Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat) der Technischen Fakultät der Universität Bielefeld

Advanced Tools for RNA Secondary Structure Analysis

vorgelegt von

Björn Voß

Bielefeld im November 2004

"Biologists should not deceive themselves with the thought that some new class of biological molecules, of comparable importance to proteins, remains to be discovered. This seems highly unlikely."
(Francis Crick, 1958)

Gedruckt auf alterungsbeständigem Papier nach ISO 9706.

Acknowledgements

In the following I want to thank persons who supported me either during the practical part of my work or during the preparation of this thesis or both.

Prof. Dr. Robert Giegerich who gave me the possibility to prepare this thesis, supervised me and assisted me with fruitful discussions and helpful tips throughout the whole time of my PhD project.

Dr. Marc Rehmsmeier for being my office colleague, answering my questions and helpful discussions on scientific as well as non-scientific manners.

The International NRW Graduate School in Bioinformatics and Genome Research, represented by PD Dr. Klaus Prank, for funding and attending my project.

Matthias Hoechsmann, Carsten Meyer, Peter Steffen, Michael Beckstette and Alexander Szcyrba who helped me with practical support and for nice times out of university.

All people of the AG Praktische Informatik for making my work a pleasant time.

My family, friends and especially my wife who supported me and had to endure my moods.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The analysis of RNA secondary structure has become more and more important throughout the last decades after it was recognised that RNA does not only serve as a passive messenger (mRNA), but also as a functional compound of the cell. Furthermore, it was elucidated that mainly the structure rather than the sequence determines the function of such non-protein-coding RNA. This means that two RNA molecules which have low sequence similarity but high structure similarity are likely to have a similar function.

The prediction of RNA secondary structure is based on parameters that have been measured *in vitro*. This results in rather static parameters, that do not incorporate the dynamic change of environment occurring in living organisms. Nevertheless, the use of these parameters, that are summarised in the energy model, gave valuable results, especially for short sequences. Several refinements throughout the years improved the predictions, but still the calculated optimal structure is not guaranteed to correspond to the native one. In this case, and due to the fact that the native structure is feasible under the energy model, it is common practice to additionally calculate suboptimal structures and incorporate these in the study. The set of all suboptimal structures is referred to as the structure space, which actually holds the information needed to answer questions such as: Is the optimal structure also the native one? Are there more than one structure an RNA molecule can adopt? How well-defined is the optimal structure?

Major problems in the analysis of the structure space are its size and its shape. The number of suboptimal structures is exponential in the sequence length, which means that for sequences of moderate length the size quickly exceeds several billion. Besides the size, the appearance of the structure space complicates its study. The structure space can be imagined as a rough landscape with valleys, holding local optimal structures, separated by mountains and saddles. This landscape is not smooth but cliffy and complex (see Figure 1.1), which prevents the development of a practical and still intuitive visualisation. In general, the intention of structure space analysis is not its visualisation, but its complexity also hampers approaches to derive specific features hidden in the structure space. Despite these problems, several tools exist that analyse the complete structure space or at least a part of it to answer the aforementioned questions. Among these are `MFOLD` which produces a subset of all possible structures according to a threshold of

Figure 1.1: Badlands as an example for a rugged landscape. This picture should give an impression on the complexity of the structure space of RNA, which is even more complex.

structural similarity, `SFOLD` which samples the structures in a probabilistic fashion and provides a method to identify alternating structures, `RNAsubopt` to produce all suboptimal structures within a given energy threshold, `barriers` to identify valleys, mountains and saddles of the structure landscape, and others.

My contribution to this area of research is twofold: First, I present `paRNAss` (prediction of alternating RNA secondary structures) which focuses on the detection of conformational switches and analyses the structure space based on pairwise comparisons. `paRNAss` has been available since 1997 and I could improve its predictive power as well as its speed which made possible a systematic evaluation. During this evaluation it turned out that `paRNAss` can even be used to identify more than two competing structures and hence get a deeper insight into the structure space. The second tool I introduce is `RNAshapes` which facilitates different kinds of analyses. The algorithm makes use of abstract representations of the secondary structure to compute only those that are morphologically dissimilar, i.e. are composed of different structural elements. Structures being morphologically similar are pooled in a class of structures and each class is represented by its best member. The list of these representatives gives a general overview of what is there in the structure space. In addition to this, I introduce an algorithm to compute probabilities of the aforementioned classes of structures. This gives hints to properties such as alternating secondary structures (two classes with similar probabilities) and structural well-definedness (one class with very high probability).

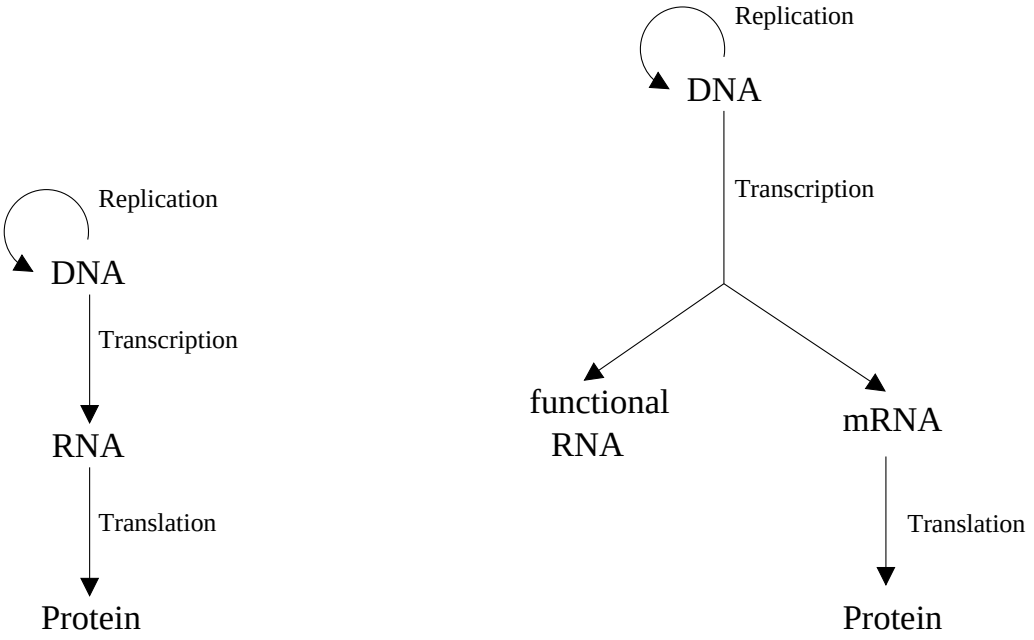
CHAPTER 2

RNA - Biology, Chemistry and Theory

2.1 RNA and its Vital Functions

Early experiments in molecular biology revealed, that DNA (**D**eoxyribo**N**ucleic **A**cid) is the carrier of the genetic information whereas proteins are the catalytic active compounds of the cell. For RNA (**R**ibo**N**ucleic **A**cid), the only function left was that as a passive transport system of genetic information from DNA to the protein factories of the cell, the ribosomes. This view of the concerted operation of DNA, RNA and proteins is known as the "Central Dogma" of molecular biology and depicted in Figure 2.1(a). In 1982 the discovery of a catalytic RNA with a complex chemical reactivity, as it was only known for proteins until then, changed the view on RNA: The ribosomal RNA (rRNA) molecules of the ciliate *Tetrahymena* are first synthesised as one large precursor which is subsequently cleaved to the mature rRNAs. Surprisingly, these cleavage reactions also take place in absence of any protein. Later it was shown that the intron sequence itself carries an enzyme-like catalytic activity [1]. The last decades revealed more and more functions of RNA molecules besides serving as a blueprint for proteinbiosynthesis. Today these non-coding RNAs (ncRNAs, because they do not encode a protein) are known to possess regulatory functions, like the recently discovered micro RNAs (miRNAs) [2], catalytic functions, like the aforementioned self-splicing introns and like small nuclear (sn)RNAs in the spliceosome. Additionally, they may serve as a scaffold, such as rRNAs do in ribosomes and in Retroviruses RNA is actually the carrier of the genetic information. All these facts led to the idea of an early "RNA World", where RNAs were the central molecules for information storage, catalysis and regulation. In the course of time RNA lost its predominance, as DNA, a more stable storage possibility, and proteins, as catalysts with higher versatility, emerged. This does not mean, that RNA lost all its functions, but it was eclipsed by DNA and proteins.

In addition, according to the "Central Dogma" all diversity of the species must be due to differences in the repertoire of genes. Contradictory to this, the genome sequencing projects revealed that the overall number of protein coding genes is much smaller than expected and that the protein coding fraction of the genome is only about 5%. Additionally, there seems to be no clear correspondence between the complexity of a



(a) The "Central Dogma" (b) The "Modern Central Dogma"

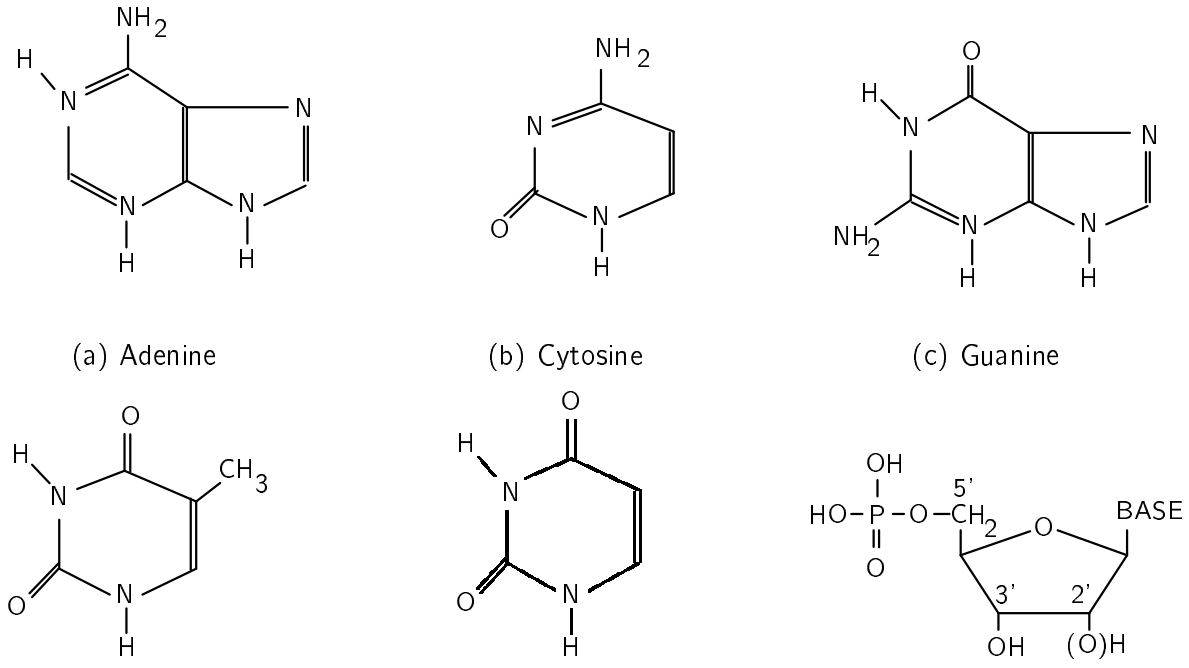
Figure 2.1: The existing dogmas in molecular biology. RNA was originally proposed to serve as a passive messenger only, but experiments revealed that RNA is also part of the functional and regulatory compounds of the cell.

species and the number of genes in its genome. Fruit flies have fewer coding genes than roundworms, and rice plants have more than humans. This put forth the question, if the remaining 95% of the DNA are just "junk" or if there is vital information in it. Analyses showed that parts of this non-genic DNA are well conserved across different species and that loss of these regions leads to specific phenotypes, such as cartilage hair hypoplasia (CHH) [3]. Altogether, this results in a "Modern Central Dogma", where RNA is also part of the functional and regulatory cell compounds, like shown in Figure 2.1(b).

2.2 Chemical Structure of RNA

RNA is a polymer of the nucleotides Adenylate (A), Guanylate (G), Cytidylate (C) and Uridylate (U) which are connected via phosphodiester bonds between the 5'-phosphate- and the 3'-OH-group of the sugar-phosphate part of adjacent nucleotides, see Figure 2.2. This is similar to the chemical structure of DNA where the 2'-OH of the sugar is deoxygenated to 2'-H and the nucleotide Uridylate is replaced by Thymidylate (T).

DNA is known to occur solely double stranded, where A–T and G–C pair by forming two resp. three hydrogen bonds and thereby leading to the characteristic double helical structure of DNA. In contrast to this, RNA mainly occurs single stranded. The nucleotides of an RNA molecule as well have the possibility to form hydrogen bonds. This time they are not formed between two strands but between nucleotides of the same



(a) Adenine (b) Cytosine (c) Guanine

(d) Thymine (e) Uracile (f) Sugar-phosphate backbone

Figure 2.2: The bases and the sugar-phosphate backbone of RNA and DNA.

strand. This forces an RNA molecule to fold back onto itself, forming a characteristic structure which is unique for each RNA. Hence, RNAs do not only vary in sequence, like DNA, but also in structure, which opens up another dimension of flexibility. In general, RNA has four structural levels, resembling different levels of information encoding:

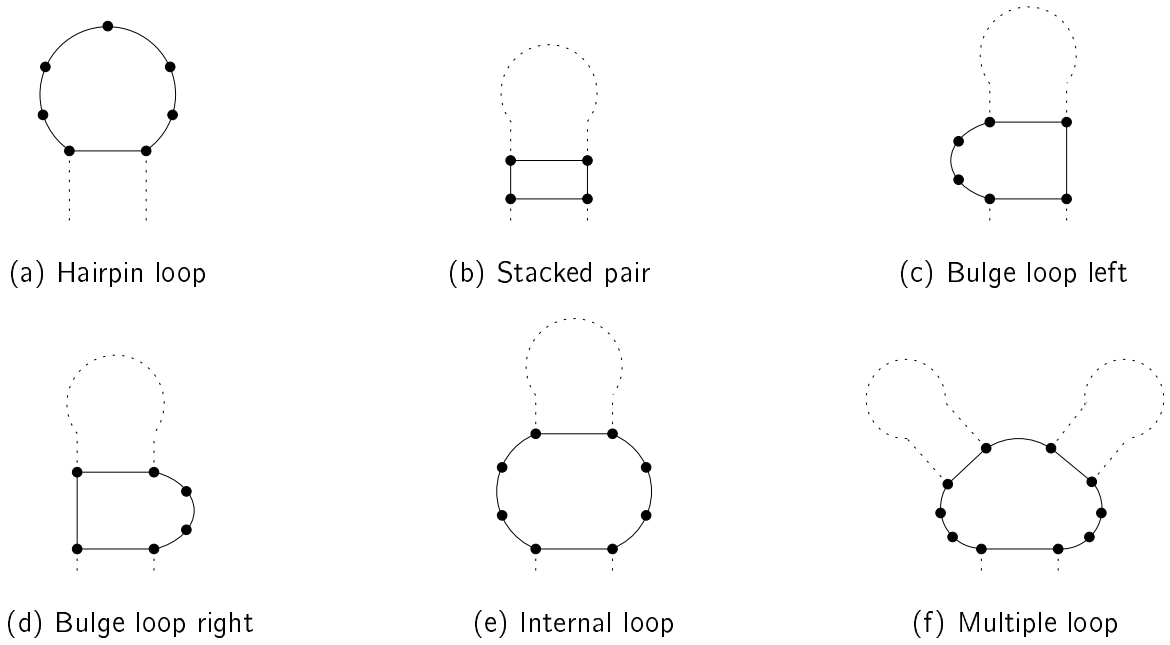
Primary structure: The plain sequence of nucleotides, which is in the case of mRNA sufficient to determine the sequence of the encoded protein.

Secondary structure: Formation of hydrogen bonds between A–U, G–C and G–U leads to formation of helical (paired) and intervening singlestranded (unpaired) regions. This definition is restricted to nested base pairs (no pseudoknots), meaning that each two base pairs (i, j) and (k, l) of a valid secondary structure have to satisfy the following constraint:

$$i < k < l < j \;\;||\;\; i < j < k < l \;\;||\;\; k < l < i < j \;\;||\;\; k < i < j < l \qquad (2.1)$$

For small regulatory motifs, the formation of the secondary structure might be sufficient for their correct function, but in most cases a certain tertiary structure is needed.

Tertiary structure: This is the spatial arrangement of all elements of the RNA in three dimensions. Like for proteins, this is the active structure of RNAs that act without co-factors, e.g. ribozymes like the self-splicing intron of *Tetrahymena*.



(a) Hairpin loop (b) Stacked pair (c) Bulge loop left

(d) Bulge loop right (e) Internal loop (f) Multiple loop

Figure 2.3: Elements of RNA secondary structure. All elements have a closing base pair and zero (hairpin loop), one (stacked pair, bulge loop, internal loop) or more (multiple loop) internal base pairs.

Quaternary structure: Agglomerate of RNAs, proteins and chemical co-factors, like small nuclear ribonucleoproteins (snRNPs) which consist of snRNAs and several proteins.

Elements of the primary structure are the nucleotides A, C, G, U and sometimes modified bases, such as pseudouridine or methylated bases.The secondary structure of each RNA molecule can be decomposed into a series of six distinct elements, namely stacked pair, hairpin loop, bulge loop left, bulge loop right, internal loop and multiple loop as shown in Figure 2.3. Tertiary and quaternary structure cannot be divided into unique elements as they are based on various chemical interactions of parts of the secondary resp. tertiary structure.

