing probabilities for the input RNA sequence. Base-pairing and base-unpairing probabilities can be considered as two of the main inputs to `RNAMotid`, accordingly, making a specialized module to obtain those probabilities is important. By the presence of such specialized module, probabilities can be obtained using any other approaches with simple modifications in this module without any requirements to change other modules.

Windowing probability calculations

A windowing mechanism is included in this module. This mechanism removes any dependency on the `pfl_fold()` function concerning the longest allowed RNA sequence that can be processed at a time. Also, it enables the program to calculate base-pairing and base-unpairing probabilities for large sequences without having any difficulties in allocating memory. With this windowing mechanism, the module is capable of making successive calls to `pfl_fold()` function to process a long RNA input sequence whose total length may exceed the limits of the `pfl_fold()` function.

Between two successive calls to `pfl_fold()` an overlapping sequence is considered. One can argue that the length of this overlapping sequence can be determined according to the maximum length of a valid base-pair. By adapting this approach, it is not possible that the module will miss any probable base-pairing. However, this approach implies a dependency of the computed base-pairing probabilities on their positions in the *probability calculation window*. This dependency comes from the fact that bases which are located at the beginning of the window are averaged over a less number of `pfl_fold()` windows.

Therefore, `RNAMotid` extends the sequence length of the current probability calculation window by `RNAplfold` window size on each side - left and right ends - of the probability calculation window. However, we ignore any base-pairs in these regions. By considering such overlapping sequences, we make sure that we are not ignoring any potential base-pairing probabilities due to the windowing approach. Moreover, we make sure that the bases at the extremes of the *probability calculation window* are averaged over the same number of `RNAplfold` windows. These additional paddings remove any dependency of the calculated probabilities on the chosen folding window.

Generating dot-plot PostScript file

When choosing the first way to compute base-pairing and base-unpairing probabilities, the probability calculation module gives the option to export those base-pairing probabilities to a PostScript file. This PostScript file contains a dot-plot representation of the base-pairing probabilities. Using the dot-plot file, base-pairing probabilities can easily be parsed and visually inspected. Generating the PostScript file that contains the dot-plot can be done by calling the `PS_dot_plot_turn` function provided by the Vienna RNA library.

Complexity Analysis

The running time complexity of the `pfl_fold()` function is in $O(Lw^2)$ with L being the chunk for calling the function and w is the `RNAplfold` window size parameter.

Let n be the total length of the sequence to be processed by the probability calculation module and m be the length of a chunk to `pfl_fold()` - the probability calculation window size. The `pfl_fold()` function will be called n/m times. This implies an overall time complexity of $O((n/m)(m + 2w)w^2)$ which equals $O(nw^2)$. The $2w$ term in the $m + 2w$ factor comes from the additional sequence size when calling `pfl_fold()`.

Considering the space complexity of the module, the space allocated by a function call is in $O(L + w^2)$. However, the calculated probabilities are stored after each call in an $(n \times n)$-matrix which leads to an overall space complexity in $O(n^2)$.

4.1.2 Sequence Folding Module

The folding module makes use of the previously computed probabilities to execute the sparse folding algorithm and prepares information for the traceback module.

This module uses a window scanning approach similar to the one implemented in the probability calculation module. However, the length of the overlapping sequence considered between successive executions of the folding algorithm is different. This overlapping length is determined according to the maximum allowed motif length. By considering an overlapping sequence of such length, `RNAMotid` makes sure that no motifs are ignored because of the windowing mechanism.

Complexity Analysis

Let n be the length of the input RNA sequence to the algorithm. Let m denote the maximum allowed motif length. w denotes the size of a chunk to the sparse folding algorithm - scan window size.

The running time of the sparse folding algorithm is $O(n^2\psi(n))$ as mentioned in section 3.2, where n is the length of the algorithm input sequence, $\psi(n)$ is proved to be constant on average in [37].

According to the windowing mechanism described, the algorithm will be executed n/w times. This lead to an overall running time complexity of $O((n/w)(w + m)^2\psi(w + m))$. The m term in the $w + m$ factor comes from the length of the addition considered sequence when calling the algorithm.

Essentially, the module allocates three matrices, each has a size in $O((w + m)^2)$ leading to an overall space complexity in $O((w + m)^2)$.

4.1.3 Traceback Module

In this module, the information gathered by the folding module is used to reproduce the corresponding structures. Each piece of information denotes how a given structure could be obtained from a smaller one. A call to the trace back module is made after the folding module is finished with a given window.

For a given folding window, the traceback module iterates through a matrix whose size is in $O(n^2)$ with n being the length of the folding window including any additional overlapping sequences between successive windows. In the worst case, an entry in this matrix will require $O(n)$ steps to finish. This leads to a worst case running time in $O(n^3)$.

Overlapping Structures

The traceback module is capable of detecting traced structures that are not contributing any additional base-pairs. While tracing a given structure, every base-pair added to the structure is marked only if it was previously involved in any traced structure with higher accuracy value. If the current structure has a higher accuracy value, it takes the priority. By keeping track of the status of each base-pair, this feature is implemented in a way that does not increase the order of the running time.

The folding module ensures the validity of the final output. This is done after the algorithm is executed for the whole input sequence. The set of output motifs is checked against the specified input arguments by the user.

Moreover, if any sorting is required for the final output before passing it over to the exporting module, the traceback module executes an in-place $O(n^2)$ sorting algorithm for the gathered structures. Since the module is responsible for gathering all output motifs during the whole life time of the program, it is more critical to choose an in-place sorting algorithm as long as it is still within the essential running time of the module.

Motifs Regions

`RNAMotid` is also capable of computing motifs regions. By motifs region we mean the continuous interval of input RNA sequence that contain at least a part of a valid output motif. A set of those regions is also prepared for the data exporting module to take over the task of writing them to the disk.

4.1.4 Data Exporting Module

In the previous module, a final set of output motifs was created. This set was ensured for validity as well. In this module, the task of writing output files is carried out. The modularity of this task is important because it enables the program to output additional different file formats that make use of the same computed output information. This can be done by adding modifications to a single part of the program without the need to alter other modules.

Currently, `RNAMotid` generates `FASTA` file formats for both the output motifs and the output motifs regions. Moreover, it generates `CSV` files that could be used in data visualization and output analysis.

4.2 RNAMotid: The big picture

In the previous section we introduced each module in detail and discussed the features presented by each one. Moreover, we discussed the running time and space complexities for each module separately. In this section we zoom out to view the big picture of `RNAMotid` and see how those previously presented modules can fit together.

4.2.1 Window Scanner

We start this section by introducing the windowing mechanism. One of the functional requirements of `RNAMotid` is to be able to deal with input RNA sequences that are in genome-wide scale. Moreover, it is also required that this windowing approach does not affect the output due to the window position. In order to achieve this, a promising windowing approach was implemented.

The windowing manager is responsible for initiating the execution of other modules. It is also responsible to define and update the input borders to other modules and update them upon the termination of those modules. The update of the borders depends on some parameters. Theses borders are used to define a *section* of the input RNA sequence. Some of these parameters depend on the input arguments specified by the user. The windowing manager makes use of these parameters to ensure that the modules' output is independent on the current position of the window. The pig picture of `RNAMotid` is described in figure 4.1.

In order to achieve this independency, the windowing manager adds padding sequences to that computed section. By adding such paddings we can make sure that the computed output is not affected by the position of a window. Those paddings are visualized in figures 4.4, 4.2 and 4.3.

From a probability calculation module perspective, the length of such paddings can be set to two `RNAplfold` window sizes at both sides of the scan window as discussed in subsection 4.1.1. However, from a folding module perspective, the length of the padding sequences should be the maximum allowed motif length as discussed in 4.1.2. By adding this padding to the left side of the scan window - the side at which the execution of the sparse algorithm ends, we make sure that there is no possibility that in a given folding window an output motif starts in a given window and ends in another one. Accordingly, this ensures that no output motifs are missing. Accordingly, the total length of the padding sequences is calculated as the summation of the length of two `RNAplfold` windows - at the extremes - plus an additional maximum motif length - on left side of the window. This window structure is depicted in figures 4.4 and 4.3.