2.3 The Structure Space of RNA

The tertiary structure of an RNA, which is the biological active one, is very hard to determine, both in vitro and in silico. As the building blocks of the tertiary structure are given by the secondary structure and the latter is much easier to access, one commonly uses the secondary structure as an approximation of the tertiary structure. In vivo, the secondary structure is influenced by its environment (ion concentrations, hydrophobicity). These biophysical parameters differ between individual cells or cellular compartments, and hence the secondary structure may differ, too. These influences are generally assumed to be rather small compared to the thermodynamic forces involved in base pairing and

base pair stacking. For this reason the term structure space from now on refers to the thermodynamic secondary structure space, being defined as follows:

The structure space of a given RNA sequence is described by a thermodynamic model, given by experimentally determined energy parameters [4]. Base pairing and base pair stacking are structure stabilising factors associated with negative energy contributions depending on the nucleotides involved. Single-stranded parts (loops) have destabilising effects associated with positive energy contributions, mainly depending on the length of the single-stranded parts. Based on these energy contributions, a free energy value for each structure is calculated. The structure attaining minimal free energy is called *mfe-structure*. Since the energy parameters are determined in vitro under conditions which are near but not identical to in vivo conditions and biophysical influences are neglected, it is not guaranteed that the predicted mfe-structure corresponds to the native structure. As a consequence to this and due to the possibility of several important conformations, one is also interested in energetically near-optimal structures.

Two structures are considered as *neighbours* if they differ by just opening or closing one base pair. Neighbouring structures achieve a similar energy value. This leads to the presence of many similar structures when looking at near optimal structures. The structures one is really interested in are the *local energy minima*, having lowest free energy with respect to all their neighbours. Since the size of the structure space is exponential in the length of the sequence [5], it is very likely that a large number of such local energy minima are present in the structure space. Thinking of the structure space as a landscape, suggests the notion of a *valley* for a local minimum and all its neighbouring structures that can be reached by opening (or closing) base pairs and thereby always increasing the free energy. All structures that belong to one valley are called a *family* of structures. A structure having two neighbours from different valleys, corresponds to a saddle point in the landscape.

The shape of the structure space gives hints at structural properties of the RNA molecule. The existence of one valley being much "deeper" than all other valleys, implies structural well-definedness. Two equally deep valleys could show up for a conformational switch or, if the native structure is determined by a specific folding pathway, trapping the structure in a local minimum.

These definitions suggest a rather ordered appearance of the structure space. This holds true only for the near-optimal part of the structure space. Generally the structure space is very large (exponential in the sequence length, e.g. 25,986,090,120,790 structures for a sequence of length 30 nt) and complex. The complexity arises from the fact that each structure has at least (when considering only opening of base pairs) as many neighbours as its number of base pairs. To give an example: A structure with 20 base pairs has 20 direct neighbours with 19 base pairs. Each of them has 19 neighbours with 18 base pairs and so on. Bear in mind that neighbours to the second degree can be reached on two different ways, those of third degree on six ways and those of n_{th} degree on $n!$ ways.

2.4 Analysing the Structure Space – Current Methods

The prediction of functional properties of an RNA is the central goal in analysing the structure space. Due to the size and especially the complexity there is, until today, no general approach for analysing the structure space. However, some methods have been developed to scrutinise specific features. The following section gives an overview of current methods. More detailed descriptions of the algorithms will follow in Chapter 3.

2.4.1 Alternative Secondary Structures

The *mfe*-structure is, especially for long sequences, unlikely to be equal to the native in vivo structure. Hence, it is common practice to include suboptimal solutions in the process of structure elucidation. The first approach to compute also suboptimal solutions was implemented in `MFOLD` [6]. Due to the implementation, this approach is not capable of producing all suboptimal structures, but only those being dissimilar to other structures to some degree. The heuristic filters that are used for this lead to a biased sample which holds only structures that are more dissimilar than a certain threshold. Additionally, structures that are composed solely of base pairs that are also part of "better" structures never show up. The advantage of this approach is that the user has to handle only a small number of structures, but the danger of missing the one he is looking for remains.

An algorithm producing precisely all suboptimal structures was presented in [7] and is implemented in the tool `RNAsubopt` of the Vienna RNA package [8]. The problem arising is that the output gets very large (exponential size of the structure space). Analysis of such large numbers of structures is very costly, especially in the case of pairwise comparisons, as the number of comparisons grows quadratic with the size of the sample. Restricting to structures that are near-optimal is a reasonable approach, but poses the problem of defining near-optimality.

A (reasonable) reduction of the number of suboptimal structures is implemented in the tool `SFOLD` by Ding and Lawrence [9]. It uses a statistical sampling procedure to compute a subset of all suboptimal solutions. The authors state that each structure occurs in the sample according to its probability, given that the sample size is large enough. As the sampling is based on random variables each run of the program produces a different set of structures. Furthermore, the same structure can be sampled multiple times, especially the *mfe*-structure as it is the most probable one.

2.4.2 Structural Well-Definedness

In case of an RNA functioning through a certain structure, this structure has to be well-defined, meaning that there are no dissimilar structures of similar energy. The idea behind is, that functional (by means of structure) RNAs can be detected based on a measure for structural well-definedness.

The approaches to infer structural definition share the common idea of comparing the optimal structure to suboptimal structures based on two different aspects. One is the difference in energy and the other is structural dissimilarity. The method by Wuchty et al.

[7] uses the so called "base pair distance" to measure structural dissimilarity, whereas the approach by Kitagawa et al. [10] makes use of the so called "tree representation distance". Both groups could show the applicability of their approaches to certain problems or classes of RNAs (tRNAs and snRNAs respectively), but did not show the applicability in general, i.e. to classify unknown sequences as functional by means of structure.

2.4.3 Alternating Secondary Structures

The property of an RNA to be present in either one of two alternate conformations both having a different function is known as conformational switching. Ding and Lawrence [9] propose a method to elucidate such conformational switching based on their statistical sampling procedure for RNA secondary structure. The idea is that the existence of two alternating conformations is resembled in a two-valley shape of the structure space. For this reason, the drawn structure sample should hold two families of structures representing the two valleys.

Another approach is `paRNAss` [11], which is based on the pairwise comparison of suboptimal structures. It uses an estimate for the energy barrier and a distance measure to infer structural dissimilarity. This data is subsequently used for a clustering step which is the basis for the prediction of consensus structures. These consensus structures should resemble the two conformations of the switch and get verified by comparing each consensus structure to all sampled structures. If all of these have a high energy barrier to one consensus while having a low to the other, the consensus structures as well as the property of conformational switching is predicted. This method is the starting point for my work and is described in Chapter 4.

2.4.4 Analysing the Folding Pathway

The folding of RNA is a kinetic process, which starts with the formation of a first (random) base pair, the so called nucleation point. The addition and removal of base pairs leads to formation of intermediate structures that may correspond to local minima. In this case the RNA might get trapped in this state because it cannot overcome the surrounding (energy) barriers within reasonable time and, hence, is not able to achieve its energetic optimum. The state the RNA gets trapped in has to be a valid secondary structure and therefore it has to be present in the set of suboptimal structures. Analysis of the structure space can therefore be used to elucidate such kinetic traps of the folding pathway. Approaches addressing this problem are described in Section 3.4.2.

2.4.5 Local Minima

The appearance of the structure landscape is predominantly determined by the local minima and the saddle points separating them. Thus, to roughly describe the structure space it is sufficient to know these. The tool `barriers` [12] addresses this idea. It is designed to analyse the complete or part of the structure space, which is given as an energy sorted list of structures generated by `RNAsubopt`. From this `barriers` constructs

the barrier tree as a representation of the energy landscape. This barrier tree holds information on local minima, the saddle points connecting them and their differences in energy.

CHAPTER 3

Computational Tools and Methods

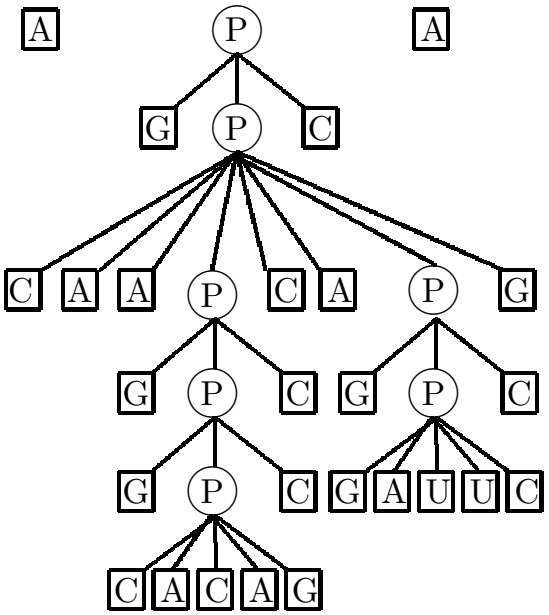
3.1 Representations of RNA Secondary Structure

During the (still ongoing) research on RNA secondary structure different descriptions resp. representations have emerged, all having their specific advantages and disadvantages. Two dimensional representations are intuitive to the human eye, whereas they are impractical as computer input. The opposite holds for one-dimensional or string representations. In the following paragraphs I will give an overview of existing representations, which does not claim to be complete, as a large number of variations or very specialised representations exist.

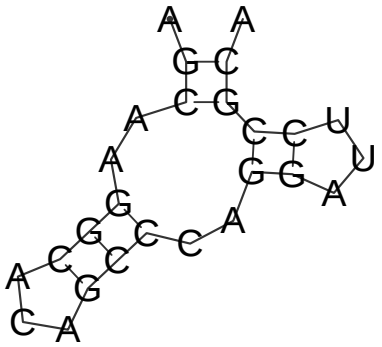
3.1.1 Two-Dimensional Visualisation

Tree/Forest Representation The secondary structure of RNA is inherently tree-like, so it is straightforward to use a tree representation. In fact, the secondary structure is represented as a forest, as each element (stem, unpaired base) of the external loop, such as the two unpaired 'A's in Figure 3.1(a), are trees themselves. Such a tree is build of P(air)-nodes for base pairs and leafs holding the individual bases, as shown in Figure 3.1(a). Note, that the left- and rightmost children of a P-node are actually forming the base pair. This representation is intuitive only to a small degree, but apart from serving as a visualisation it also resembles a data structure used by algorithms dealing with RNA secondary structure.

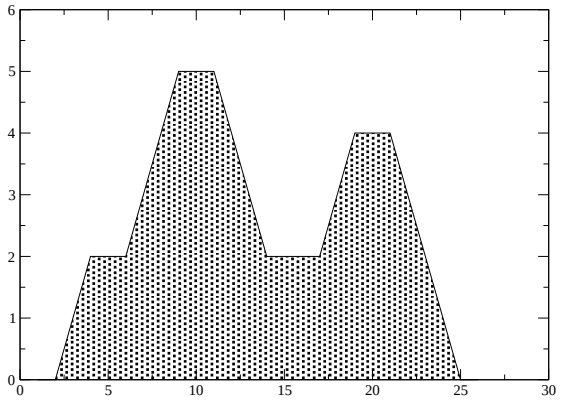
Squiggle Plot The so called *squiggle* plot is probably the most widely used representation, as it is also the most intuitive for the human reader. It gives a good impression about the two dimensional arrangement of the elements of the secondary structure (see Figure 3.1(b)). Several variations of the squiggle plot exist, which mainly differ in the information displayed and the treatment of bulge and asymmetric internal loops. Common to all is, that adjacent bases are connected by a line and bases forming a base pair are also connected by some element, e.g. line, rhomb or dot.



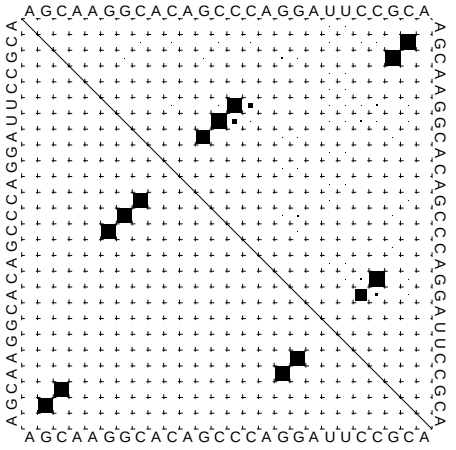
(a) Tree/Forest representation



(b) Squiggle plot



(c) Mountain Plot



(d) Dot plot

Figure 3.1: Two dimensional representations for RNA secondary structure.

(a)	Base pair list	[(2,25),(3,24),(6,14),(7,13),(8,12),(17,23),(18,22)]
(b)	"Vienna" string	`.((..(((...)))..((...)))).`
(c)	HIT structure	`(U)((U2)((U3)P3)(U2)((U3)P2)P2)`
(d)	Coarse grained structure	((H)(H)M) or ((((H)S)((H)S)M)S)
(e)	Weighted coarse grained	(((((H3)S3)((H2)S3)M2)S4)E2)
(f)	Abstract Shape	`[[][]]`

Figure 3.2: Exact and approximate string representations for RNA secondary structure.

Mountain Plot In order to display the structural constraints over the sequence, the *mountain* plot (Figure 3.1(c)) has been developed in [13]. Each position in the sequence that is the positional smaller partner of a base pair adds a positive contribution whereas a positional greater partner contributes negatively. This contribution may be either ± 1 or $\pm E$, where E is the free energy contribution for this base pair.

Dot Plot The previous representations where capable of depicting one structure at the time. A way to show all, or at least the most probable, structures a sequence can form is realised in the *dot* plot produced by `RNAfold` from the Vienna RNA package and shown in Figure 3.1(d). In general, it is a matrix that holds for each pair of bases the probability to form a base pair. This probability is depicted by a square, whose area corresponds to the actual probability. The resulting plot is split into an upper-right triangle holding all information and a lower-left triangle showing only those base pairs that are part of the most probable structure.

3.1.2 String Representation

The diverse two-dimensional representations focus on the ease-of-use for the human reader, while being inappropriate for further computational analysis. This lead to the creation of more computer-friendly notations of secondary structure, either in an exact or an approximate fashion.

Exact Representations

Base pair list: The base pair list holds for each base pair the indexes of the two participating bases, as shown in Figure 3.2.

"Vienna" string: The so called "Vienna" string or "Dot-Bracket" notation, represents the secondary structure as a sequence of dots and opening or closing brackets. The dots represent unpaired bases and pairs of brackets represent base pairs, see Figure 3.2.

HIT structure: This representation is based on 'U' for unpaired bases and 'P' for paired bases. The numbers following give the size of the individual elements, e.g. `((U3)P3)` is a hairpin consisting of 3 base pairs and 3 unpaired bases in the hairpin loop. An example is depicted in Figure 3.2.

Approximate Representations In [14] several approximate representations for RNA secondary structure have been proposed:

Coarse grained structure: The coarse grained structure notation abstracts from the individual sizes of the structural elements. Hairpins, interior loops, bulges, multi-loops, stacks and external bases are represented by (H), (I), (B), (M), (S), and (E), respectively. For the example it results in the representation shown in Figure 3.2.

Weighted coarse grained: This representation is derived from the coarse grained by adding specific weights (e.g. size) to each element, and shown in Figure 3.2.

In Chapter 5, I describe the implementation of a new method for RNA secondary structure prediction. It is based on the notion of *abstract shapes* or *shapes* for short. Shapes are classes of structures which share structural features, such as the nesting of hairpins. Each shape is described by a representative structure and a specific notation, the shapes notation.

Shapes notation: With the shapes notation it is possible to abstract from structural details and to get an overall impression of the RNA structure. In general, unpaired regions are represented by '_' and paired regions by pairs of '[' and ']'. The level of abstraction can be chosen at wish: It is possible to exclude any type of structural element as well as single strands from being represented. The most abstract version displays only hairpins and multiloops without regarding unpaired bases, as it is exemplified in Figure 3.2. A detailed description is given in Section 5.2.

3.2 Comparison of RNA Secondary Structures

Often, it is of interest to determine the relation of two or more RNA secondary structures. To infer this relation it is useful to know how similar or dissimilar the structures are. For this reason several measures of dissimilarity, i.e. distance measures have been developed, each having its specific advantages and disadvantages. Here I describe the ones that are used throughout the following chapters.

3.2.1 Base Pair Distance

The *base pair distance* of two structures is defined as the number of base pairs that are unique to one of the structures. For two structures s_1, s_2 with base pairs $x \in s_1$ and $y \in s_2$ the following applies:

$$d_{bp}(s_1, s_2) = \sum_{x \in s_1} \begin{cases} 1, & x \notin s_2 \\ 0, & x \in s_2 \end{cases} + \sum_{y \in s_2} \begin{cases} 1, & y \notin s_1 \\ 0, & y \in s_1 \end{cases} \tag{3.1}$$

This distance measure is provided with `RNAdistance` from the Vienna RNA package.

3.2.2 Morphological Distance

The *morphological distance* d_{MD} is a slightly modified version of Equation 3.6. Here structures are represented as sets of base pairs. $(i, j) \in s$ means that residues i and j form a base pair in s. For two structures s_1, s_2 the following applies:

$$d_{MD}(s_1, s_2) = \max \begin{cases} d'_{MD}(s_1, s_2) \\ d'_{MD}(s_2, s_1) \end{cases} \text{, where}$$
$$d'_{MD}(s_1, s_2) = \sum_{(i_1, j_1) \in s_1} \min_{(i_2, j_2) \in s_2} \max \begin{cases} |i_1 - i_2| \\ |j_1 - j_2| \end{cases} . \tag{3.2}$$

d_{MD} is strictly positive and symmetric, but does not satisfy the triangle inequality. Although it is not a metric in the mathematical sense, it behaves quite reasonably as a distance measure.

3.2.3 Tree Alignment Distance

The *tree alignment distance* d_{TAD} is based on the extended RNA forest representation introduced by Höchsmann et al. [15] and depicted in Figure 3.1(a). The nodes of trees representing RNA structures are either P(air)- or B(ase)-nodes. The following edit operations are applicable: base replacement, base deletion, pair replacement and pair deletion, scored by cost contributions b_r, b_d, p_r, p_d respectively. Given a distance function δ, with $\delta(B, B) = b_r, \delta(B, -) = b_d, \delta(P, P) = p_r, \delta(P, -) = p_d$ and a structure alignment A represented as a forest, the distance score of the alignment is defined as $\Delta(A) = \sum_{v \text{ node in } A} \delta(v)$. This leads to the tree alignment distance

$$d_{TAD}(s_1, s_2) = \min\{\Delta(A(s_1, s_2))\} \quad , \tag{3.3}$$

where $A(s_1, s_2)$ varies over all structure alignments of s_1 and s_2. This distance measure is incorporated in the tool `RNAforester`. Note that d_{TAD}, although based on a tree edit model, is different from the "tree edit distance" [16].

3.2.4 Energy Barrier Distance

Remembering the landscape representation of the secondary structure space, two valleys are separated by a mountain or saddle. A structure morphing from one valley to the other has to overcome this obstacle, meaning it has to gain free energy to climb up the mountain. Once the peak is reached the way down to the bottom of the other valley can be performed without effort. The energy needed for the ascent resembles the activation energy needed for the transition reaction between the two structures and is called *energy barrier*. An example is given in Figure 3.3.

The *energy barrier distance* d_{EB} was first introduced by Rehmsmeier [17]. It is designed

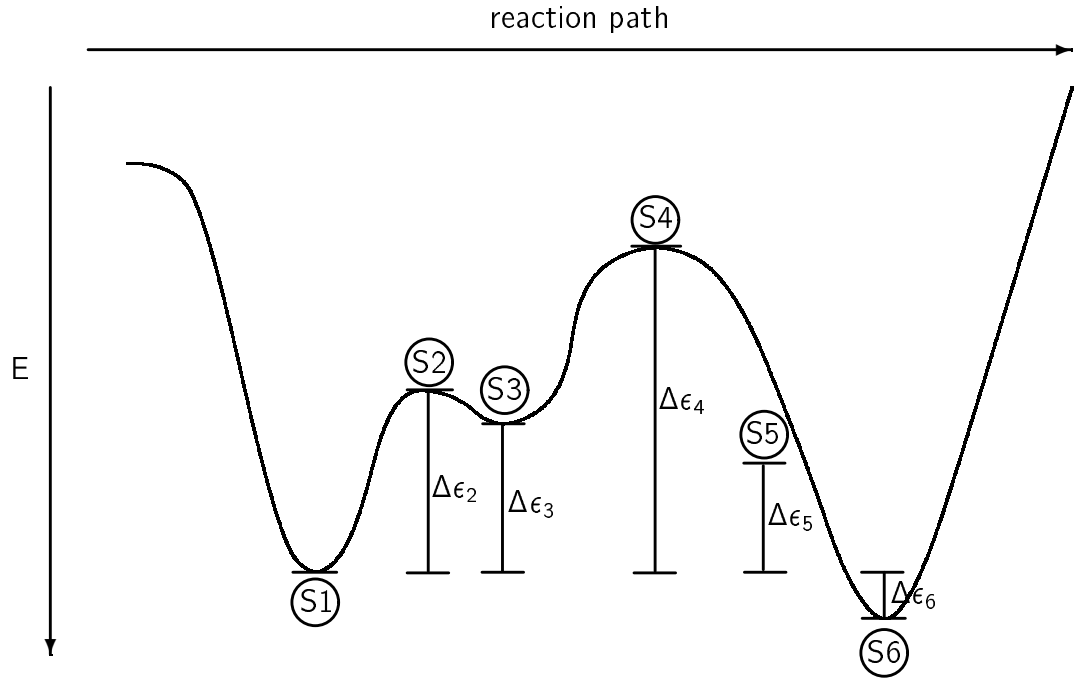


Figure 3.3: Example path from structure S1 to structure S6. The energy barrier between structures S1 and S6 is $\Delta\epsilon_4$, whereas between S6 and S1 it is $|\Delta\epsilon_4| + |\Delta\epsilon_6|$.

to capture the minimal amount of energy necessary for the molecule to shift between two structures. A transition path from s_1 to s_2 is given by a sequence of intermediate structures. Let $e(s)$ denote the free energy of s.

$$\begin{aligned} d_{EB}(s_1, s_2) = &\quad \min \begin{cases} d'_{EB}(s_1, s_2) \\ d'_{EB}(s_2, s_1) \end{cases} \quad \text{, where} \\ d'_{EB}(s_1, s_2) = &\quad \min\{e(p) \mid p \text{ is transition path from } s_1 \text{ to } s_2\} \quad \text{, where} \\ e(p) = &\quad \max\{e(s) - e(s_1) \mid s \text{ is intermediate structure in } p\} \end{aligned} \tag{3.4}$$

As the number of possible paths is excessively large, the implementation uses a greedy heuristic to approximate d_{EB}: A list is build which holds for each base pair that has to be closed, shortly *close*, the base pairs that have to be opened, shortly *opens*. The close with the least number of opens is chosen, its opens are performed and deleted from the open list of the remaining closes. Then the close is performed and eliminated from the list. This procedure is repeated until no close is left. Finally, possible opens not connected to a close are performed. The energy barrier distance satisfies the axioms of a metric. It is always non-negative, as the starting structure of a path is also in the set of intermediate structures.

In this implementation the heuristic allows to introduce or to leave behind lonely base

pairs. This is not favourable, as lonely base pairs mostly have positive energy contributions. The result is that lonely base pairs lead to higher free energies without good reason. Furthermore, the heuristic does not take care of neighbouring closes or opens, meaning that the formation of a five base pair stem might be performed as follows: First base pair 1 is closed, next nr.3, then 5, 2, 4, revealing an intermediate structure with two internal loops and, therefore, high free energy.

To cope with these drawbacks I refined the heuristic to allow only canonical structures and to also take care of sequential closes or opens. In more detail this means, that two closes and their opens are combined if they are neighbours and no other neighbour of the first close is already closed. In the case of opens it is checked whether the current open is followed by an open which has no neighbouring open to come. If this is the case these two opens are performed together.

3.2.5 Pseudoknot Measure

A pseudoknot is a structural element which is commonly regarded as part of the tertiary structure. The basic property of a pseudoknot is that its base pairs are not nested properly, but cross each other (i.e. they do not satisfy Equation 2.1). Standard RNA folding algorithms are not capable of predicting pseudoknots. Therefore, when comparing two different structures of one RNA sequence, they might represent the two possible unknotted projections of a pseudoknot structure, meaning that the union of the base pairs of these structures is one structure comprising a pseudoknot. To get an idea how likely two structures can combine in a pseudoknot, the *pseudoknot measure (pkMeasure)* is calculated as follows:

Let $s_1 \cup s_2$ denote the union of two structures (i.e. base pair sets). Define

$$pkMeasure(s_1, s_2) = \begin{cases} -1 & \text{, if } s_1 \cup s_2 \text{ is a valid structure without a pseudoknot,} \\ 0 & \text{, if } s_1 \cup s_2 \text{ contains a pseudoknot} \\ k & \text{, where } k \text{ is the number of bases with conflicting pairings in } s_1 \cup s_2. \end{cases} \tag{3.5}$$

The larger *pkMeasure*, the less likely it is that the two conformations actually can combine in a pseudoknot.

3.3 Cluster Analysis

The goal of cluster analysis is to divide a collection of objects into subsets or clusters, such that those objects within each cluster are more closely related to one another than objects assigned to different clusters. Central to cluster analysis is the notion of distance

(see Section 3.2) between the individual objects being clustered. There are two major methods of clustering – hierarchical clustering and k-means clustering.

Hierarchical clustering can further be subdivided into agglomerative methods, which proceed by series of fusions of the objects into groups, and divisive methods, which separate the objects successively into smaller subsets. A widely used hierarchical clustering method is the one of Ward [18]. He proposed a clustering method, where at each step in the analysis the union of every possible cluster pair is considered and scored using the error sum-of-squares criterion. The union getting the lowest error sum-of-squares is then performed. These steps are repeated until no more pairing is possible. A representation of the fusions performed in the successive steps is given by a dendrogram. As this method always ends up with one cluster of all objects, it is actually the dendrogram which points to the best number of clusters and their contained objects.

3.4 Analysis of RNA Secondary Structure

Today, there are generally two types of approaches to study RNA secondary structures. First, the single sequence approaches which predict the secondary structure based on experimentally determined energy parameters. Second, comparative approaches that try to improve their results by using functionally related sequences. This means that a researcher who found a new functional RNA in one species has to search for its functional homolog in other species to be able to infer the common structural features of these. In my opinion, the real task is to improve the analysis of single sequences as this would allow for reliable predictions for single unknown sequences. This improvement could (or should) of course be based on results from comparative studies. In the following sections I depict the most relevant single sequence approaches of RNA structure analysis. For comprehensibility reasons and as I do not make use of comparative approaches, I will not present these.

3.4.1 Structure Prediction based on Dynamic Programming

Dynamic programming (short DP) algorithms are central to computational sequence analysis. Well known examples are the Needleman-Wunsch algorithm [19] for global sequence alignment, the Smith-Waterman algorithm [20] for local sequence alignment and the Nussinov folding algorithm for RNA secondary structure prediction [21]. As all proper DP algorithms, these algorithms are guaranteed to find the optimal solution for their underlying scoring scheme, meaning the pairwise sequence alignment with maximal score and the secondary structure with maximal number of base pairs, respectively. This expresses that the results of the algorithms are depending very much on their scoring schemes. The base pair scoring of the Nussinov algorithm is too simplistic to give accurate structure prediction, since the driving force in structure formation is the loss of free energy. This loss of free energy depends on the individual bases that form base pairs as well as on the stacking of base pairs. Regions of unpaired bases implicate in

most cases a gain in free energy and destabilize the structure. The individual energy parameters are determined experimentally and get refined from time to time [4, 22]. Altogether they are called the *nearest neighbour energy model*[1] and form the scoring scheme for RNA structure prediction based on free energy minimisation.

A DP algorithm for RNA folding works in two stages. The first part, called the fill algorithm, calculates minimum folding energies for all subwords of the sequence. It starts with all pentanucleotides[2] and extends to larger subwords using the specified recurrences. To prevent exponential explosion, intermediate results are stored in tables and can therefore be reused in subsequent calculations. The second algorithm, called backtracing, computes the structure attaining minimum free energy by searching systematically through the matrices of precomputed energies. The task for this algorithm is to rebuild the path (the sequence of structural elements) that lead to the matrix element with minimum free energy. An elaborate refinement of this backtracing can be used to additionally derive suboptimal solutions.

Besides the "energy model", the results of the structure prediction, especially for the suboptimal solutions, also depend on the types of structures that are considered by the algorithm. Three types are known from literature:

Feasible structures: These are all structures that are viable with regard to the energy model.

Canonical structures: Isolated base pairs, meaning base pairs with no adjacent base pairs, are generally regarded as biologically less meaningful and can therefore be neglected without loosing biological relevance.

Saturated structures: A structure is called saturated if all its helices cannot be extended any further by legal base pairs.

The above distinction of types of structures has minor influence on the minimum free energy structure but major influence on the suboptimal solutions. For RNA sequences S, the number of suboptimal structures (or size of the search space) grows exponential with the length N of $s \in S$ [5]. For feasible structures the approximation of the growth is 2.8^N, for canonicals 1.8^N and for saturated 1.6^N. The effect is also shown in table 3.1.

MFOLD

The first DP algorithm based on the *nearest neighbour energy model* was introduced by Zuker and Stiegler [24]. It is a traditional DP algorithm, meaning that it is dissected into a fill and a backtracing stage. This first version was capable of calculating the *mfe*-structure only. An improved version, also computing suboptimal solutions was introduced in [6]. The most recent version of the algorithm [25, 4] is implemented in the `MFOLD`

[1] This naming is due to the stacking energies depending on the neighbouring base pairs.

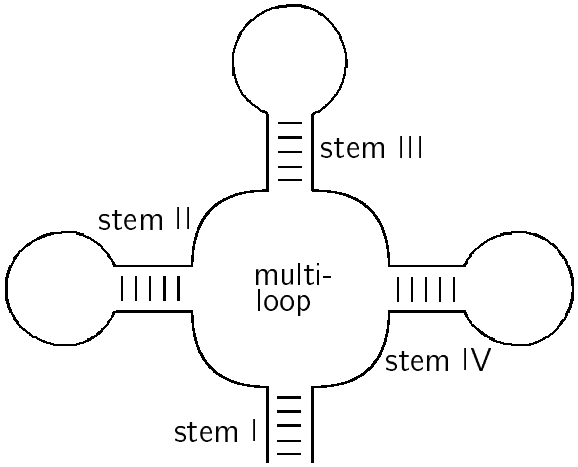
[2] A hairpin, being the smallest structural element, can be formed by at least five nucleotides (two for the closing base pair and a minimum of three for the unpaired region).

Table 3.1: Comparison of the structure space sizes for feasible, canonical and saturated structures. (a) Spliced Leader RNA from L. collosoma. (b) T. thermophila Group 1 Ribozyme Domain. (c) N. crassa 5S rRNA. Taken from Evers and Giegerich [23]

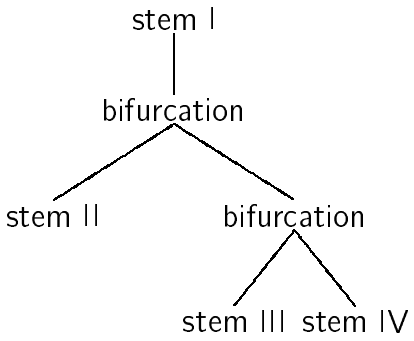
RNA	n	saturated			canonicals			feasibles		
% mfe:		10%	20%	all	10%	20%	all	10%	20%	all
(a)	56	4	19	$2 * 10^6$	9	46	$2.5 * 10^7$	9	49	$6 * 10^{12}$
(b)	69	6	22	$1.15 * 10^8$	10	168	$2 * 10^9$	10	196	$2.2 * 10^{16}$
(c)	40	1	5	1 857	1	5	4 222	1	6	$5.5 * 10^7$

package [26]. The recurrences used in this algorithm reflect the decomposition of the secondary structure into its elementary parts, namely hairpin loop, stacked pair, left bulge, right bulge, internal loop, multiloop and external loop. In fact, the algorithm distinguishes between loops containing 1, 2 or more than 2 base pairs. A hairpin loop is a 1-loop because of its closing base pair. 2-loops are stacked pair, left bulge, right bulge and internal loop all possessing a base pair additional to their closing one. A k-loop ($k > 2$) is referred to as a multiloop with k stems (one stem containing the closing base pair and $k-1$ stems branching out). The 1- and 2-loops find their exact analogon in their reccurences. The reccurence for a k-loop case is not explicit, as it always decomposes the loop into a series of bifurcations regardless of the exact number of structural elements in the multiloop. In the case of a 4-loop (such as in tRNAs, Figure 3.4(a)) this means, that the (same) multiloop can be decomposed in two ways: 1. stem II plus (stem III plus stem IV), as shown in Figure 3.4(b), 2. (stem II plus stem III) plus stem IV, as shown Figure 3.4(c). When calculating the *mfe*-structure only, this ambiguity has no disadvantages, since the minimum is still the minimum. The negative effect of this ambiguity comes into play when also suboptimal structures are computed, resulting in repeated output of exactly the same structure. In the case of an tRNA, the algorithm would compute the cloverleaf structure twice.

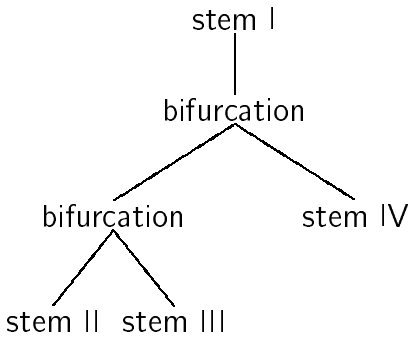
An extension to suboptimal folding and a first solution to the problem of repeated structures was introduced in [6]. The basic idea came with attempts to extend the algorithm to fold circular RNA. Each base pair (i, j) divides the secondary structure into an "included fragment" with energy $E(i, j)$ and an "excluded fragment" with energy $E(j, i)$. The sum of these folding energies is the minimum free energy E_{min} of a structure containing the base pair (i, j). The minimum of $E(i, j) + E(j, i)$ over all possible base pairs is the minimum free energy of the circular RNA [27]. In order to get multiple foldings for a circular RNA within $P\%$ of the *mfe*, it is sufficient to identify those base pairs (i, j) for which $E(i, j)+E(j, i) \geq (1-P/100)*E_{min}$ and subsequently derive the best structure containing that base pair. To generalise this procedure to linear RNA, the linear molecule is handled as if it were circular. In addition, loops containing the origin must be treated as special cases. For example, a hairpin loop containing the origin becomes two single-stranded regions at the 5'- and 3'-end of the RNA. With these methods it is possible to derive multiple foldings for one RNA molecule, but the problem of redundant output remains. For example, each base pair of the *mfe*-structure generates the *mfe*-



(a)



(b)



(c)

Figure 3.4: Decomposition of a multiloop of degree 4, such as in tRNAs. (a) Sketch of a multiloop of degree 4. The Zuker-Stiegler procedure yields two possible decompositions ((b) and (c)), whereas the Wuchty procedure yields solely the decomposition shown in (c).

structure. In order to eliminate this redundancy, the following distance measure d_{Zuker} was introduced:

$$d_{Zuker}(s_1, s_2) = \max_{(i_1,j_1)\in s_1} \min_{(i_2,j_2)\in s_2} \max \begin{cases} |i_1 - i_2| \\ |j_1 - j_2| \end{cases}. \tag{3.6}$$

In words, for each base pair in s_1, find the base pair in s_2 such that the positional difference is minimal. The positional difference of two base pairs (i, j) and (k, l) is the maximum of the values of $|i-k|$ and $|j-l|$. Finally, the actual distance d_{Zuker} is the value for the base pair in s_1 with maximal positional difference. Based on d_{Zuker} the algorithm generates structures so that the distances of all pairs are greater than a preassigned threshold.