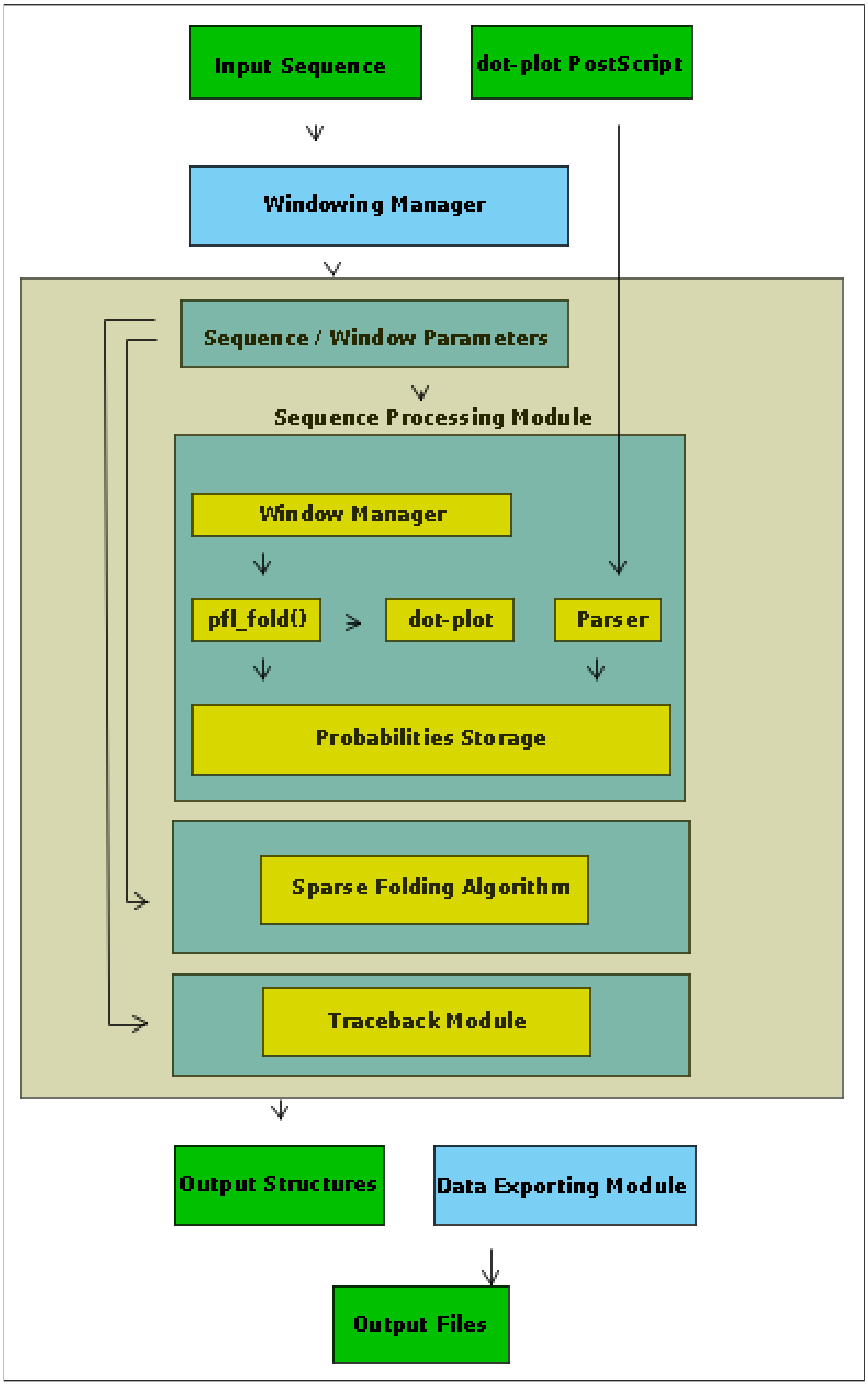


Figure 4.1: A flow chart describing the flow of the control in RNAMotid.

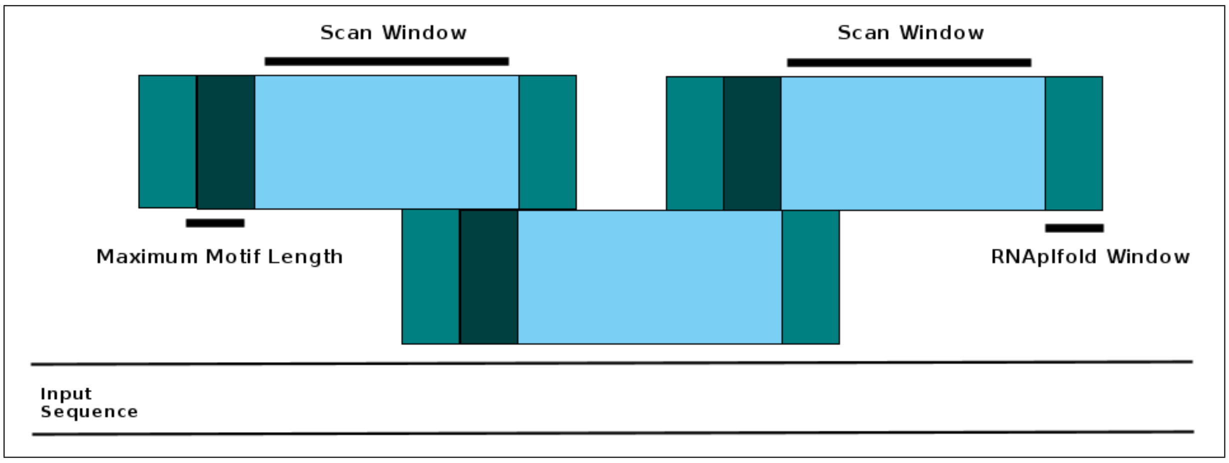


Figure 4.2: A diagram visualizing the big picture of the windowing mechanism implemented in `RNAMotid`. This window depicted in the figure is referred to as the program window. A *scan window* size specifies the maximum sequence length to be scanned for motifs at one time excluding any additional sequences that should be added to ensure the independency of the result on the position of the window or on being local or global in folding. A *maximum motif length* is also added to the program window which is considered by the folding algorithm to ensure that no motifs are missing in the output because of windowing. An additional sequence of length `RNAplfold` window size is also added to the program window to ensure - when calculating probabilities by the probability calculation module - that base-pairing probabilities of bases at the extremes of the program window are averaged over the same number of windows as inner ones by the `pfl-fold` function. These three value combined together form the final program window length and is calculated by the windowing manager.

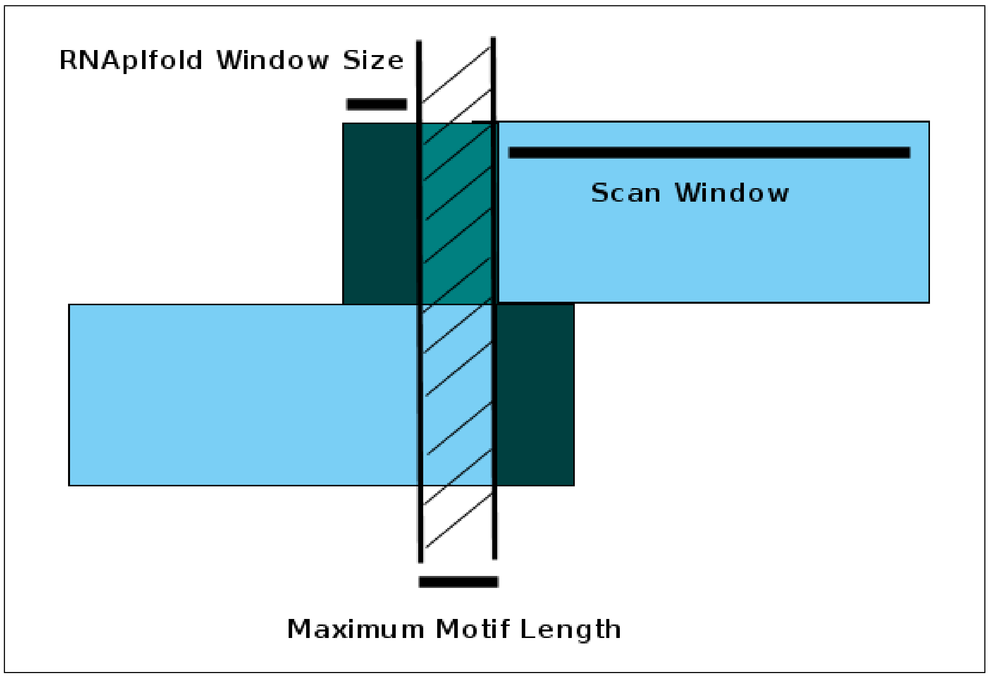


Figure 4.3: Repeated motifs in the shaded section are detected and included only once in the final output.

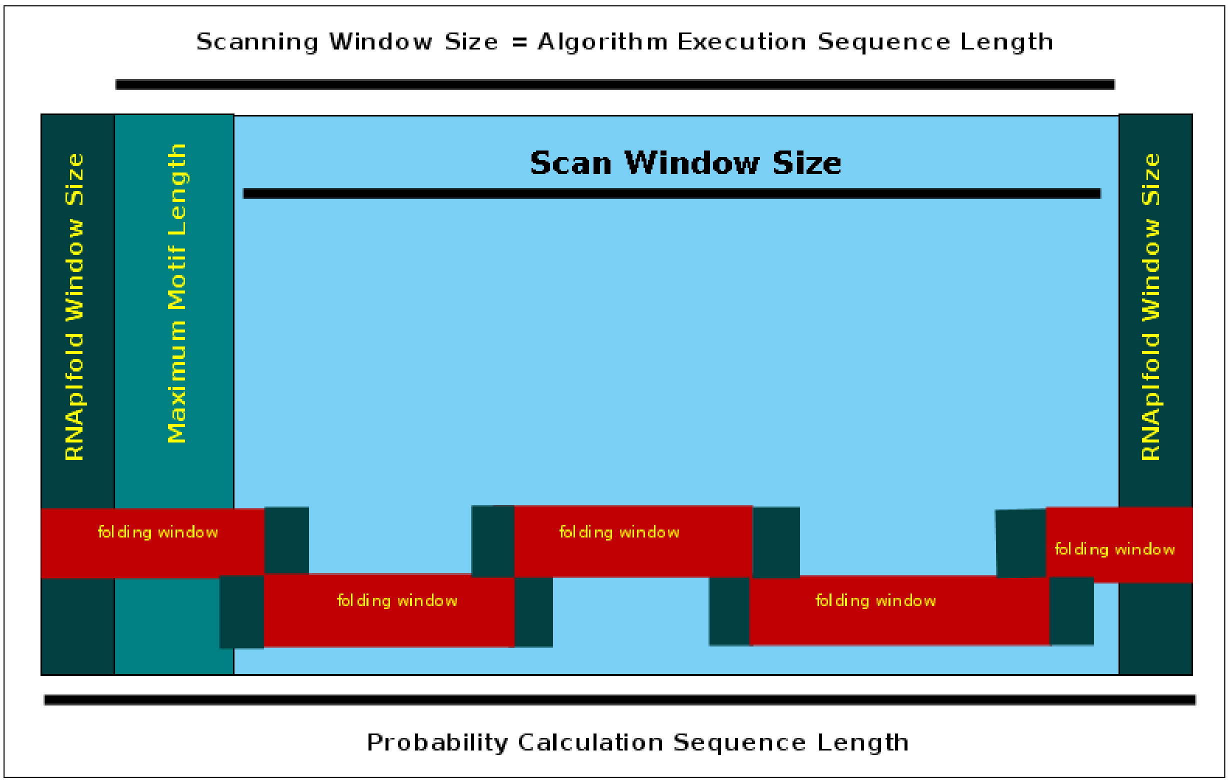
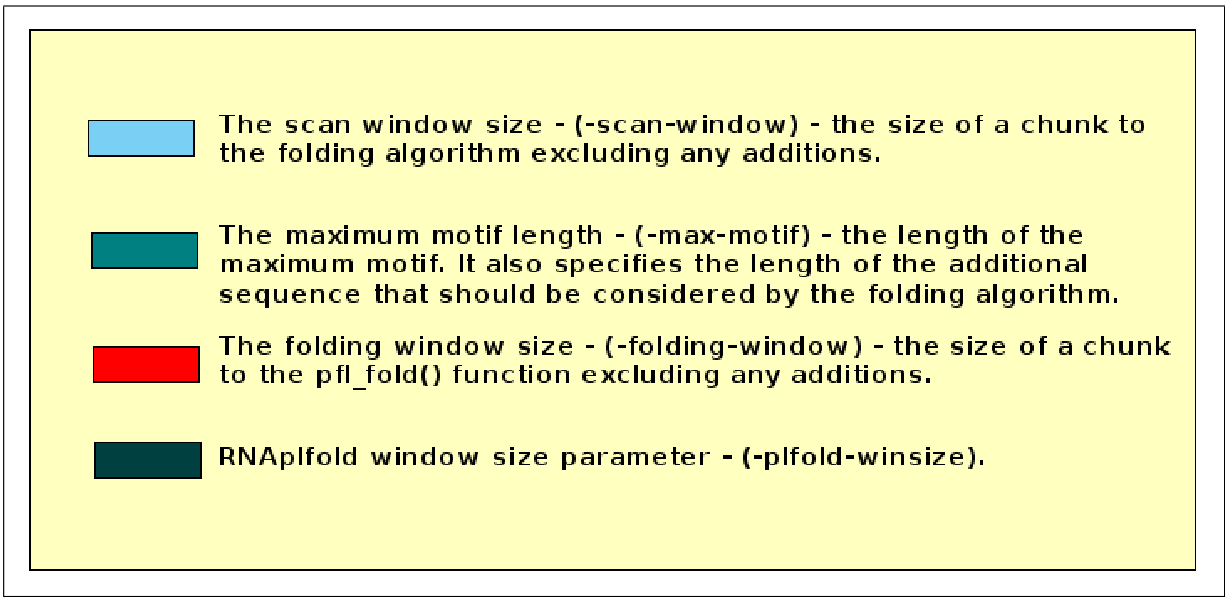


Figure 4.4: A visualization of the windowing mechanism implemented in `RNAMotid` showing the constituents of the program window and inner windowing mechanism.



🟦 The scan window size - (-scan-window) - the size of a chunk to the folding algorithm excluding any additions.

🟩 The maximum motif length - (-max-motif) - the length of the maximum motif. It also specifies the length of the additional sequence that should be considered by the folding algorithm.

🟥 The folding window size - (-folding-window) - the size of a chunk to the pfl_fold() function excluding any additions.

⬛ RNAplfold window size parameter - (-plfold-winsize).

Figure 4.5: A legend for figures 4.4 and 4.2.

Chapter 5

Evaluation and results

In this chapter, we evaluate the performance of `RNAMotid`. in section 5.1, we start by evaluating the ability of the program to identify local motifs in a randomly shuffled context. Then we evaluate the ability of the program to deal with genome-wide sequences in section 5.2.

5.1 Motif identification evaluation

A data set consisting of 4016 sequences collected from 134 different families if cis-regulatory motifs from the `Rfam` database [33] is used. The sequences have varying lengths. Each family has different number of sequences in the data set.

The evaluation was carried out to estimate the ability of `RNAMotid` to identify known motifs. These motifs are embedded in a randomly shuffled context. The length of the shuffled context is 200 nucleotides on both ends of the known motif. A simple parameter training was performed using this data set. The training considered two parameters, seed and degression, to find out which values give the optimum performance. First, the parameters were trained for the whole data set considering all included families. After that, a family-based parameter training was performed for the same parameters. They were tuned to obtain the optimum performance per family.