Both methods used to compute multiple non-redundant foldings of an RNA are based on heuristics and hence cannot yield the set of all possible secondary structures. The necessity for these heuristics mainly arises from the ambiguity of the algorithm. Thus it seems promising to eliminate this ambiguity, in order to come up with an algorithm capable of computing all possible secondary structures in a non-redundant way, as it is described in the following section.

RNAsubopt

The first solution coping with the ambiguity problem of the Zuker-Stiegler algorithm was published by Wuchty et al. [7]. It is, as well as `MFOLD`, based on the nearest neighbour energy model and implemented in the tool `RNAsubopt` which is part of the Vienna RNA package [8]. The essential difference to the beforementioned approach is the *unique decomposition of multiloops*. The Zuker-Stiegler procedure decomposes a multiloop using a series of bifurcations where each element can be a bifurcation itself. The Wuchty procedure treats multiloops in a different way, allowing for the first element of a bifurcation only stems and no further bifurcation. For the example from above this would result in a bifurcation into a stem and a subsequent bifurcation of two stems (see Figure 3.4(c)), being the only possible and therefore unique decomposition.

This unique decomposition of multiloops allows the systematic and non-redundant generation of suboptimal structures. The backtracing of suboptimal solutions is performed in the way that partial structures are refined by exactly reversing the optimisation procedure used to generate structures from smaller substructures. The exact procedure for the suboptimal backtrack is described in [7].

Probabilistic RNA secondary structure prediction – The partition function

In equilibrium, the individual molecules of an RNA population of the same species do not occur all with the same structure. Most of them are in a state close to the mfe, but some are in different states, or in other words attain alternative secondary structures. The set of these states is often referred to as the ensemble of structures in equilibrium. Of main interest for this ensemble is, which of its states are the ones being occupied most of the time and by most of the individual molecules, or in other words the probabilities of

the states and the state which has the highest probability. McCaskill [28] introduced the use of the partition function to address this problem. In general, the partition function provides a measure of the total number of states weighted by their individual energy at a particular temperature. For an RNA sequence and the set S of all possible structures for this sequence it is defined as follows:

$$Q = \sum_{j \in S} e^{\frac{-E_j}{RT}} \tag{3.7}$$

where E_j is the energy of structure j, R the universal gas constant (0.00198717 kcal/K) and T the temperature in Kelvin. In words, this is the sum of Boltzmann weighted energies of all structures. The probability P of a certain secondary structure s is defined as:

$$P(s) = \frac{e^{\frac{-E_s}{RT}}}{Q} \tag{3.8}$$

where E_s is the energy of structure s in kcal/mol. A general feature of this approach is that for long sequences the probability of an individual structure gets very small. For the leader of HIV-2 genome (546 nt) the probability for the *mfe-structure* in the ensemble is $8.6408 * 10^{-10}$. The computation of the partition function is based on similar recurrences like those for free energy minimisation. Since the partition function is a sum over all possible structures, it is essential for its correct calculation that the recurrences are unambiguous. An implementation is available in `RNAfold` from the Vienna RNA package. With this implementation one has to bear in mind, that for each outgoing stem in the external or a multiple loop, a dangling energy contribution is added regardless of whether the neighbouring bases are actually paired or unpaired. The authors state that this can be seen as an approximation to coaxial stacking, whose energy contribution is in general higher than the summed dangling energies. In addition to the partition function and the probability of an individual structure, the algorithm is capable of computing the probability of base pairing for each feasible base pair.

SFOLD

The partition function cannot only be used to calculate the probability of individual structures or base pairs. Ding and Lawrence [9] introduced a statistical sampling algorithm which is implemented in the tool `SFOLD`. In each step of the recursive backtracing procedure, base pairs and the structural element they belong to are sampled according to their probability, obtained from the partition function calculation in the forward step of the algorithm. Features of the sampling procedure are, that each run is likely to produce a different sample and that the same structure can be sampled multiple times, where the *mfe*-structure is the most frequent structure, as it has the highest probability. Note, that the *mfe*-structure is not guaranteed to be present in the sample, especially for long sequences. The major problem, in my point of view, is that the algorithm splits the struc-

ture sample into 10 clusters based on energy values. For this purpose the free energy range covering all structures in the sample is divided into ten equally spaced intervals. For each free energy interval, the structure with the lowest free energy is selected as the representative. A consequence could be that the representative structure is from another family than all other sampled structures in the interval.

Algebraic Dynamic Programming

The previous sections contained information about individual tools for RNA structure analysis. In this section I do not describe a tool, but a method for the rational design of tools that use dynamic programming. The central idea is to describe the structure space of RNA in an abstract way and to use this as the basis for its analysis. A program that was developed with this method is `RNAshapes`, which will be presented in Chapter 5.

The main part of developing a DP algorithm is the design of recurrences that express relations between tables holding intermediate results. These recurrences are difficult to explain, hard to implement, susceptible to subscript errors and almost impossible to debug: slight errors give rise to suboptimal solutions, which are very laborious to detect. Additionally, no general guidance in the development of DP algorithms is available. Classical DP algorithms perform two tasks in an interleaved fashion: First they construct the search space, or in other words, the set of all possible answers. Second they evaluate these answers and make choices according to some optimality criterion, e.g. minimum free energy. This interleaving is essential to prevent combinatorial explosion, but the result is that all information on the problem, which is addressed with this algorithm, is also hidden in its recurrences. In the case of RNA folding such hidden information are the description of the folding space, the objective function (e.g. minimisation) and the scoring.

Algebraic Dynamic Programming (ADP) is a method for algorithm development, designed to alleviate this situation, and provides a framework to design, tune and test DP algorithms on an abstract level. ADP enables the separate description of the search space and the evaluation, shifting the fusioning task from the programmer to the framework. In the following I will give a very brief overview of the essential parts of an ADP algorithm, details about ADP can be found in [29, 30, 31, 32].

In ADP the search space is described by a grammar, which can be seen as the device that derives all candidate solutions that are to be evaluated. The evaluation and the choice are incorporated in an evaluation algebra, represented as a set of formulas. The following simplistic example will give the reader an impression on how a grammar and an evaluation algebra can be designed: Think of the early approaches to RNA folding by Nussinov. The building blocks of RNA are hairpin loops, stacked base pairs, left bulges, right bulges, internal loops and splits for simplicity. A hairpin loop consists of a closing base pair and unpaired bases in between, a stacked base pair of a base pair and some structural element in between the bases of the base pair. Left bulges, right bulges and internal loops have a closing base pair and unpaired bases on either the left, the right, or both sides. A split is an element that concatenates two structural elements. The ADP

framework provides a notation using ||| ('or') to separate alternative rules, <<< ('apply to') to denote the application of some operator to its arguments which are separated by ~~~ ('next to') and finally ... ('choose') reflecting the application of the choice function. Here is the resulting grammar:

```
folding_grammar = axiom structure where
   structure = hairpin <<< base ~~~             region              ~~~ base |||
               st_pair <<< base ~~~            structure            ~~~ base |||
               l_bulge <<< base ~~~ region ~~~ structure            ~~~ base |||
               r_bulge <<< base ~~~            structure ~~~ region ~~~ base |||
               in_loop <<< base ~~~ region ~~~ structure ~~~ region ~~~ base |||
               split   <<<       structure        ~~~        structure       |||
               unpair  <<<         base           ~~~        structure       |||
               nil     <<< empty                                    ... f_choice
```

Note, that the `split` production in this example gives rise to the same ambiguity as in the Zuker-Stiegler algorithm. The next step is to design an evaluation algebra that holds for each operator of the grammar a function how to score it and the choice function reflecting the optimality criterion. In order to stay simple, the scoring is based on base pairs and the choice function will be maximum to compute the structure with maximum base pairs. The resulting evaluation algebra is:

```
hairpin base1         region          base2 = 1
st_pair base1        structure        base2 = 1 + structure
l_bulge base1 region structure        base2 = 1 + structure
r_bulge base1        structure region base2 = 1 + structure
in_loop base1 region structure region base2 = 1 + structure
split     structure1       structure2       = structure1 + structure2
unpair        base         structure        = structure
nil = 0
f_choice = maximum
```

The above combination of grammar and evaluation algebra computes the structure with the maximum number of base pairs in a recursive way. This means that it would compute the same partial structure each time it is used and does not make use of precomputed and stored solutions. Tabulation can be added by simply assigning the keyword `tabulated` to each element that should be tabulated. A further drawback is, that neither the grammar nor the algebra checks for correct base pairing (A–U,C–G,G–U). For this purpose the ADP framework provides the `with` operator which is used to apply predicates. In order to ensure that the two bases in the `hairpin`, `st_pair`, `l_bulge`, `r_bulge` and `in_loop` productions can actually form a base pair it is sufficient to apply the predicate `with basepairing` where appropriate. Adding tabulation and

base pair checking results in the following grammar:

```
folding_grammar = axiom structure where
   structure =  tabulated(
                (hairpin <<< base ~~~              region              ~~~ base |||
                 st_pair <<< base ~~~            structure             ~~~ base |||
                 l_bulge <<< base ~~~ region ~~~ structure             ~~~ base |||
                 r_bulge <<< base ~~~            structure ~~~ region ~~~ base |||
                 in_loop <<< base ~~~ region ~~~ structure ~~~ region ~~~ base
                                                        ) 'with' basepairing  |||
                 split   <<<       structure         ~~~         structure       |||
                 unpair  <<<          base           ~~~         structure       |||
                 nil     <<< empty                                    ) ... f_choice

basepairing i j = basepair seq!i seq!j

basepair 'a' 'u' = TRUE
basepair 'u' 'a' = TRUE
basepair 'c' 'g' = TRUE
basepair 'g' 'c' = TRUE
basepair 'u' 'g' = TRUE
basepair 'g' 'u' = TRUE
basepair  _   _  = FALSE
```

This example expresses the ease-of-use once being familiar with the ADP notation. It is straightforward to implement RNA folding based on free energy minimisation, using the nearest neighbour energy model. The corresponding grammar and evaluation algebras for free energy minimisation, pretty printing of structures as "Vienna"-strings, counting structures and more are given in Appendix B.

3.4.2 Other Approaches to RNA Secondary Structure Prediction

The DP approaches described above have in common that they will always find the overall optimal solution, i.e. the *mfe*-structure, which is (very) often not the native *in vivo* structure. A reason for this might be, that during structure formation intermediate states appear which are energetically stable enough to prevent their melting. In other words, the native structure is determined by a specific folding pathway which captures the molecule in a local minimum [33]. These kinetic effects do not only lead to structures different from the *mfe*, but also play an important role for ribozyme activity and conformational switching.

The algorithms for simulating RNA folding pathways usually consider the folding as a stepwise addition or removal of stems, starting from the most favourable, or "nucleation points". In the simplest case, the most stable stem from those compatible with previously formed stems is added at each step of the procedure, such as presented by Nussinov and Pieczenik [34], Martinez [35], Abrahams et al. [36]. The main disadvantage of these algorithms is, that they do not consider deletion of stems when other pairings become more favourable. In the following I describe more elaborate approaches to kinetic folding.

STAR

Gultyaev et al. [37] proposed a genetic algorithm (GA) for secondary structure prediction as part of the `STAR` package. GAs solve optimisation problems by simulating the process of natural evolution. In short, after generating an initial population of solutions, a GA produces some new solutions by randomly changing (GA-Mutation) the previous ones and combining certain features of different parental solutions (GA-Crossover). Subsequently, a new population is generated based on selection according to a "fitness" criterion. Iteration of this procedure eventually yields a solution with a high level of fitness. In the case of RNA secondary structure, the GA-Mutations are addition and removal of stems and GA-Crossover is recombination between structures, resulting in a structure that contains some stems from one structure and other stems from another structure. The "fitness" of an individual structure is calculated as the difference between its energy and that of the best structure, divided by the number of stems that are present only in one of these solutions. In addition to folding entire RNA molecules, the authors present a procedure for the simulation of folding during RNA synthesis. This is achieved by restricting the folding at every GA iteration to a growing part at the 5' end.

GArna

The program `GArna` by Titov et al. [38] is rather similar to the previous approach. It makes use of the same genetic operations. The major difference is the fitness function. The probability of eliminating structure i from the population is calculated according to the following equation:

$$p_i = \frac{M * e^{\frac{E_i}{\Delta E}}}{\sum_{i=1}^{N} e^{\frac{E_i}{\Delta E}}}, \tag{3.9}$$

where E_i is its free energy. Parameters M and ΔE have the following meaning: M is the expected number of structures eliminated in one generation, ΔE is the effective energetic resolution, that is, the difference in energy that makes the fitness of two structures differ e-fold.

Kinefold

Another approach allowing for both addition and disruption of stems was proposed by Isambert and Siggia [39]. It is based on a kinetic Monte Carlo algorithm, where each step involves making or breaking a stem. The transition rates for these events are computed based on entropic and free energy terms. This is done for each new structure along the folding path. The actual transition is selected at random with a probability proportional to its rate. Iteration of this procedure is performed until the system appears stationary. This approach as well as the approach by Gultyaev *et al.* is capable of predicting certain kinds of pseudoknots, e.g. those of the H-type.

Kinfold

An approach going into more structural detail is `kinfold` by Flamm et al. [40], as it considers opening and closing of base pairs as the elementary transitions. This means that all suboptimal structures can be reached and, hence, the complete structure space is considered. `kinfold` simulates a stochastic process based on a so called "move set" which defines the elementary transitions a secondary structure can perform when in solution. This "move set" contains opening and closing of an individual base pair, as well as the so called "shift move" which allows the changing of one partner in a base pair. The `kinfold` approach models RNA folding as a Markov process where each transition (from structure S_i to S_j) occurs dependent on a transition rate k_{ij}. This rate is defined as $k_{ij} = e^{\frac{-\delta G_{ij}}{2RT}}$, where $\delta G_{i,j} = E_{S_i} - E_{S_j}$ [41].

3.4.3 Structure Space Analysis

In this section I want to give detailed information on existing approaches for studying the structure space of RNA. Among these are tools to deduce conformational switching, structural well-definedness and other properties.

Structural well-definedness: Approach by Wuchty et al.

Based on the algorithm for complete suboptimal folding Wuchty et al. [7] describe three indicators to capture the definition of the *mfe*-structure: (a) mean gap energy, that is, the average energy difference of structures in the vicinity of the global minimum; (b) Boltzmann weighted mean base pair distance; and (c) topological diversity, that is, the number of different coarse-grained structures around the ground state. These were used to assess the influence of base modification on structural well-definedness in tRNA sequences. Therefore natural tRNA sequences (with modifications) were compared to modified and unmodified artificial sequences having the cloverleaf structure as their optimum. Their study showed that the base pair distance is the best indicator for how well the ground state is defined. Furthermore, it arose that base modification considerably enhances the definition of the *mfe*-structure, as the unmodified sequences showed minor determination of the ground state than modified sequences. Additionally, the position of the modifications in the sequence was shown to influence structural definition, as ground states of the natural tRNA sequences were better defined as those of the modified artificial sequences.

The analyses relied on the comparison of different sets of sequences and it was able to show that these sets show different behaviour for the chosen indicator. Unfortunately, this indicator cannot be used for the classification of individual sequences, as the sets of sequences could not be completely separated from each other, at least for the chosen examples.

Structural well-definedness: Approach by Kitagawa et al.

Another method to deduce structural well-definedness of the *mfe*-structure is proposed in [10]. Central to this is the introduction of a new distance measure for secondary structures, termed "tree representation distance" (TRD). The TRD uses a representation of RNA secondary structure which codes the nesting of the structural elements by the number of outgoing branches for each element, e.g. the structure shown in Figure 3.1(b) is represented as "1200" (1 stem going out of the external loop, 2 stems branching out of the multiloop and two times 0 for the two hairpin loops). The distance between two structures in this representation is the sum of the differences at each position of the optimal alignment, i.e. the alignment minimising this sum. The complete approach works as follows: First, a set of suboptimal structures is calculated using `MFOLD`. Second, each structure of this set is compared to the *mfe*-structure. Thereby, the energy difference ΔE and the TRD is calculated and plotted in a TRD,ΔE coordinate system. In the case of a well-defined structure the points in the plot mainly appear above the bisecting line which is termed "uni-valley" profile. In the other case on or more additional valleys appear, meaning that structures exist which have a low TRD and a high ΔE. An outcome like this leads to the classification as a "multi-valley" profile. Based on this method it was shown that the human snRNAs U2, U3, U5, U6 have a "uni-valley" and U1 has a "multi-valley" profile, which is in correspondence to experimental data.

Prediction of Conformational Switching using SFOLD

The statistical sampling procedure implemented in `SFOLD` computes a subset of all suboptimal structures according to the equilibrium distribution. For a conformational switch this means that structures from two valleys (corresponding to the alternative positions of the switch) should be sampled. Ding and Lawrence [9] used the Spliced Leader of *Leptomonas collosoma* [42] to exemplify the use of `SFOLD` for this purpose. In their presentation they describe that they manually assigned 100 sampled structures to two equivalence classes and that these classes are in terms of structure rather dissimilar. The representative, i.e. energetic best, structures of these classes could be shown to correspond to the experimentally deduced ones. In my point of view this simple example is not sufficient to show the applicability of `SFOLD` to reveal conformational switching. First of all, the near-optimal structure space (e.g. up to 3 kcal/mol above the *mfe*) generated with `RNAsubopt` holds only a small number of structures (~ 50) and is therefore much easier to handle, while delivering the same results. Second, a manual assignment to equivalence classes is hard to trust, especially when knowing the number of classes you should get. The last point I want to mention is that this approach is only capable of predicting the alternating conformations and not the property of serving as a conformational switch, as this would at least require to check for an energy barrier separating the two conformations.

barriers

The tool `barriers` [12] analyses the complete or part of the folding space of an RNA to find local minima and saddle points connecting these. As input it takes the energy sorted list of suboptimal structures generated by `RNAsubopt`. During the calculation two lists of valleys are needed, an active and an inactive one. The global minimum x_1 belongs to the first active valley V_1, while the list of inactive valleys is empty initially. Going through the energy-sorted list of secondary structures in increasing order there are three possibilities for each structure x_k at step k:

- x_k has one or more neighbours in exactly one of the active valleys V_i. In this case x_k belongs to V_i.
- x_k has no neighbours in either the active or the inactive valleys that have been found so far. Then x_k is a local minimum and determines a new active valley V_l.
- x_k has neighbours in more than one active valley, e.g. $\{V_{i1}; V_{i2}; ...; V_{iq}\}$. In that case x_k is a saddle point connecting these local minima. x_k is added to the valley with the lowest energy (V_{i1}). All structures from the valleys $V_{i2}; ...; V_{iq}$ are then copied to V_{i1} while the status of the valleys $V_{i2}; ...; V_{iq}$ is changed from active to inactive. As a result, a new structure having neighbours only in (the inactive) valley V_{i2} is assigned to (the active) valley V_{i1}.