`RNAMotid` output is collected for each sequence in the data set. For each sequence in the data set, a set of candidate output motifs within the specified suboptimum threshold are produced by the program. Each individual output is classified whether it is an *individual true* or an *individual false*. An output is classified as an individual truth iff it overlaps with the known motif with an allowed tolerance length for the mismatch. The tolerance length is 5 percent of the known motif length. The *final result* of a sequence is determined according to the greater of individual trues and individual falses. The *individual positive predictive value* - or alternatively the *precision* - is calculated as shown in equation 5.1. Similarly, the *final positive predictive value* is calculated as shown in equation 5.2.

$$\text{Individual PPV} = \frac{\text{Total Individual Trues}}{\text{Total Individual Falses} + \text{Total Individual Trues}} \tag{5.1}$$

$$\text{Final PPV} = \frac{\text{Total Final Trues}}{\text{Total Final Falses} + \text{Total Final Trues}} \tag{5.2}$$

5.1.1 Overall parameter tuning and evaluation

When tuning the parameters for the whole data set considering all families, a sample from each family is taken. The size of such sample is at most 10 sequences per family. The seed and degression parameters were found to have an optimum value of 65 for the seed and an optimum value of 0.8 for the degression considering the overall evaluation. Table A.1 shows the values for other parameters used to perform an overall performance evaluation. Detailed results are shown in tables from table A.2 to table A.9. The results are categorized according to the family category and sorted descendingly according to the final PPV values. The *Avg Ex* column denotes the average exact value, which is the average of the accuracies of motifs identified when restricting the scan boundaries for `RNAMotid` to the boundaries of the known motif. The *Avg Fin* column denotes the average final value, which means the average of the highest accuracy motifs identified by `RNAMotid` when scanning the whole sequence including the shuffled context. By comparing these values, we can have an estimate for the value that the suboptimum threshold parameter can take in order to include motifs from this family in the output.

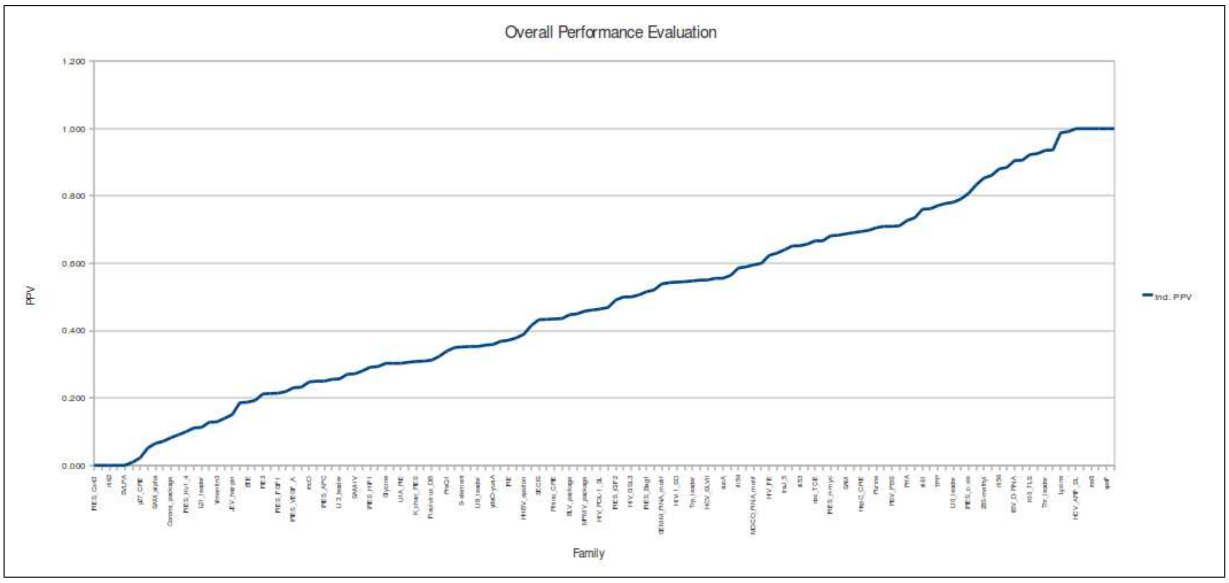


Figure 5.1: A graph visualizing the individual PPV value for each family after performing the overall parameter tuning for `RNAMotid`

Overall evaluation analysis

The underlying criteria for `RNAMotid` for favouring the score of a given section in the input RNA sequence is based mainly on the calculated base-pairing probabilities by `RNAplfold`.

Accordingly, the program favours those highly structured regions in the input sequence.

The overall performance for each category is summarized in table A.10. The *true families count* column indicates the number of families in a given category with an individual PPV greater than 0.5. From this table, we can have an idea about the performance of each category. However, the number of families in each category is not the same. The *Category-relative PPV* column gives a picture of the performance of the families in a given category separately. However, the number of families in each category is not the same.

From this table, we can see that some categories, like the *cis-reg(Riboswitch)*, have high category-relative PPV. Also, some categories are performing bad. For example, the *cis-reg(IRES)* category. Next, we are going to investigate the reasons behind the low performance of some categories even after applying a simple parameter training.

We take a sample from the HIV-1_SL4 family from the evaluation results shown in table A.3. One can notice the large difference between the scores of motif regions suggested by `RNAMotid` and the region where the real motif is located. We can investigate the reason behind such big difference by going back to the underlying criteria of calculated base-pairing probabilities as mentioned before.

In figure 5.2, a dot-plot depicting base-pairing probabilities calculated by `RNAplfold`. The region where the real motif resides is indicated by a double line and the region where the `RNAMotid` suggested motif with the highest score resides is indicated by a single line. From this figure, we can notice that the real motif regions is not containing any potential base-pairing probabilities. In other words, the region containing the real motif is not structured. However, if we take a look at the region of the suggested motif, it's obvious that the suggested motif region by `RNAMotid` with the highest score is much more structured. Moreover, it's obvious that the base-pairing probabilities in this regions is much higher than the region containing the real motif.

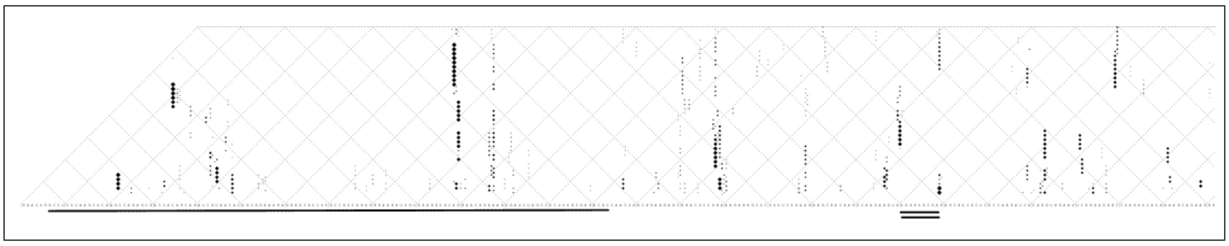


Figure 5.2: A dot-plot generated by `RNAplfold` visualizing base-pairing probabilities for a sample from the HIV-1_SL4 family. The single line indicates the region where the highest score motif resides and the double line indicates the region where the real (known) motif resides.

5.1.2 Family-based evaluation

Another parameter tuning is carried out for the same data set. However, this time the tuning aimed at optimizing the same parameters - the seed and the degression - for each family separated. The program output is then observed. A sample of at most 30 sequences is taken from each family from the data set. Detailed results for some chosen families are shown in tables from table A.11 to table A.13. The results are categorized according to the family category and sorted descendingly by final PPV values.

From these tables, we can see that an improvement is done in the performance of `RNAMotid` after using the parameters that were trained per family. For example, the *IRES(Bip)* family from the *cis-reg(IRES)* category. The performance (PPV) increased from 0.255 to 0.727, which is a high improvement. Recall that the *cis-reg(IRES)* category was showing a bad overall PPV as discussed in the subsection 5.1.1.

Moreover, we can see in figure 5.3 the improvement in the performance of `RNAMotid` after performing the family-based parameter tuning relative to the previous individual PPV value after performing an overall parameter tuning. The individual PPV value is dropped with some families as a result of increasing the sample size when performing the family-based parameter tuning.

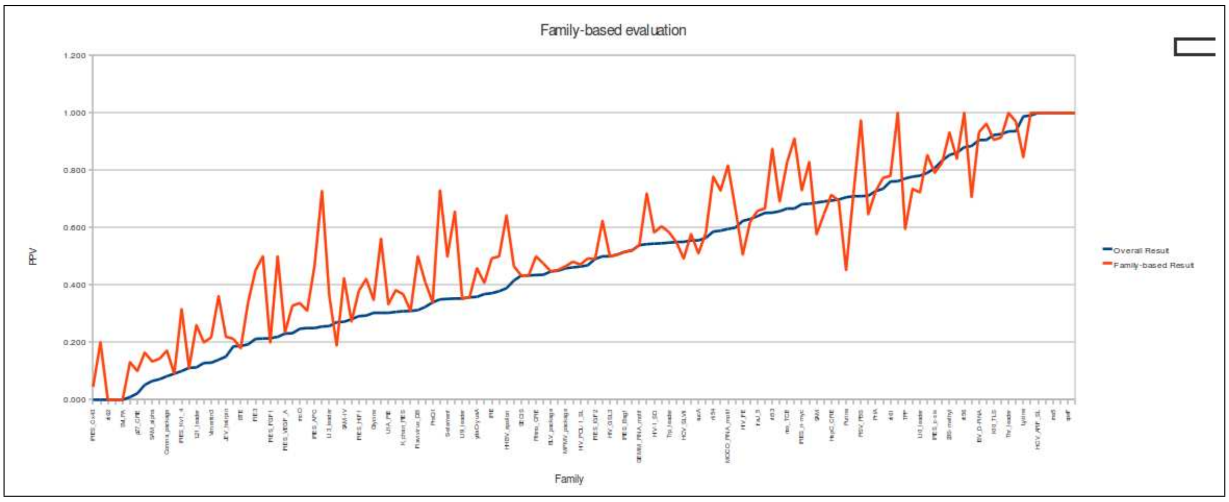


Figure 5.3: A graph visualizing the individual PPV value for each family after performing the family-based parameter tuning - red color - together with the corresponding individual PPV value after the overall parameter tuning - blue color - for `RNAMotid`.

5.2 Genome-wide evaluation

`RNAMotid` was designed with the requirement to include the ability to perform processings of input RNA sequences in a genome-wide scale. Thanks to windowing, `RNAMotid` was able to scan the 22nd human chromosome for *structured regions* - more than 50 million bases. Recall a region is continuous interval in the input sequence containing at least a part of an output RNA element. Approximately 2 million bases are processed per hour using a single cluster core (2 Mbases/hour). Thanks to the fast sparse algorithm. This time includes the time taken to calculate the base-pairing and the base-unpairing probabilities. RNAMotid was able to identify regions that are overlapping with known genome transcripts. The results are depicted in the graph in figure 5.4.

Because of the capability of RNAMotid to process genome-wide sequences, it has many applications. One of its applications is to narrow down the scope of input RNA sequences to those regions that contain only structured regions. These regions are used by alignment algorithms. By performing this narrow down we can reduce the overall running time. This idea is used when processing input RNA sequences for long ncRNAs. A sample of this usage is depicted in figure 5.5.

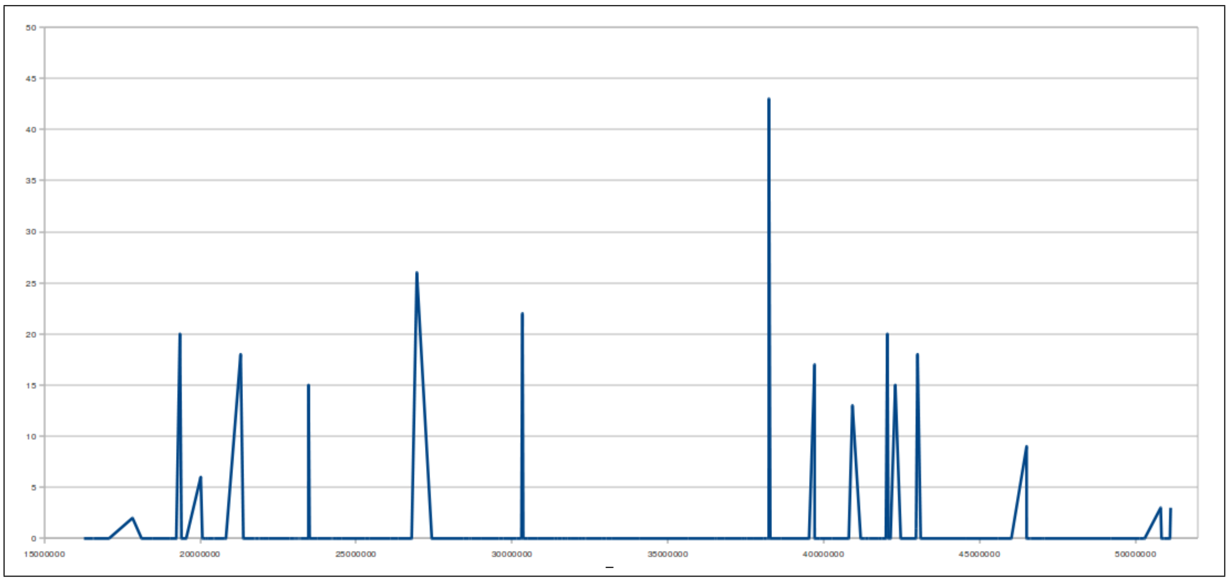


Figure 5.4: A graph depicting the results of processing the 22^{nd} chromosome of the human genome with RNAMotid. The x-axis represents the position in the sequence of the chromosome. The y-axis represents the number of regions of RNA elements discovered by RNAMotid that are overlapping with a genome transcript.

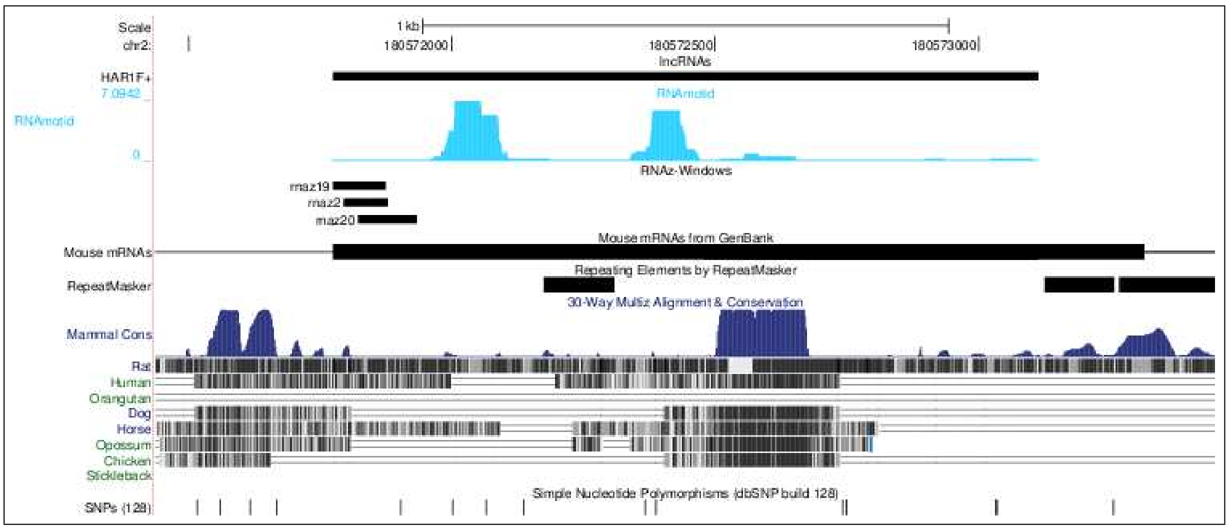


Figure 5.5: The usage of `RNAMotid`'s ability of identifying RNA structured *regions* in long ncRNAs. The bright blue track displays the output of `RNAMotid`

Chapter 6

Conclusion

In this thesis we introduced the new program `RNAMotid`. In conclusion, we employed the potentials of `RNAplfold` program in the computation of base-pairing and base-unpairing probabilities. This was combined with a fast sparse algorithm that is able to efficiently detect local structured RNA elements like cis-regulatory elements. Moreover, we introduced a new definition for locality through the presented accuracy function.

Also, a windowing mechanism was implemented in our program. It enables `RNAMotid` to scan long RNA sequences and operate in a genome-wide scale in a memory-efficient way.

Future work may target improving the performance of `RNAMotid` when dealing with genome-wide sequences. Also, additional investigations are still required concerning the structured regions.

In addition to performance related future work, other extensions are still available. Some of them may target the structures that `RNAMotid` is able to detect. Structures containing pseudoknots can be detected by `RNAMotid` by adding the required modifications in the sequence folding module.

Also, the modularity of `RNAMotid` enables the usage of individual modules in other aspects. For example, the window scanner can be easily used to include a window scanning mechanism in other programs.

Many features representing several modules were integrated together in our program. This modularity counts for the flexibility and optionality of the analysis. They also enables the reusability and also the extendability for future work.