This algorithm can be imagined as flooding the structure landscape. Think of a simple landscape with global minimum A and one local minimum B. The algorithm starts and sees structures with neighbours only in A, which are added to V_A. As soon as the energy of the local minimum B is reached a new active valley V_B is opened. The following structures are assigned to either V_A or V_B, until a structure is seen that has neighbours in both. This is the saddle point connecting A and B and it is added to V_A, as this one achieves lower energy. The two valleys are merged into the active valley V_A, and valley V_B gets inactive.

The program `barriers` does not only compute local minima and saddle points, but also generates the so called "barrier tree" as a visualisation of the landscape. In the barrier tree the local minima are leaves and saddle points are nodes connecting either two local minima, a local minimum and a saddle point, or two saddle points. The length of the edges corresponds to the energy difference between the connected elements.

Applying `barriers` to a conformational switch (Spliced Leader of *L. collosoma*), reveals a barrier tree showing two major valleys connected by a saddle point with very high energy (see Figure 3.5), meaning that the minima of these valleys are separated by a very large energy barrier, such as it is expected for a conformational switch.

The actual implementation requires to store all previously read conformations in memory which causes the use of `barriers` to be limited to (partial) landscapes of moderate size, e.g. $\sim$ 20 million structures. This is enough to analyse the structure space with negative energy of the Spliced Leader, but is insufficient for longer sequences.

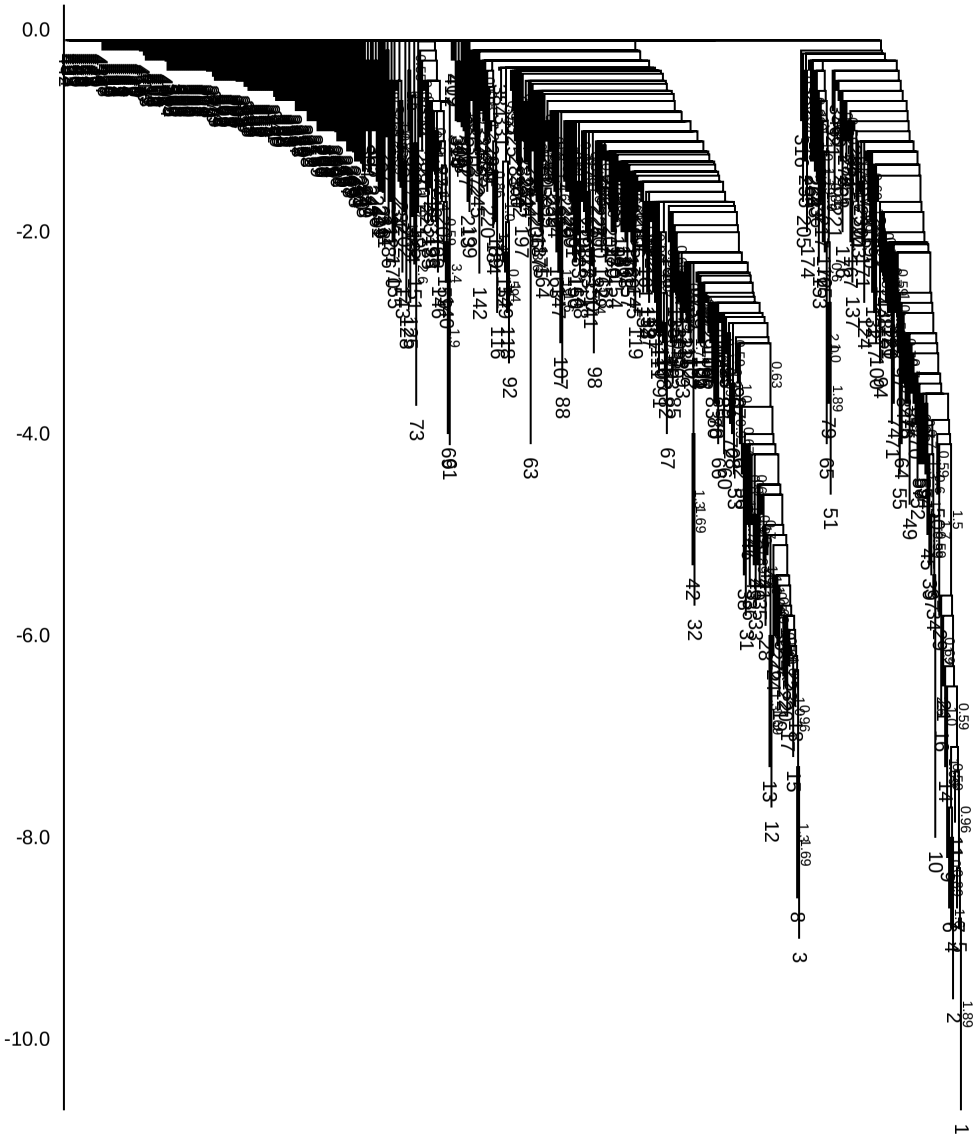


Figure 3.5: Barrier tree of the structure space of the Spliced Leader of *L. collosoma*. The leaves number 1 and 3 correspond to the alternating conformations of this switch. They are separated by an energy barrier of $10.6 kcal/mol$.

CHAPTER 4

Prediction of Alternating RNA Secondary Structures

The property of an RNA to change between two different conformations is known as conformational switching. As this property is hidden in the structure space, it cannot be detected based on the *mfe*-structure only. Therefore, it is necessary to take also suboptimal structures into account. An approach to the elucidation of conformational switches based on stochastic sampling of suboptimal structures was recently presented by Ding and Lawrence [9] and is described in Section 3.4.1. For the recently discovered riboswitches (described below) Bengert and Dandekar [43] present a program named `Riboswitch finder`, which is intended to search for riboswitches in a given sequence. It makes use of combined sequence and structure motifs for the *B.subtilis*-like riboswitches, but does not check for alternating conformations. An algorithm for the design of multi-stable RNA sequences by combinatorial optimisation was described in Flamm et al. [44]. More recently Barash [45] presented an approach which focuses on systems that initially reside in a stable state. The algorithm uses a local procedure to predict which mutation, given a stable wildtype structure as input, can be introduced in order to create the optimal bi-stable RNA. In the following sections I will describe the `paRNAss` approach for the prediction of alternating RNA secondary structures, of which the idea was developed by Marc Rehmsmeier [17]. A first version of `paRNAss` was implemented by Dirk Haase and presented in [11]. I used this version as a starting point, incorporated new algorithms to improve its performance and evaluated the approach to show its applicability.

4.1 Conformational Switching in RNA

Conformational RNA switches have been proven or are suspected to be involved in several important processes: regulation of gene expression in prokaryotes by attenuation [46], translational regulation of *E. coli* ribosomal protein S15 [47], regulation of self-cleavage activity of *Hepatitis Delta Virus* [48], translocation process in protein biosynthesis [49], trans splicing in trypanosomes [42] and splicing of pre-mRNA by spliceosomes [50]. Re-

cently, a new class of conformational switches, the so called *riboswitches*, has been described [51, 52, 53]. Members of this class are characterised as mRNAs that possess an aptamer in the 5'-UTR. Aptamers are RNA elements which directly bind a molecule. In case of riboswitches these aptamers bind a metabolite of the biosynthetic pathway in which the protein, the mRNA encodes, is involved. The binding of the metabolite to the aptamer induces a structural rearrangement which alters translational efficiency.

To give an idea about conformational switching I will explain the mechanism of the Attenuator of the pheS-pheT-Operon of *Escherichia coli* in more detail: The operon codes for two subunits of an enzyme of the phenylalanine-tRNA biosynthetic pathway. The Attenuator is an element in the coding part of the mRNA of this operon and contains five codons for phenylalanine, three of which are consecutive. The secondary structure of the Attenuator comprises a stable hairpin at its 3'-end, which leads to termination of translation. If the amount of tRNA-phe is high the ribosome reads fast over the three adjacent phe-codons and translation gets terminated by the "terminator" hairpin (Figure 4.1(b)). In the case of phenylalanine starvation the ribosome is stalled at the phe-codons allowing for a structural rearrangement which eliminates the terminator hairpin and ,thereby, leads to the translation of the entire operon, enabling biosynthesis of phenylalanine-tRNA (Figure 4.1(c)).

4.2 The paRNAss Approach

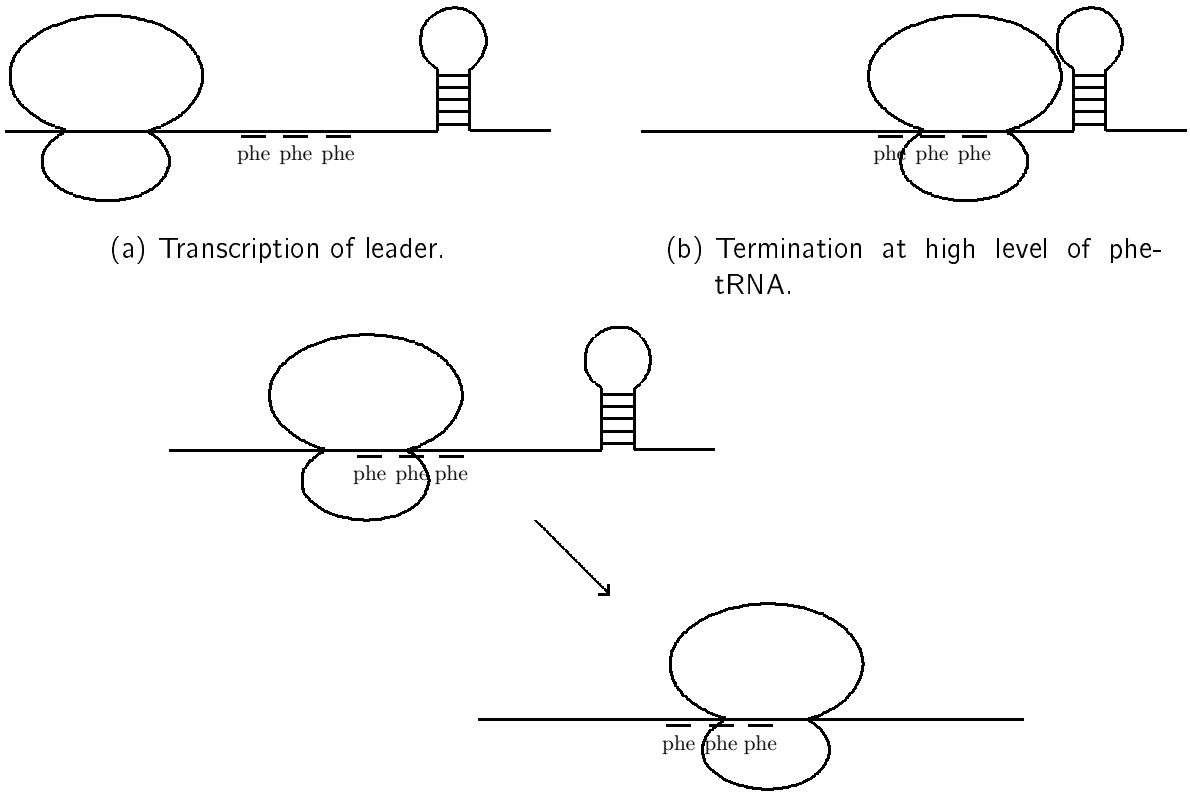
`paRNAss` is a computational approach to the prediction of conformational switching in RNA which makes use of RNA secondary structure prediction and pairwise structure comparison. It is based on the following hypotheses about the structure space of a conformational switch which is depicted in Figure 4.2:

1. There must be a local minimum in the structure space close (in terms of energy) to the overall energetic minimum. Their structures must be significantly different, in order to represent two positions of the switch with different regulatory function.

2. The structures residing in these minima must be clearly separated by an energy barrier to ensure that each conformation is stable and switching can only be triggered by external events.

3. The structure space must not provide another local minimum close to the overall energetic minimum, as we assume that the RNA automatically finds the alternative position once the change is triggered.

The folding space can contain tens or hundreds of local minima. Only by checking all three conditions one achieves reliable predictions. The application of the first and second condition filters out RNAs whose energy landscape provides only one valley, whereas the third condition rejects those with three or more.



(a) Transcription of leader.

(b) Termination at high level of phe-tRNA.

(c) Readthrough if ribosome gets stalled.

Figure 4.1: Mechanism of attenuation. (a) Transcription of leader peptide. (b) High level of phe-tRNA enables fast processing of phe-codons. Terminator hairpin persists and translation stops. (c) Low level of phe-tRNA stalls ribosome at the three consecutive phe-codons. This delay allows for a structural rearrangement which breaks up the terminator hairpin and enables readthrough into coding region.

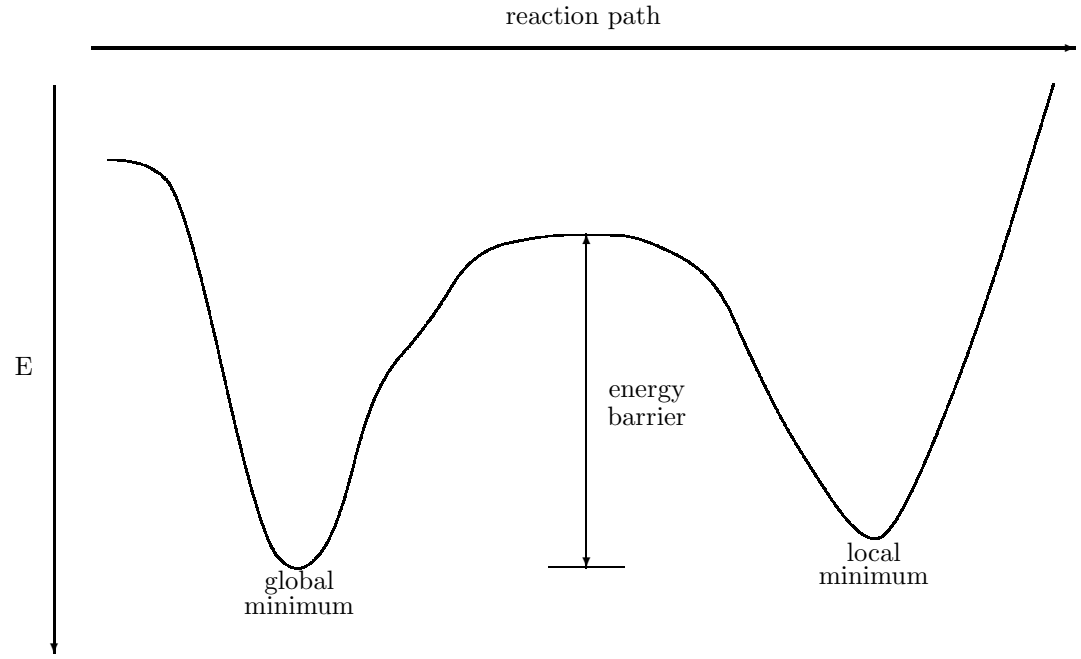


Figure 4.2: Sketch of the structure space of a conformational switch. The structure space holds two valleys that are separated by a mountain representing the energy barrier.

4.3 A Run Through a paRNAss Experiment

The `paRNAss` approach takes four parameters: Parameter P gives the energy range in kcal/mol (default: 3). Parameter F gives the kind of structures, namely feasibles or canonicals (probably refolded), that are to be considered by the folding program (default: canonicals). The temperature in °C is given via Parameter T (default: 37). Parameter N gives the maximal number of structures that should be analysed (default: 50).

A `paRNAss` experiment takes five steps (examples mentioned below refer to the Spliced Leader of *L. collosoma*):

Step 1: Sampling the structure space Using an RNA folding program, a sample set $S = \{s_1, .., s_p\}$ from the folding space of the target RNA is drawn. In the case that the number of structures under the given values for P, F and T exceeds N, N of them are chosen equally distributed to form the sample set. If Hypotheses 1 and 3 are fulfilled, this set should contain members of two families of structures. In the other case, this set may contain one or more than two families, which leads to the classification as a non-switch in the subsequent steps.

Step 2: Pairwise distance calculation For all $s_i, s_j \in S$, their pairwise distance $d_\delta(s_i, s_j)$ is calculated. This is done for the energy barrier distance d_{EB} and at least one other distance measure d_2 on the structure space, and the results are plotted in a d_{EB}, d_2 coordinate system. In fact the plot holds two points for each comparison as

actually the values of d'_{EB} (see Eq. 3.4) are plotted, which provides additional information. If both elements in a pair are from the same structure family, their distance should be small. Conversely, if both are from different families, their distance should be large. Thus, the plotted distance diagram should exhibit two clouds of points, one in the lower left, one in the upper right (Figure 4.3(a)). If this is not the case the diagram could look as shown in Figure 4.3(b).

Step 3: Clustering Using a standard clustering algorithm S is split into two disjoint clusters C_1 and C_2, based on the pairwise distances under either d_{EB} or d_2.

Step 4: Consensus structure calculation For each cluster C_i, a consensus structure c_i is derived by first taking all base pairs present in the majority of the members of C_i, and then reapplying the folding algorithm with these base pairs fixed. Figure 4.4 shows the consensus structures c_1, c_2 derived for the example.

Step 5: Consensus structure validation For all $s_i \in S$ the energy barrier distances $d_{EB}(s_i, c_1)$ and $d_{EB}(s_i, c_2)$ are calculated and plotted as points in a c_1, c_2 coordinate system. Again, d'_{EB} instead of d_{EB} is plotted. If Hypothesis 2 is fulfilled, the c_1 family of sample structures will show up as a cloud of points near the y-axis, the c_2 family near the x-axis, like shown in Figure 4.5. If not, either a significant number of points is near the bisecting line or more than two separate clouds appear. To ensure that the consensus structures do not represent the two possible un-knotted projections of a pseudoknot structure, the *pseudoknot measure (pkMeasure)* for the consensus structures is calculated. For the example this results in $pkMeasure = 14$.

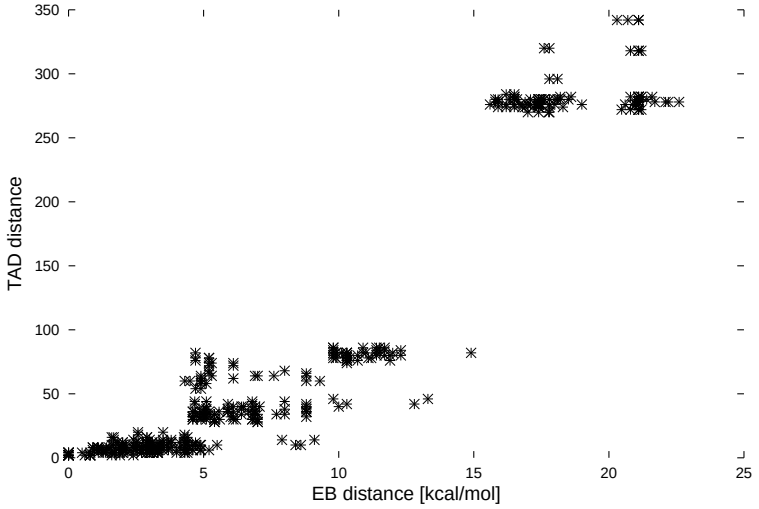
If the outcome of steps 1 - 5 is as described above, I predict the possibility of conformational switching between structures c_1 and c_2. Together, they are called *predicted alternating RNA conformations (parnac)*. This is as far as one can go in silico; the final proof for the predicted conformational switching has to be achieved experimentally.

4.4 Recent Improvements of paRNAss

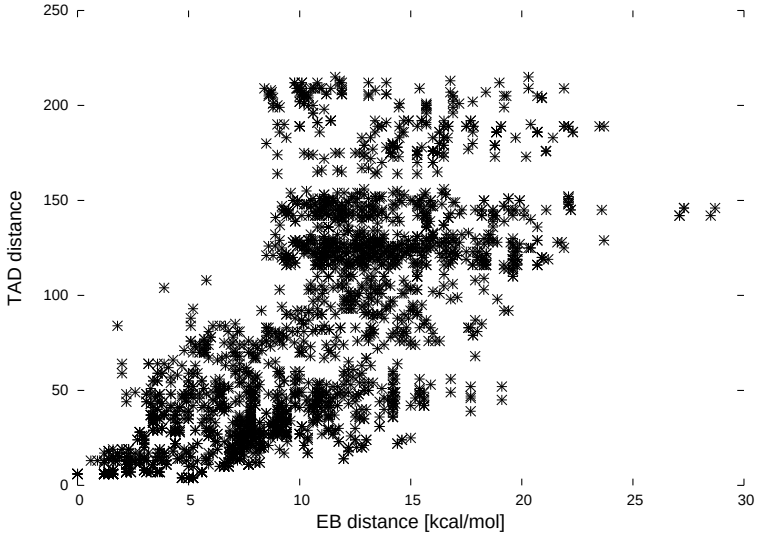
Due to insufficient algorithms, for the pairwise distance calculation and especially for generating the structure sample, the results of the first version of `paRNAss` were not as expected. The following improvements and the availability of more computational power increased the reliability and speed of this strategy and made possible the evaluation procedure described in Section 4.6.

4.4.1 Energy Model

The use of an up to date energy model [4, 54] is essential to obtain biologically relevant foldings. `paRNAss` now takes advantage of the newest versions of `RNAsubopt` and `RNAfold` (Vienna RNA package [8], Version 1.4).

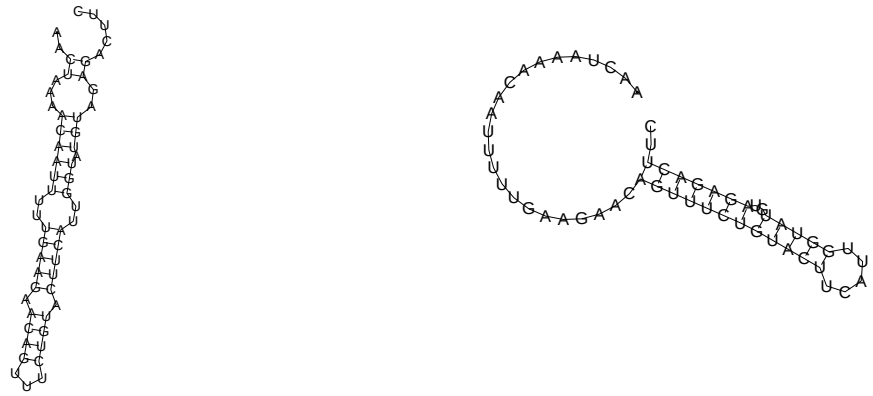


(a) Distance plot for Spliced Leader.



(b) Distance plot for a non-switch.

Figure 4.3: Distance plots for a switch and a non-switch. Each "star" corresponds to a pair of structures. Plot (a) shows two well separated clouds of points, whereas the points are widely spread in plot (b).



(a) c_1 (b) c_2

Figure 4.4: Predicted conformations for Spliced Leader of *L. collosoma*.

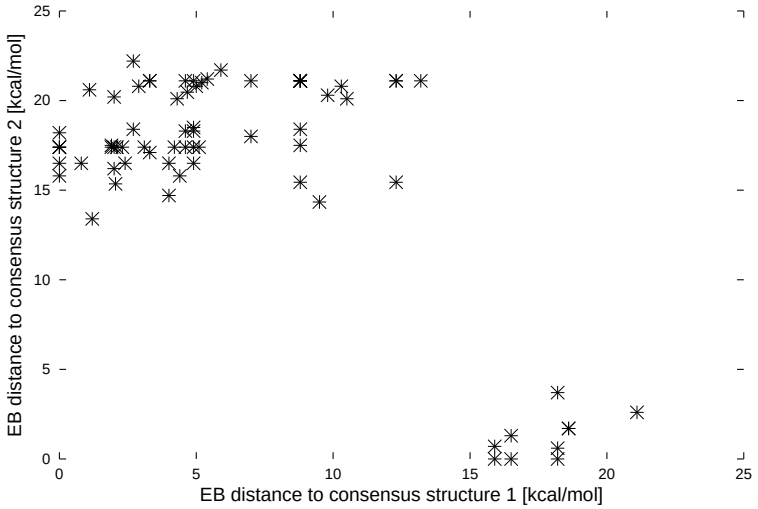


Figure 4.5: The validation plot shows the distances of all sampled structures to the two consensus structures c_1 and c_2. The pseudoknot measure is 14.

4.4.2 Restricting the Search Space to Canonical Structures

Due to the calculation of pairwise distances, the operating time of `paRNAss` is quadratic in the size of the sample set. To increase the performance it is therefore favourable to have a smaller sample set. This is achieved by calculating only canonical structures, meaning structures without isolated (unstacked) base pairs. Approximately, this restriction leads to a reduction in the size of the search space from 2.8^N to 1.8^N, where N is the sequence length. Furthermore this restriction does not only increase the performance but also the distinctive power, as in some cases the occurrence of isolated base pairs in mainly unpaired regions has a negative effect on the distance measures, which may imply uncertainties in the interpretation of the distance plots.

4.4.3 Refolding the Structure Sample – Local Minima

The structures one is interested in are the local minima of the structure space. A local minimum is a structure which has the lowest free energy with respect to its neighbouring structures. Therefore, the set of local minima depends on the definition of neighbourhood, which is normally the difference in only one base pair. As mentioned before, complete suboptimal folding produces a large number of structures and also a large number of neighbouring structures one is usually not interested in. A brute force approach to the elimination of these unwanted structures is as follows: For each suboptimal structure produced by `RNAsubopt` use this as a constraint for constrained folding using `RNAfold -C`. If the newly predicted structure is different from the original input, `RNAfold` found possibilities to optimise the structure and therefore the original structure is deleted from the structure sample.

4.4.4 An Improved Distance Measure based on RNA Secondary Structure Alignment

Good performance of the distance measure is essential for the `paRNAss` approach. Many of the commonly used methods are not free from artefacts. Recently, a promising distance measure d_{TAD} based on tree alignments has become available, implemented in the tool `RNAforester` (see 3.2.3). While `paRNAss` still provides other distance measures, I now prefer d_{TAD} .

4.5 Applications

Data-mining the literature delivered information on 23 known structural switches in RNA, two of which are described as tertiary structural switches. For the remaining 21 sequences, I extracted the part containing the switch and analysed it to test `paRNAss` on true positive and false negative predictions. The parameters were adapted to get best possible results. For readability reasons I will give detailed information of the analyses for six published switches in this section, the remaining can be found in Appendix A.

4.5.1 Attenuator

Function The mode of action of the Attenuator is described in detail in Section 4.1. Just as a quick reminder: The Attenuator is a regulatory element in the leader of the pheS-pheT-operon of *E.coli*. It can switch between a translational active and an inactive conformation depending on the concentration of phe-tRNA.

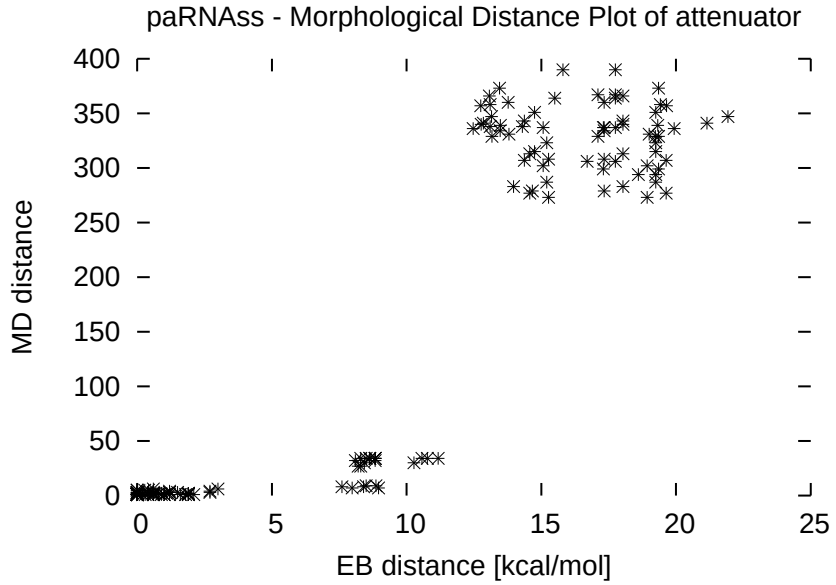
Parameters

- Energy range: $2.4 kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 50

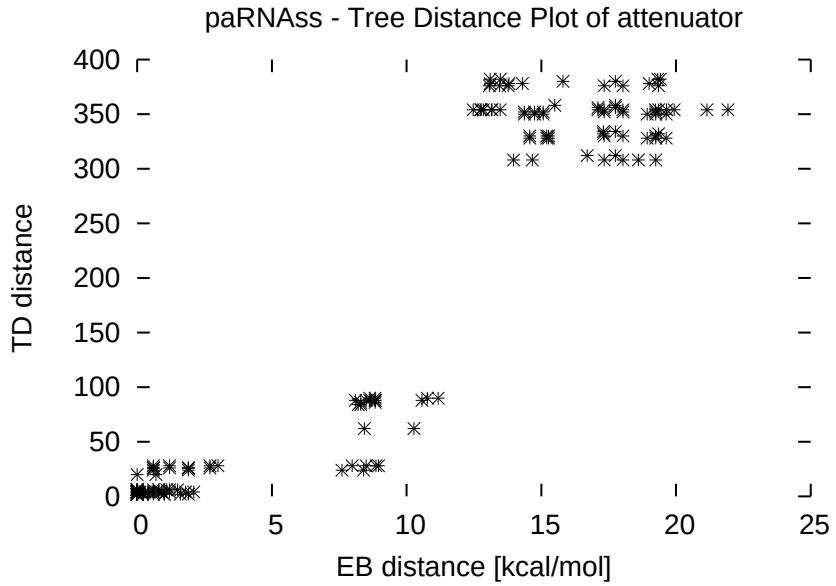
Pairwise Distance Calculation The pairwise distance calculations based on d_{MD} and d_{TAD} delivered the results shown in Figure 4.6. For both distance measures the distance plots show three clouds of points, of which the two in the lower left part of the plot are separated mainly by the energy barrier distance, while showing relatively low distances for d_{MD} and d_{TAD}. A look at the list of analysed suboptimal structures (data not shown) reveals, that this is due to the existence of one structure showing slight variations in the sense that it carries a hairpin comprising the same sequence but with different base pairs. Due to these facts it is reasonable to regard the two clouds in the lower left as one. For this reason the distance plots imply a conformational switch.

Consensus structures Clustering based on d_{MD}, d_{TAD} and d_{EB} and subsequent consensus structure calculation resulted in the conformations shown in Figure 4.7, which are the same for all distance measures. These conformations are in correspondence with the experimental results, as only c_1 carries the terminator hairpin at its 3'-end. Additionally, c_1 has a lower free energy (-21.2 kcal/mol) than c_2 (-20.93 kcal/mol) which expresses that the attenuator is normally in the termination conformation and has to switch into the readthrough conformation. In order to further validate the consensus structures the usual validation process of `paRNAss` was performed.

Consensus structure validation The validation plot in Figure 4.8 shows three clouds, which normally leads to classification as a non-switch. As already mentioned above, one structure in the sample shows slight differences which has a reasonable impact on d_{EB}, and it is again this structure which causes the two points near the bisecting line. With this extra knowledge at hand it is sensible to classify the result of the analysis as positive, meaning that the switching property of the Attenuator was successfully detected with the help of `paRNAss`. The calculation of $pkMeasure$ resulted in 23 which supports the correct prediction. A fact that is also expressed by this analysis is, that minor structural differences may lead to uncertainties in the interpretation of the distance and validation

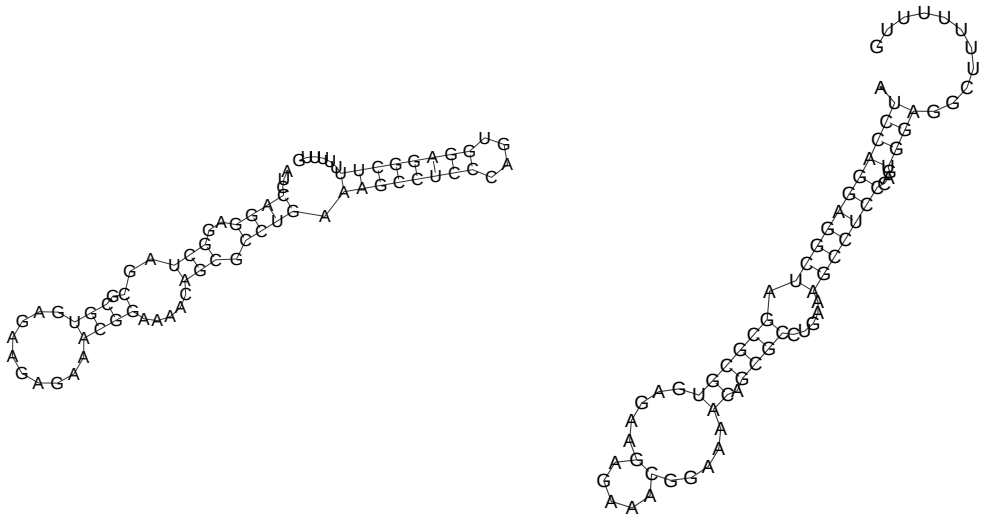


(a) d_{MD}



(b) d_{TAD}

Figure 4.6: Distance plots for pheS-pheT-Attenuator.



(a) c_1 (b) c_2

Figure 4.7: Consensus structures for Attenuator based on d_{MD}, d_{TAD} and d_{EB}.

plots. In such cases it is a good idea to take a look at the sampled structures to get an idea what raises the problem.

4.5.2 Spliced Leader

Function The Spliced Leader was the first known conformational switch, which was also the reason for me to choose it as an example in the description of the `paRNAss` approach. Here, I want to give a more detailed description and analysis of this RNA: In general pre-mRNA splicing is a process by which introns are excised and the exons are then ligated to form the mature transcript. This process is called *cis* splicing. In Trypanosomes and Nematodes also a *trans* splicing mechanism is known, which involves the transfer of a 5' exon from a short *Spliced Leader* transcript onto a preexisting mRNA [55, 56, 57]. This addition serves to separate polycistronic transcripts into individual mRNAs [58]. The actual functions of the alternate conformations of the Spliced Leader are yet to be revealed.