Appendix A

Tables

Table A.1: `RNAMotid` parameters used in the performance evaluation

Parameter	**Value**
Maximum motif length	150
Minimum motif length	20
RNAplfold window size	160
RNAplfold span	80
RNAplfold cutoff	0
Scan window size	5000
Suboptimum threshold	0.95
Suboptimum type	global
Trace type	non-overlap-branch

Table A.2: Overall Evaluation Results: Cis-reg

Family	Count	Ind Trues	Ind Falses	Fin Trues	Fin Falses	Avg Ex	Avg Fin	Ind. PPV	Fin PPV
speF	6	95	0	6	0	0.3151	0.3151	1.000	1.00
K10_TLS	4	36	3	4	0	0.1783	0.2940	0.923	1.00
ylbH	3	39	0	3	0	0.2452	0.2938	1.000	1.00
HCV_ARF_SL	10	43	0	10	0	0.3054	0.3263	1.000	1.00
bicoid_3	10	113	1	10	0	0.2955	0.2961	0.991	1.00
rne5	3	19	0	3	0	0.2539	0.2644	1.000	1.00
RRE	10	66	0	10	0	0.3710	0.3710	1.000	1.00
yybP-ykoY	10	70	20	9	1	0.2247	0.3050	0.778	0.90
23S-methyl	10	64	11	9	1	0.2835	0.3362	0.853	0.90
PyrR	10	69	9	9	1	0.2537	0.3104	0.885	0.90
IBV_D-RNA	10	86	9	9	1	0.1666	0.2924	0.905	0.90
ykoK	10	74	32	8	2	0.2459	0.2743	0.698	0.80
Tombus_IRE	10	66	33	7	3	0.1980	0.3061	0.667	0.70
HCV_SLIV	10	96	30	7	3	0.1574	0.2700	0.762	0.70
T-box	10	69	28	7	3	0.2518	0.2716	0.711	0.70
RtT	10	69	36	7	3	0.1905	0.2515	0.657	0.70
RSV_PBS	9	66	27	6	3	0.2469	0.3218	0.710	0.67
HBV	3	12	8	2	1	0.1987	0.3056	0.600	0.67
rli51	8	30	25	5	3	0.1308	0.2284	0.545	0.63
purD	10	51	43	6	4	0.1882	0.2566	0.543	0.60
HCV_SLVII	10	60	49	6	4	0.1760	0.2786	0.550	0.60
nos_TCE	5	24	12	3	2	0.1561	0.2555	0.667	0.60
HIV-1_SD	10	37	31	6	4	0.0000	0.2645	0.544	0.60
HIV_POL-1_SL	10	39	45	6	4	0.1460	0.2158	0.464	0.60
CAESAR	8	30	34	4	4	0.1358	0.2645	0.469	0.50
sucA	10	55	44	5	5	0.1676	0.2796	0.556	0.50
Entero_CRE	10	50	46	5	5	0.1595	0.2605	0.521	0.50
HHBV_epsilon	6	21	33	3	3	0.1293	0.2548	0.389	0.50
Gammaretro_CES	10	59	83	5	5	0.2025	0.2790	0.415	0.50
mini-ykkC	10	48	56	5	5	0.0937	0.2626	0.462	0.50
6C	10	28	52	5	5	0.2104	0.3022	0.350	0.50
REN-SRE	4	22	9	2	2	0.0302	0.2396	0.710	0.50
RSV_RNA	8	45	36	4	4	0.2767	0.2954	0.556	0.50
MPMV_package	8	33	39	4	4	0.2040	0.2369	0.458	0.50

Table A.3: Overall Evaluation Results: Cis-reg

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
HepC_CRE	4	25	11	2	2	0.1105	0.2452	0.694	0.50
Flavivirus_DB	2	5	11	1	1	0.1499	0.2576	0.313	0.50
HIV_GSL3	10	50	50	4	6	0.1148	0.2210	0.500	0.40
HIV-1_SL4	10	36	44	4	6	0.0305	0.2240	0.450	0.40
serC	10	42	41	4	6	0.0875	0.2396	0.506	0.40
IRE	10	26	44	4	6	0.0486	0.2513	0.371	0.40
Retro_dr1	10	25	45	4	6	0.1801	0.2813	0.357	0.40
SECIS	10	29	38	4	6	0.1209	0.2660	0.433	0.40
U1A_PIE	10	20	46	4	6	0.0816	0.2484	0.303	0.40
Histone3	10	34	44	4	6	0.0647	0.2481	0.436	0.40
G-CSF_SLDE	10	56	39	4	6	0.2168	0.3180	0.589	0.40
traJ_5	5	32	18	2	3	0.2182	0.2827	0.640	0.40
Rhino_CRE	3	10	13	1	2	0.1448	0.2499	0.435	0.33
Gurken	3	6	11	1	2	0.2021	0.2941	0.353	0.33
GEMM_RNA_motif	10	69	59	3	7	0.1359	0.2437	0.539	0.30
ydaO-yuaA	10	33	59	3	7	0.1928	0.2649	0.359	0.30
Vimentin3	10	8	54	3	7	0.0686	0.2415	0.129	0.30
rncO	10	24	73	3	7	0.2090	0.2716	0.247	0.30
ykkC-yxkD	10	34	71	3	7	0.1710	0.2707	0.324	0.30
HLE	4	14	23	1	3	0.2116	0.2713	0.378	0.25
Cardiovirus_CRE	4	8	24	1	3	0.0679	0.2361	0.250	0.25
WLE3	9	28	48	2	7	0.1064	0.2350	0.368	0.22
Flavivirus_SLIV	9	22	53	2	7	0.1424	0.2462	0.293	0.22
FIE3	5	14	52	1	4	0.0659	0.2364	0.212	0.20
DPB	10	12	74	2	8	0.1195	0.2478	0.140	0.20
BLV_package	5	17	21	1	4	0.0455	0.2389	0.447	0.20
Corona_package	10	10	112	2	8	0.1634	0.2576	0.082	0.20
K_chan_RES	10	25	56	2	8	0.1536	0.2593	0.309	0.20
HIV-1_DIS	10	16	41	2	8	0.0553	0.2334	0.281	0.20
HIV-1_SL3	10	16	53	2	8	0.0501	0.2568	0.232	0.20
HepE_CRE	10	26	58	2	8	0.1597	0.2461	0.310	0.20
JEV_hairpin	10	17	96	2	8	0.1041	0.2637	0.150	0.20
S-element	10	45	83	2	8	0.1315	0.2792	0.352	0.20

Table A.4: Overall Evaluation Results: Cis-reg

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
ybhL	7	16	57	1	6	0.1074	0.2664	0.219	0.14
BTE	10	24	104	1	9	0.1290	0.2445	0.188	0.10
Tombus_3_III	10	16	70	1	9	0.1059	0.2716	0.186	0.10
SVLPA	1	0	6	0	1	0.0086	0.3259	0.000	0.00
p27_CRE	5	1	44	0	5	0.0535	0.2663	0.022	0.00
GAIT	3	2	16	0	3	0.0733	0.2820	0.111	0.00
Spi-1	3	0	15	0	3	0.1740	0.2548	0.000	0.00
HIV_PBS	10	1	108	0	10	0.1249	0.2460	0.009	0.00

Table A.5: Overall Evaluation Results: Cis-reg(Frameshift Element)

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
IS1222_FSE	4	47	21	3	1	0.1667	0.2850	0.691	0.75
HIV_FE	10	58	35	6	4	0.1312	0.2535	0.624	0.60
DnaX	5	3	39	0	5	0.1197	0.2572	0.071	0.00

Table A.6: Overall Evaluation Results: Cis-reg(IRES)