Parameters

- Energy range: $2.9 kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 50

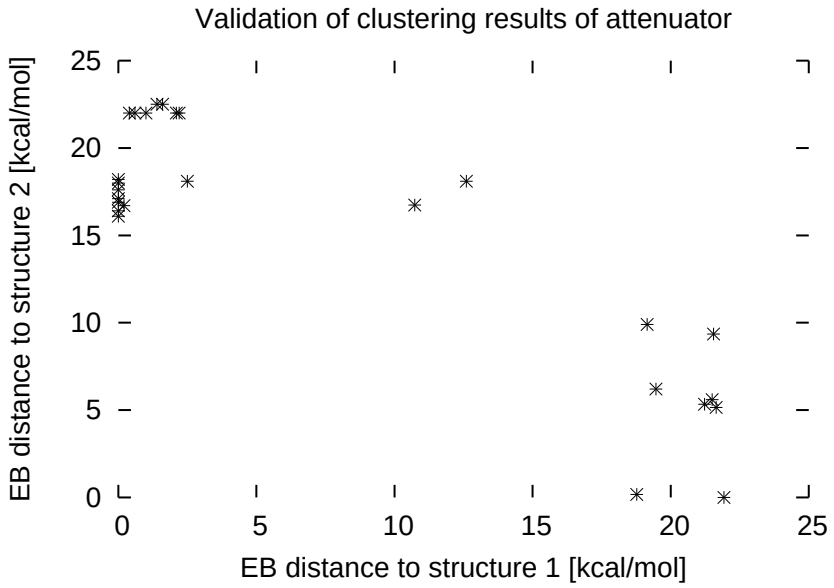
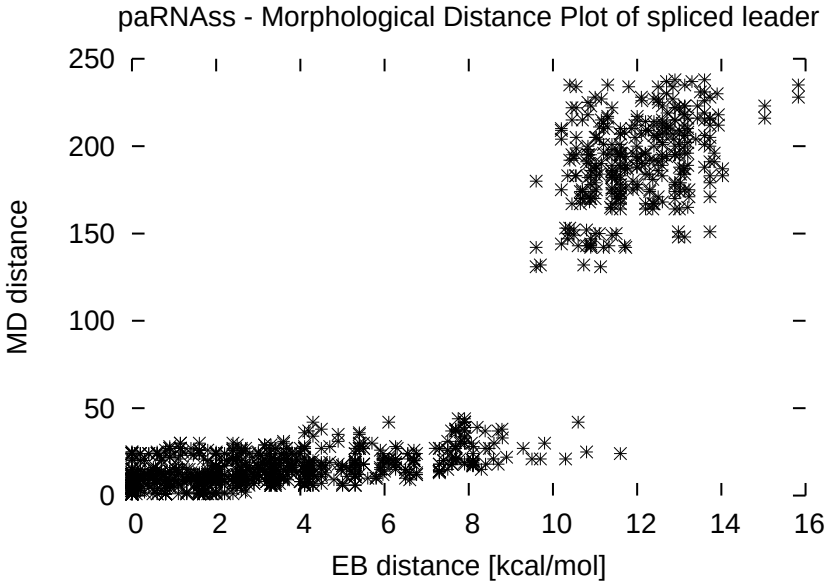


Figure 4.8: Validation plot for pheS-pheT-Attenuator.

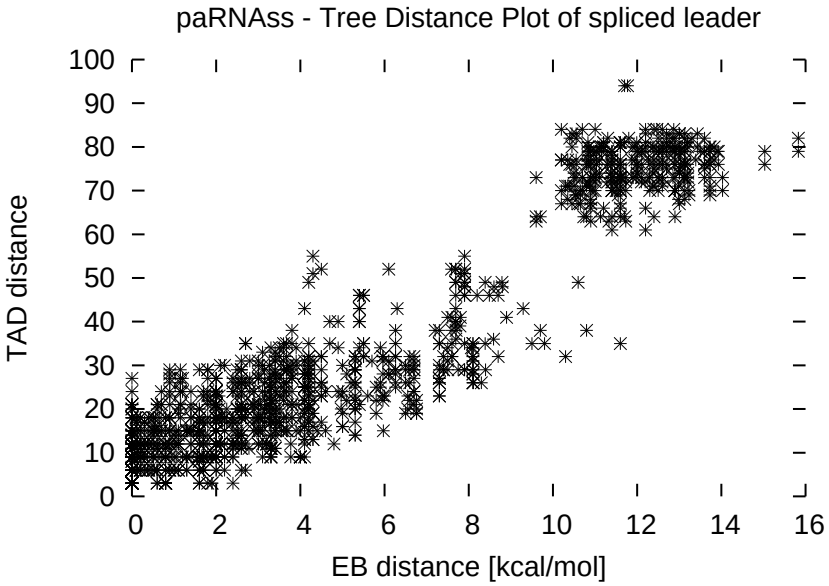
Pairwise Distance Calculation Computation and plotting of the pairwise distances for d_{MD}, d_{TAD} and d_{EB} gave the distance plots shown in Figure 4.9. The plots show two clouds of points that are reasonably separated, especially in the plot for d_{MD}. The poor performance of d_{TAD} in this case may be due to the fact that all structures in the sample comprise only one but not the same hairpin. One part of the structure sample has a hairpin comprising nearly the complete sequence, while in the other part the hairpin is smaller and shifted towards the 3'-end.

Consensus structures Despite the poor performance of d_{TAD}, I used it as well as d_{MD} and d_{EB} for the clustering and consensus calculation step of `paRNAss`. Figure 4.10 depicts the predicted consensus structures (c_1 and c_2). They show good correspondence to the experimentally verified structures (S_1 and S_2, see Figure 4.11). The only difference is the missing small hairpin at the 5'-end of c_2. The reason for this is, that this hairpin is unstable in terms of energy, i.e. it has positive energy of ~ 2 kcal/mol, which prevents its prediction when reapplying the folding algorithm. Although this is enough evidence to substantiate that `paRNAss` succeeded for the Spliced Leader, a validation is still of interest. Since the functions of the alternating conformations are unknown, there remain some doubts on the switching nature of this RNA.

Consensus structure validation Since the analyses for the different distance measures all delivered the same results, the validation has to be performed just once. The outcome is shown in the validation plot in Figure 4.12 and confirms the prediction as a conformational switch. The *pkMeasure* of the consensus structures is 14.

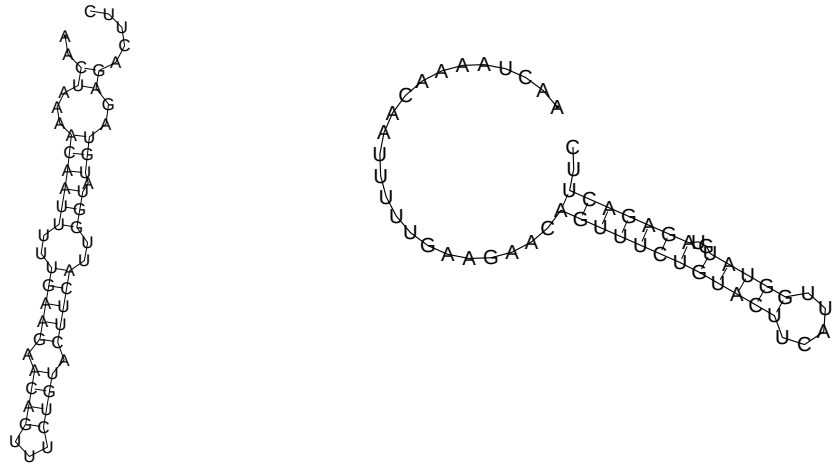


(a) d_{MD}



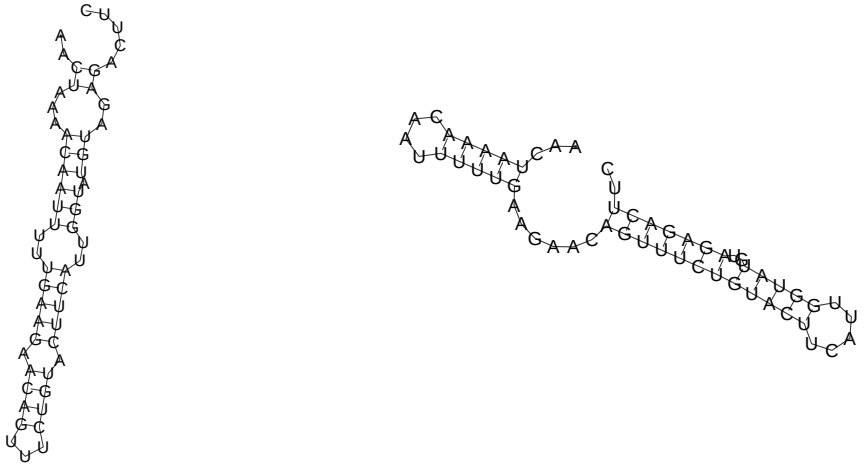
(b) d_{TAD}

Figure 4.9: Distance plots for Spliced Leader.



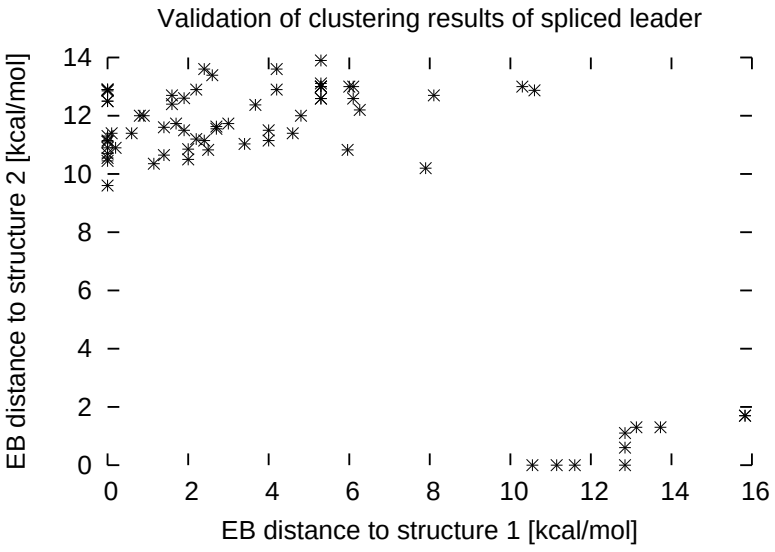
(a) c_1 (b) c_2

Figure 4.10: Consensus structures for Spliced Leader based on d_{MD}, d_{TAD} and d_{EB}.



(a) S_1 (b) S_2

Figure 4.11: Experimentally verified structures (S_1 and S_2) for Spliced Leader.



(a) d_{MD}

Figure 4.12: Validation plot for Spliced Leader.

4.5.3 Tetrahymena group I intron

Function The maturation of pre-RNA involves the excision of introns. In general the excision of introns is performed by small nuclear ribonucleoproteins (snRNPs), such as in mRNA splicing. In the case of pre-rRNA the splicing occurs in the absence of any proteins and could be shown to be performed by the introns themselves. The best characterised member of these self-splicing group I introns is that of *Tetrahymena thermophila*. Recognition of the 5' splice site requires base pairing between nucleotides at the end of the 5' exon and the internal guide sequence (IGS) of the intron. The P1 splice site helix docks into a cleft in the catalytic core of the intron. In pre-rRNAs, containing natural 26S rRNA exon sequences, splice site recognition is inhibited by an alternative hairpin, called P(-1), that is conserved in the mature 26S rRNA. The 3' side of P(-1) involves the same nucleotides of the 5' exon that also base pair with the IGS. As a result, P(-1) directly competes with formation of the P1 splice site helix. This leads to two alternative pre-rRNA conformations: an inactive form that contains P(-1), and an active form that includes P1 and is competent to self-splice [59].

Parameters

- Energy range: $1.6 kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 50

Pairwise Distance Calculation In case of the *Tetrahymena* group I intron, `paRNAss` produced the distance plots displayed in Figure 4.13. Both plots show two clusters of points which are well separated, as it is expected for such an RNA.

Consensus structures The clustering step was performed for all three distance measures. The subsequent calculation of the consensus structures lead in all three cases to the same two conformations shown in Figure 4.14. The differences between structures c_1 and c_2 are mainly at the 3'-end, where either one or three hairpins are present. This resembles the existence of either the P1 (active) or the P(-1) (inactive) hairpin. The correct assignment of the P1 and the P(-1) hairpins to the consensus structures is rather subtle, as I could not find a structure model which also gives the sequence positions of these hairpins. For this reason it is essential to perform the validation step.

Consensus structure validation Pairwise comparison of c_1 and c_2 to all structures in the sample resulted in a validation plot (see Figure 4.15) holding two reasonably separated clouds of points. Combination into a pseudoknot is unlikely as $pkMeasure$ equals 34. Together with the distance plots this validation approves the switching property of the tetrahymena group I intron.

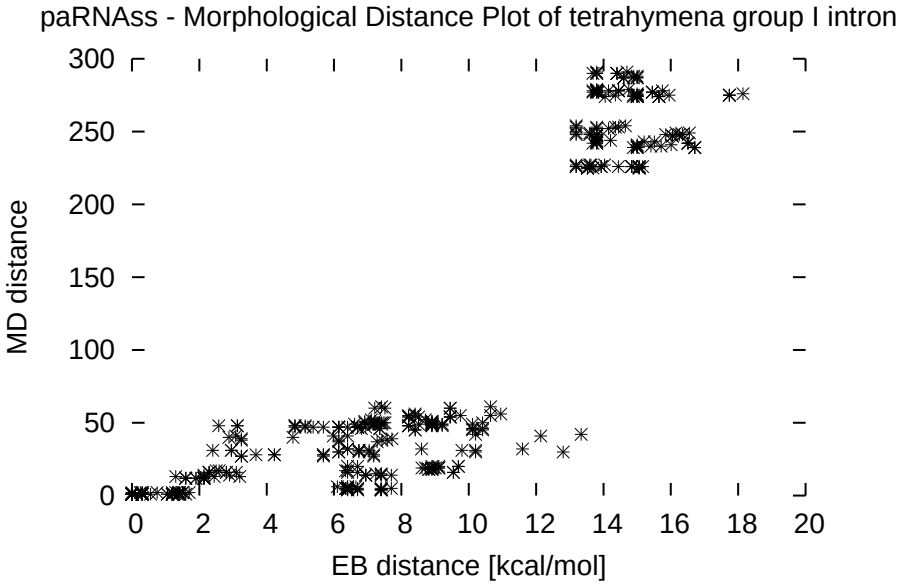
4.5.4 HIV-1 leader

Function The full-length HIV-1 RNA serves both as messenger RNA (mRNA) and as the viral genome. The untranslated leader of this RNA carries several regulatory elements. Their regulatory functions can be roughly subdivided into two groups: regulation of gene expression (transcription, translation, etc.) and virion-associated functions (dimerization, reverse transcription, etc.). Laborious experiments by Huthoff and Berkhout [60] showed that this dual nature goes parallel with two alternating conformations, a branched structure (S_2) and a more stable structure (S_1) which mainly consists of two adjacent helices.

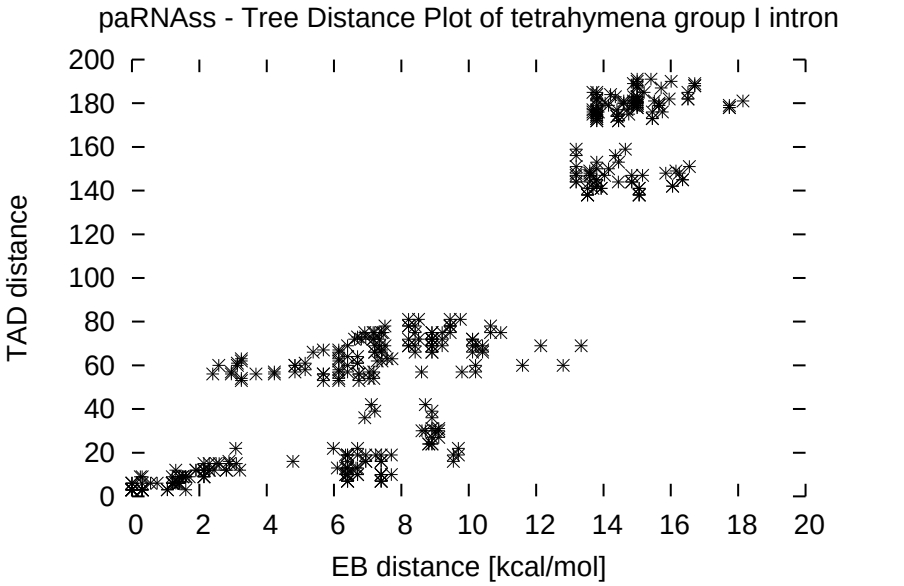
Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 100

Pairwise Distance Calculation Pairwise comparison of all structures in the sample and generation of the distance plots shown in Figure 4.16 reveals the existence of two structurally dissimilar families in the sample. They are well separated in terms of structure (for d_{MD} and d_{TAD}) as well as in terms of energy barrier.

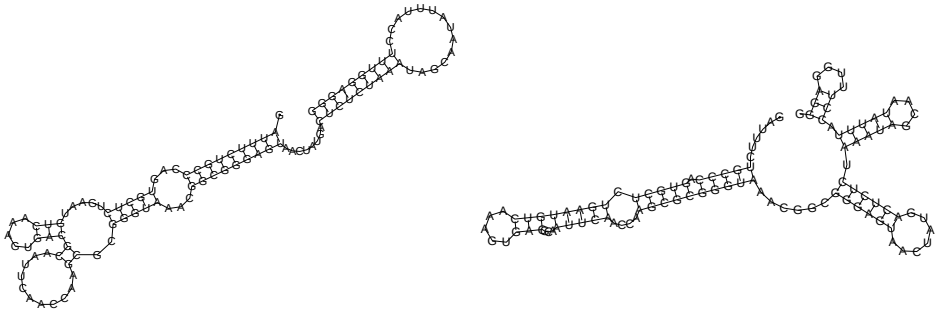


(a) d_{MD}



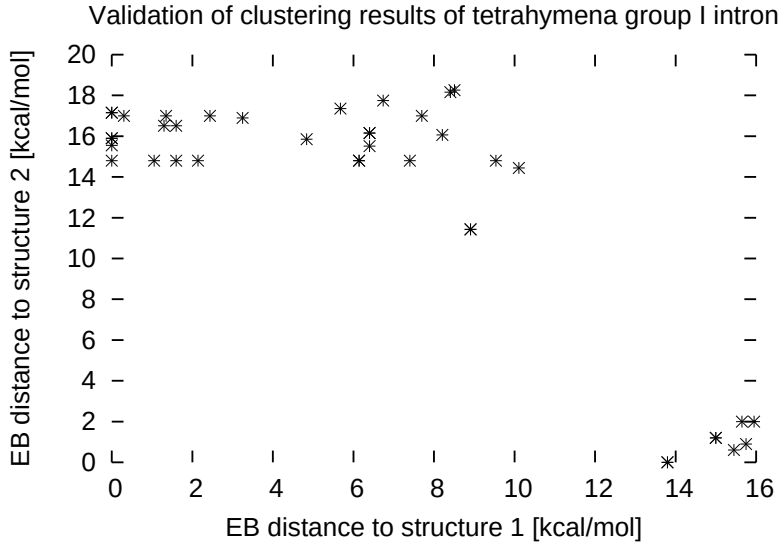
(b) d_{TAD}

Figure 4.13: Distance plots for tetrahymena group I intron.