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
IRES_Picorna	10	77	8	9	1	0.2848	0.2938	0.906	0.90
IRES_Aptho	10	59	4	9	1	0.3129	0.3279	0.937	0.90
IRES_n-myc	4	15	7	3	1	0.2446	0.2748	0.682	0.75
IRES_c-sis	5	42	10	3	2	0.2688	0.2956	0.808	0.60
IRES_FGF2	5	43	23	3	2	0.2824	0.2888	0.652	0.60
IRES_Bip	4	12	35	2	2	0.2056	0.2568	0.255	0.50
IRES_Bag1	8	50	47	4	4	0.2592	0.2752	0.515	0.50
IRES_IGF2	6	27	28	3	3	0.1692	0.2652	0.491	0.50
IRES_KSHV	4	11	9	2	2	0.2043	0.2584	0.550	0.50
IRES_FGF1	2	3	11	1	1	0.1751	0.2521	0.214	0.50
IRES_HIF1	10	28	68	3	7	0.2395	0.2811	0.292	0.30
IRES_TrkB	10	22	92	3	7	0.2355	0.2841	0.193	0.30
IRES_Hsp70	8	30	68	2	6	0.1891	0.2723	0.306	0.25
IRES_APC	4	7	21	1	3	0.0484	0.2071	0.250	0.25
IRES_EBNA	5	20	46	1	4	0.1935	0.2423	0.303	0.20
IRES_Kv1_4	5	2	18	1	4	0.2229	0.2590	0.100	0.20
IRES_mnt	10	33	89	2	8	0.2066	0.2612	0.270	0.20
IRES_Tobamo	7	0	62	0	7	0.1380	0.2427	0.000	0.00
IRES_Cx32	4	5	50	0	4	0.1420	0.2053	0.091	0.00
IRES_VEGF_A	2	3	10	0	2	0.2678	0.3116	0.231	0.00
IRES_Cx43	3	0	35	0	3	0.1625	0.2677	0.000	0.00

Table A.7: Overall Evaluation Results: Cis-reg(Leader)

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
Thr_leader	10	87	6	9	1	0.2561	0.2986	0.935	0.90
L20_leader	10	87	23	9	1	0.2632	0.3064	0.791	0.90
His_leader	10	62	10	8	2	0.2577	0.3118	0.861	0.80
Leu_leader	4	15	3	3	1	0.2567	0.3089	0.833	0.75
L10_leader	10	50	14	7	3	0.2452	0.2896	0.781	0.70
Trp_leader	10	46	38	6	4	0.1807	0.2594	0.548	0.60
L19_leader	10	42	77	4	6	0.0468	0.2350	0.353	0.40
L21_leader	10	8	63	2	8	0.1115	0.2672	0.113	0.20
L13_leader	10	19	55	2	8	0.1283	0.2387	0.257	0.20
S15	10	4	73	1	9	0.1624	0.2617	0.052	0.10

Table A.8: Overall Evaluation Results: Cis-reg(Riboswitch)

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
rli55	3	32	0	3	0	0.2989	0.2989	1.000	1.00
Lysine	10	80	1	10	0	0.2944	0.3062	0.988	1.00
rli56	6	59	8	5	1	0.2182	0.2760	0.881	0.83
TPP	10	81	24	8	2	0.1936	0.2720	0.771	0.80
Purine	10	60	25	8	2	0.1475	0.2423	0.706	0.80
glmS	10	89	32	8	2	0.2666	0.2832	0.736	0.80
rli61	4	35	11	3	1	0.2564	0.2987	0.761	0.75
MOCO_RNA_motif	10	53	36	6	4	0.2301	0.2931	0.596	0.60
SAM	10	66	30	6	4	0.2213	0.2847	0.688	0.60
rli53	5	30	16	3	2	0.2694	0.2885	0.652	0.60
FMN	10	46	27	6	4	0.1943	0.2709	0.630	0.60
preQ1-II	10	57	44	6	4	0.1870	0.2679	0.564	0.60
rli52	6	26	34	3	3	0.2095	0.2751	0.433	0.50
Cobalamin	10	48	48	5	5	0.2465	0.2726	0.500	0.50
rli54	5	17	12	2	3	0.2666	0.2857	0.586	0.40
SAM-IV	10	28	75	3	7	0.1938	0.2718	0.272	0.30
PreQ1	10	36	70	3	7	0.0526	0.2402	0.340	0.30
Glycine	10	30	69	2	8	0.1736	0.2657	0.303	0.20
SAH_riboswitch	10	16	59	2	8	0.1296	0.2633	0.213	0.20
SAM_alpha	10	6	86	1	9	0.1021	0.2607	0.065	0.10
rli62	2	0	6	0	2	0.2321	0.3277	0.000	0.00
Mg_sensor	4	5	34	0	4	0.1712	0.2602	0.128	0.00

Table A.9: Overall Evaluation Results: Cis-reg(Thermoregulator)

Family	**Count**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Avg Ex**	**Avg Fin**	**Ind. PPV**	**Fin PPV**
Hsp90_CRE	4	25	2	3	1	0.1953	0.2390	0.926	0.75
PrfA	4	24	9	3	1	0.1830	0.2303	0.727	0.75
ROSE	10	82	38	6	4	0.2427	0.3008	0.683	0.60

Category (Sorted by families count)	Category-relative PPV (Trues count/Total count)	True Families Count	Total Families Count
Cis-reg	0.44	33	75
Cis-reg(Riboswitch)	0.64	14	22
Cis-reg(IRES)	0.33	7	21
Cis-reg(Leader)	0.60	6	10
Cis-reg(Thermoregulator)	1.00	3	3
Cis-reg(Frameshift element)	0.67	2	3

Table A.10: Overall performance summary for each category

Table A.11: Family Specific Evaluation Results: Cis-reg

Family	**Seed**	**Deg**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Ind PPV**	**Fin PPV**	**Avg Ex**	**Avg Fin**
RSV_PBS	15	0.95	37	1	9	0	0.9737	1.0000	0.319	0.317
HCV_SLIV	10	0.90	46	0	15	0	1.0000	1.0000	0.366	0.389
speF	10	0.50	89	0	6	0	1.0000	1.0000	1.701	1.701
K10_TLS	20	0.85	29	3	4	0	0.9063	1.0000	0.340	0.400
ylbH	10	0.50	38	0	3	0	1.0000	1.0000	1.299	1.586
HCV_ARF_SL	10	0.50	131	3	18	0	0.9776	1.0000	1.584	1.691
bicoid_3	10	0.50	110	0	10	0	1.0000	1.0000	1.595	1.598
rne5	10	0.50	24	0	3	0	1.0000	1.0000	1.367	1.424
RRE	10	0.50	152	0	30	0	1.0000	1.0000	2.014	2.014
23S-methyl	10	0.70	136	10	16	1	0.9315	0.9412	0.809	0.886
IBV_D-RNA	10	0.50	83	6	9	1	0.9326	0.9000	0.858	1.576
Tombus_IRE	10	0.95	61	6	10	2	0.9104	0.8333	0.274	0.314
6C	10	0.95	70	26	16	4	0.7292	0.8000	0.306	0.323
nos_TCE	10	0.95	24	5	4	1	0.8276	0.8000	0.239	0.269
yybP-ykoY	40	0.80	147	53	19	5	0.7350	0.7917	0.291	0.375
ykoK	55	0.90	197	87	23	7	0.6937	0.7667	0.204	0.228
PyrR	10	0.85	174	72	21	9	0.7073	0.7000	0.389	0.456
RtT	10	0.50	195	87	21	9	0.6915	0.7000	1.038	1.340
HBV	25	0.95	8	4	2	1	0.6667	0.6667	0.244	0.267
Gurken	15	0.85	19	10	2	1	0.6552	0.6667	0.371	0.415
HIV-1_SD	70	0.80	70	50	9	5	0.5833	0.6429	0.000	0.248
T-box	60	0.80	196	107	19	11	0.6469	0.6333	0.247	0.279
G-CSF_SLDE	25	0.95	89	33	8	5	0.7295	0.6154	0.234	0.274
IRES_c-sis	10	0.70	38	10	3	2	0.7917	0.6000	0.745	0.820
traJ_5	40	0.80	27	14	3	2	0.6585	0.6000	0.284	0.358
sucA	25	0.95	147	141	16	14	0.5104	0.5333	0.192	0.245
Entero_CRE	15	0.90	79	77	15	15	0.5064	0.5000	0.274	0.310
rli51	10	0.75	26	17	4	4	0.6047	0.5000	0.322	0.529
HHBV_epsilon	10	0.95	9	5	3	3	0.6429	0.5000	0.207	0.271
REN-SRE	10	0.60	23	9	2	2	0.7188	0.5000	0.134	0.952
RSV_RNA	70	0.85	52	38	4	4	0.5778	0.5000	0.236	0.252
HepC_CRE	10	0.95	5	2	2	2	0.7143	0.5000	0.188	0.257
Flavivirus_DB	10	0.95	1	1	1	1	0.5000	0.5000	0.225	0.271

Table A.12: Family Specific Evaluation Results: Cis-reg(Frameshift Element)

Family	**Seed**	**Deg**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Ind PPV**	**Fin PPV**	**Avg Ex**	**Avg Fin**
IS1222_FSE	20	0.90	22	12	2	2	0.6471	0.5000	0.213	0.323
HIV_FE	10	0.60	121	118	14	16	0.5063	0.4667	0.574	1.015
DnaX	10	0.95	2	12	1	4	0.1429	0.2000	0.195	0.279

Table A.13: Family Specific Evaluation Results: Cis-reg(Riboswitch)

Family	**Seed**	**Deg**	**Ind Trues**	**Ind Falses**	**Fin Trues**	**Fin Falses**	**Ind PPV**	**Fin PPV**	**Avg Ex**	**Avg Fin**
rli56	10	0.95	11	0	6	0	1.0000	1.0000	0.296	0.348
L20_leader	10	0.85	191	33	27	3	0.8527	0.9000	0.437	0.471
Lysine	10	0.70	231	42	27	3	0.8462	0.9000	0.771	0.815
MOCO_RNA_motif	20	0.85	218	49	24	6	0.8165	0.8000	0.358	0.397
rli53	10	0.85	14	2	4	1	0.8750	0.8000	0.427	0.448
glmS	20	0.90	116	34	13	4	0.7733	0.7647	0.306	0.324
rli52	10	0.75	16	33	4	2	0.3265	0.6667	0.558	0.644
TPP	45	0.90	146	99	20	10	0.5959	0.6667	0.180	0.240
IRES_FGF2	35	0.80	46	23	3	2	0.6667	0.6000	0.377	0.386
preQ1-II	10	0.80	82	59	8	6	0.5816	0.5714	0.386	0.511
SAM	10	0.50	142	104	17	13	0.5772	0.5667	1.010	1.480
Cobalamin	10	0.75	207	125	16	14	0.6235	0.5333	0.567	0.644
HLE	10	0.90	10	10	2	2	0.5000	0.5000	0.285	0.338

List of Tables

Bibliography

[1] Zubiaga AM, Belasco JG, and Greenberg ME. The nonamer UUAUUUAUU is the key AU-rich sequence motif that mediates mRNA degradation. *Mol Cell Biol.*, 15(4):2219–30, April 1995.

[2] Uzilov AV, Keegan JM, and Mathews DH. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics*, pages 7–173, March 2006.

[3] Hans-Joachim Böckenhauer and Dirk Bongartz. *Algorithmic Aspects of Bioinformatics (Natural Computing Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[4] Flamm C, Gardner P, Gautheret D, Giegerich R, Griffiths-Jones S, Hofacker I, Mathews D, Meyer I, Nieselt K, Stadler P, Steger G, Westhof E, and Zavolan M. *Theory and Practice of Computational RNA Biology.* EMBO workshop, 2008.

[5] Flamm C, Fontana W, Hofacker IL, and Schuster P. RNA folding at elementary step resolution. *RNA*, 6:325–38, 2000.

[6] Llave C, Xie Zhixin, Kasschau KD, and Carrington JC. Cleavage of *Scarecrow-like* mRNA Targets Directed by a Class of *Arabidopsis* miRNA. *Science*, 2002.

[7] Do CB, Woods DA, and Batzoglou S. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–8, July 2006.

[8] Athanasius F Bompfnewerer Consortium, Backofen R, Bernhart SH, Flamm C, Fried C, Fritzsch G, Hackermller J, Hertel J, Hofacker IL, Missal K, Mosig A, Prohaska SJ, Rose D, Stadler PF, Tanzer A, Washietl S, and Will S. RNAs everywhere: genome-wide annotation of structured RNAs. *J Exp Zool B Mol Dev Evol.*, 308(1):1–25, January 2007.

[9] Mathews DH. Predicting a set of minimal free energy RNA secondary structures common to two sequences. *Bioinformatics*, 21:2246–2253, May 2005.

[10] Mathews DH and Turner DH. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J Mol Biol*, 317(2):191–203, March 2002.

[11] Goodwin E, Okkema P, Evans TC, and Kimble J. Translational regulation of tra-2 by its $3'$ untranslated region controls sexual identity in C. elegans. *Cell*, 75(2):329–39, October 1993.

[12] Rivas E and Eddy SR. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16(7):583–605, July 2000.

[13] Rivas E and Eddy SR. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):2–8, 2001.

[14] Nawrocki EP, Kolbe DL, and Eddy SR. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–7, 2009.

[15] Tang G, Reinhart BJ, Bartel DP, and Zamore PD. A biochemical framework for RNA silencing in plants. *Genes Dev.*, 2003.

[16] Kiryu H, Kin T, and Asai K. Robust prediction of consensus secondary structures using averaged base pairing probability matrices. *Bioinformatics*, 23(4):434–41, 2007.

[17] Hofacker I. Vienna rna secondary structure server. *Nucleic Acids Res.*, 31(13):3429–31, July 2003.

[18] Hofacker I, Priwitzer B, and Stadler P. Prediction of Locally Stable RNA Secondary Structures for Genome-Wide Surveys. *Bioinformatics*, 20:186–90, 2004.

[19] Cousin J. Breakthrough of the year: Small RNAs make big splash. *Science*, 298:2296–97, December 2002.

[20] Ross J. mRNA stability in mammalian cells. *Microbiol.*, 59(3):423–50, September 1995.

[21] Havgaard JH, Torarinsson E, and Gorodkin J. Fast Pairwise Structural RNA Alignments by Pruning of the Dynamical Programming Matrix. *PLOS computational biology*, 3:e173, 2007.

[22] Havgaard JH, Lyngs RB, and Gorodkin J. The FOLDALIGN web server for pairwise structural RNA alignment and mutual motif search. *Nucl. Acids Res.*, 33:W650–3, 2005.

[23] Pedersen JS, Bejerano G, Siepel A, Rosenbloom K, Lindblad-Toh K, Lander ES, Kent J, Miller W, and Haussler D. Identification and Classification of Conserved RNA Secondary Structures in the Human Genome. *PLoS Computat Biol*, 2(4):e33, April 2006.

[24] Shu-yun L, Chen JH, Currey K, and Maizel J. A program for predicting significant RNA secondary structures. *Comput. Appl. Biosci.*, 4:153–59, 1988.

[25] Höchsmann M. *The Tree Alignment Model: Algorithms, Implementations and Applications for the Analysis of RNA Secondary Structures.* PhD thesis, Bielefeld, 2005.

[26] Höchsmann M, Voss B, and Giegerich R. Pure Multiple RNA Secondary Structure Alignments: A Progressive Profile Approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):53–62, 2004.

[27] Höchsmann M, Tller T, Giegerich R, and Kurtz S. Local Similarity in RNA Secondary Structures. *Proceedings of the IEEE Bioinformatics Conference*, pages 159–68, 2003.

[28] Welsh M, Scherberg N, Gilmore R, and Steiner DF. Translational control of insulin biosynthesis. *Biochemical Journal*, 235:459–467, 1986.

[29] Zuker M and Stiegler P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucl. Acids. Res.*, 9:133–48, 1981.

[30] Nussinov R, Pieczenik G, Griggs JR, and Kleitman DJ. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35:68–82, 1978.

[31] Klein RJ and Eddy SR. RSEARCH: Finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44, 2003.

[32] Bernhart S, Hofacker I, and Stadler P. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–615, 2006.

[33] Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy SR, and Bateman A. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res*, 33:D121–4, January 2005.

[34] Heyne S. Pairwise Comparison of RNA Secondary Structures via Exact Pattern Matches. Master's thesis, Friedrich-Schiller-Universität Jena, November 2007.

[35] Washietl S, Hofacker IL, and Stadler PF. Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci*, 102(7):2454–59, January 2005.

[36] Eddy SR and Durbin R. RNA sequence analysis using covariance models. *Nucleic Acids Res.*, 22(11):2079–88, June 1994.

[37] Wexler Y, Zilberstein C, and Ziv-Ukelson M. A study of accessible motifs and RNA folding complexity. *J Comput Biol.*, 14(6):856–72, July-August 2007.

[38] Yao Z, Weinberg Z, and Ruzzo WL. CMfindera covariance model based RNA motif finding algorithm. *Bioinformatics*, 22:445–452, February 2006.