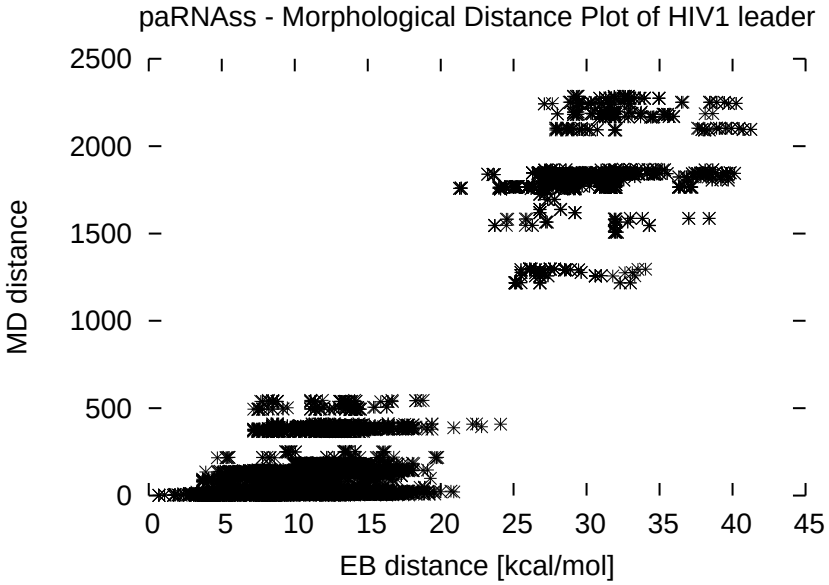
(a) c_1 (b) c_2

Figure 4.14: Consensus structures for tetrahymena group I intron.

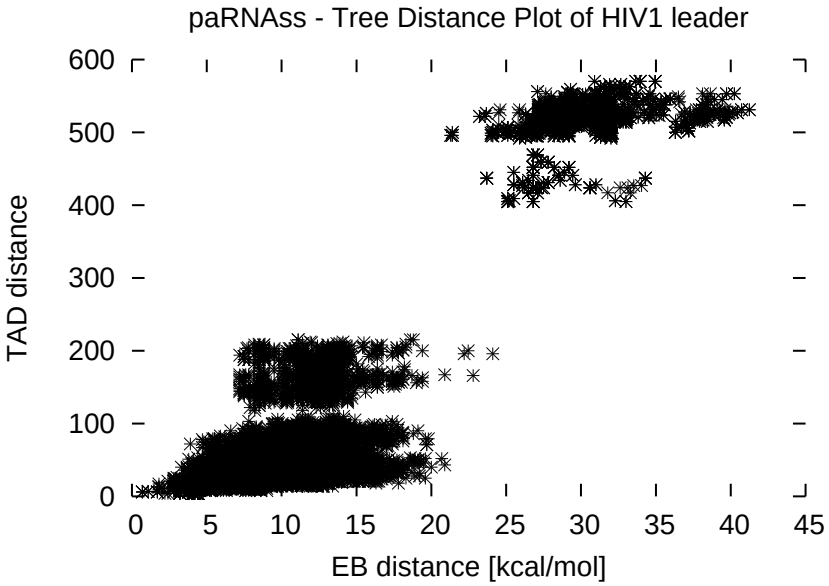


(a) d_{MD}

Figure 4.15: Validation plot for tetrahymena group I intron.

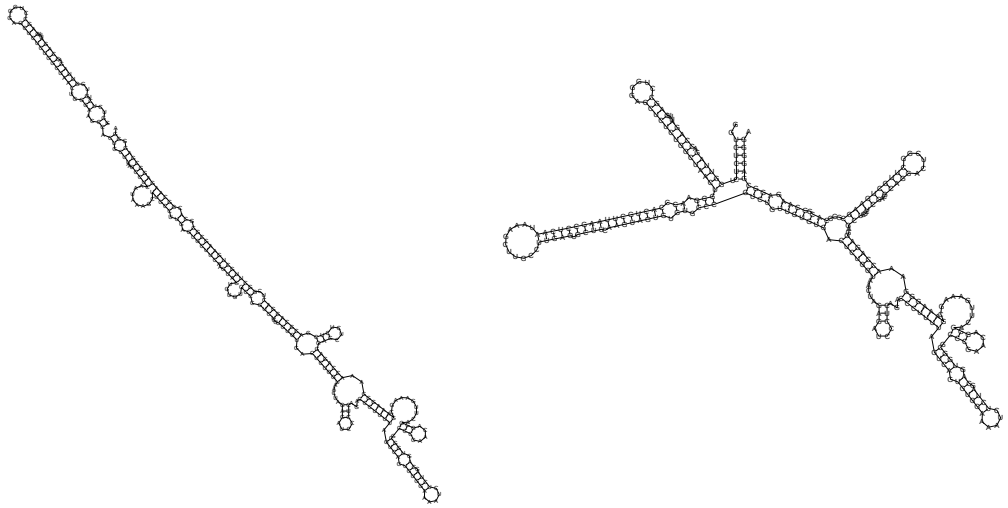


(a) d_{MD}



(b) d_{TAD}

Figure 4.16: Distance plots for HIV1 leader.

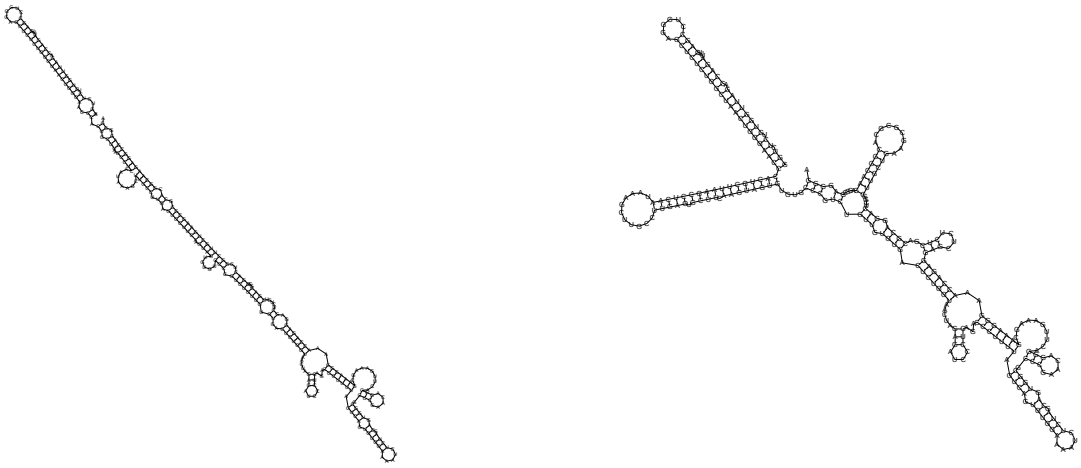


(a) c_1 (b) c_2

Figure 4.17: Consensus structures for HIV1 leader based on d_{MD}.

Consensus structures The good results reflected in the distance plots should give rise to well-defined and rather dissimilar consensus structures. As shown in Figure 4.17 consensus structure c_1 is basically composed of two adjacent helices, whereas c_2 shows a more branched appearance. Comparison of the predicted with the experimentally deduced structures (see Figure 4.18) shows that c_1 is nearly equal to S_1 and that c_2 shows good correspondence to S_2. The differences may arise from artefacts during the consensus structure calculation, e.g. a base pair which is part of one of the native conformations did not appear in the majority of the structures. But they may also lie within experimental artefacts, as experimental data always leaves scope for interpretation by the experimenter. In order to approve the viability of the structures predicted by `paRNAss` I applied the usual validation procedure.

Consensus structure validation The validation of the consensus structures for the leader of HIV-1 resulted in the plot depicted in Figure 4.19. It clearly shows two well-separated clusters, meaning that if a structure of the sample has a high d_{EB} to c_1 it has a low d_{EB} to c_2, and vice versa. Calculation of *pkMeasure* yields 71. Together with the outcome of the previous steps the validation approves the potential of the HIV-1 leader to serve as a conformational switch. Furthermore, the *parnac* is in good correspondence to the experimentally derived structures, notwithstanding the fact that these might be slightly incorrect and the *parnac* correct.



(a) S_1 (b) S_2

Figure 4.18: Experimentally deduced structures (S_1,S_2) for HIV-1 leader.

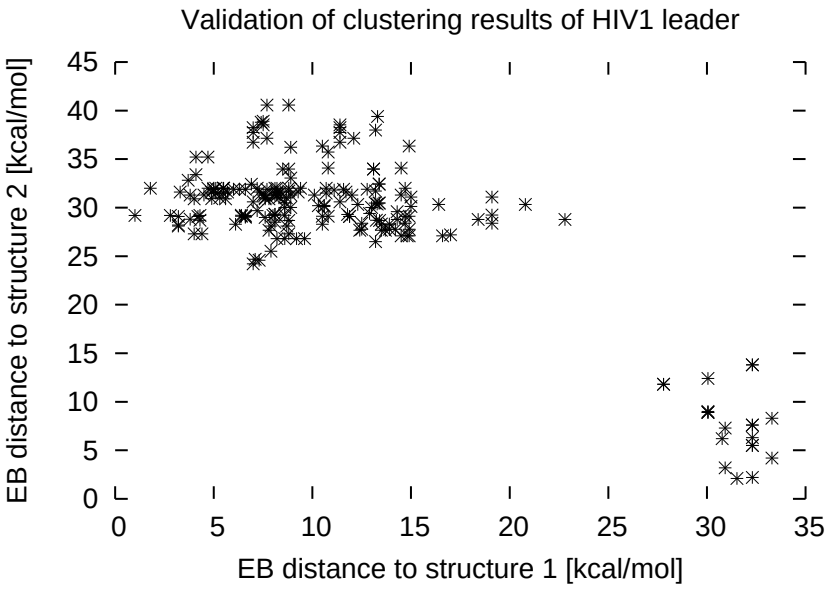


Figure 4.19: Validation plot for HIV1 leader.

4.5.5 E. coli thiM leader

Function The 5'-UTR of *E. coli thiM*, which encodes for a protein of the thiamine biosynthetic pathway, holds a member of the class of metabolite sensing riboswitches. Winkler et al. [51] showed that *thiM* translational fusion constructs exhibit thiamine-dependent suppression, whereas transcriptional fusions do not. A structure-probing process revealed that *thiM* leader RNA undergoes structure modulation upon binding of specific ligands, such as thiamine and thiamine pyrophosphate (TPP). The authors propose a mechanism where TPP-binding to the RNA leads to the sequestration of the SD sequence, which is required to be unpaired for efficient translation in prokaryotes. Upon thiamine starvation the RNA gets freed and a structural rearrangement leads to a conformation in which the SD sequence is unpaired. This enables ribosome entry and translation of the protein which is required for thiamine biosynthesis.

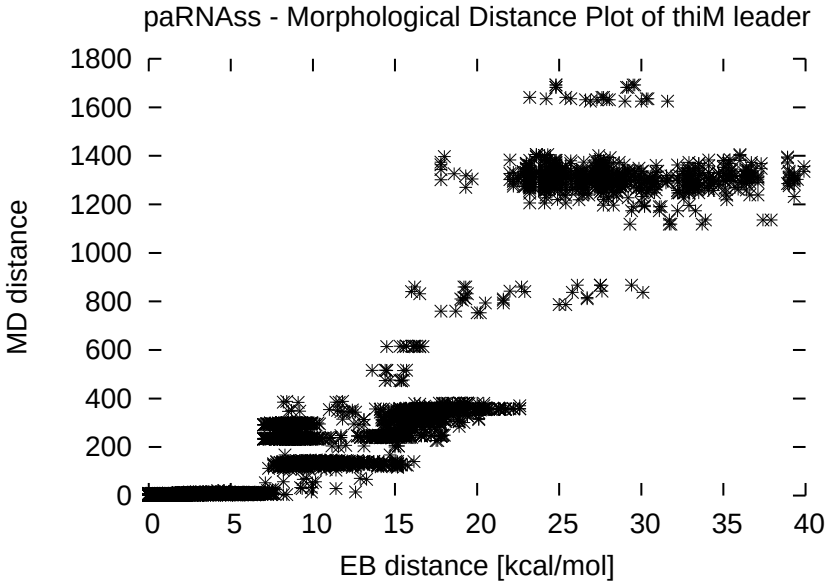
Parameters

- Energy range: $2.5 kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 100

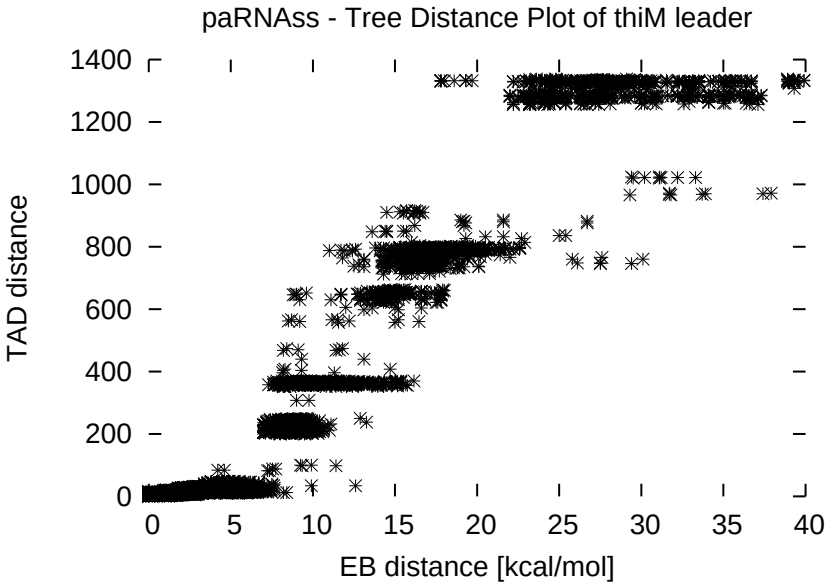
Pairwise Distance Calculation The analysis based on pairwise distance calculation does not deliver the preferred result of two separated clouds in the distance plots (see Figure 4.20). Nevertheless, there are two major clusters which show reasonable separation, especially based on d_{MD}, and the points in-between might be due to artefacts of the distance measures. A prove for this could be achieved if the validation of the consensus structures delivers a positive result.

Consensus structures As the analysis of d_{MD} delivered better results I used this for the subsequent steps. Clustering and consensus structure calculation yielded the two conformations shown in Figure 4.21. From these only c_1 is in correspondence to one of the published conformations. A more detailed analysis of the published experiments showed that these results could also be explained with the conformations predicted by `paRNAss`. Furthermore, the energy difference of the original conformations is in the range of 10 to 15 kcal/mol, whereas the newly predicted structures differ approx. 2 kcal/mol in free energy. Mr. Breaker, the corresponding author of the publication on the characterisation of thiM leader, agrees that their predicted conformations are just models and that the results reported here could as well be correct (personal communication).

Consensus structure validation In order to get further evidence for the plausibility of the *parnac* the validation step was performed. Calculation of $pkMeasure$ resulted in 55. The validation plot in Figure 4.22 shows that it is appropriate to presume that the

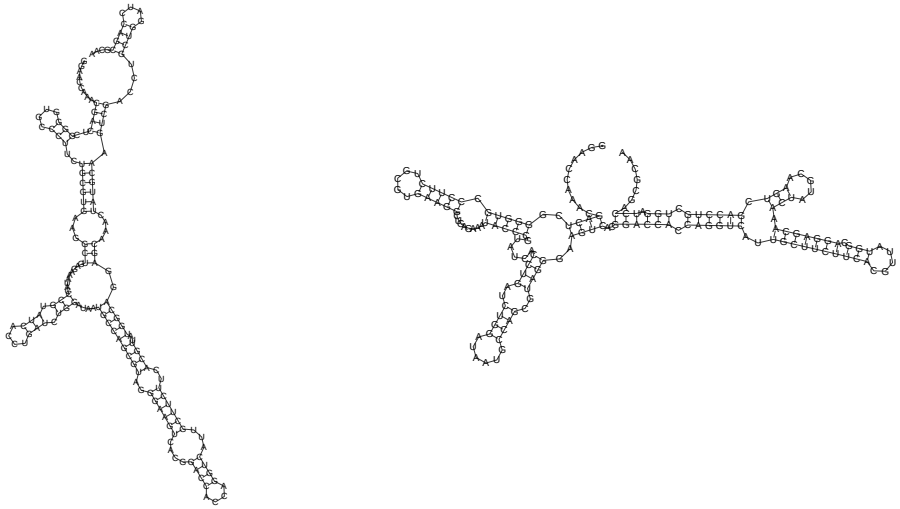


(a) d_{MD}



(b) d_{TAD}

Figure 4.20: Distance plots for thiM leader.



(a) c_1 (b) c_2

Figure 4.21: Consensus structures for thiM leader based on d_{MD}.

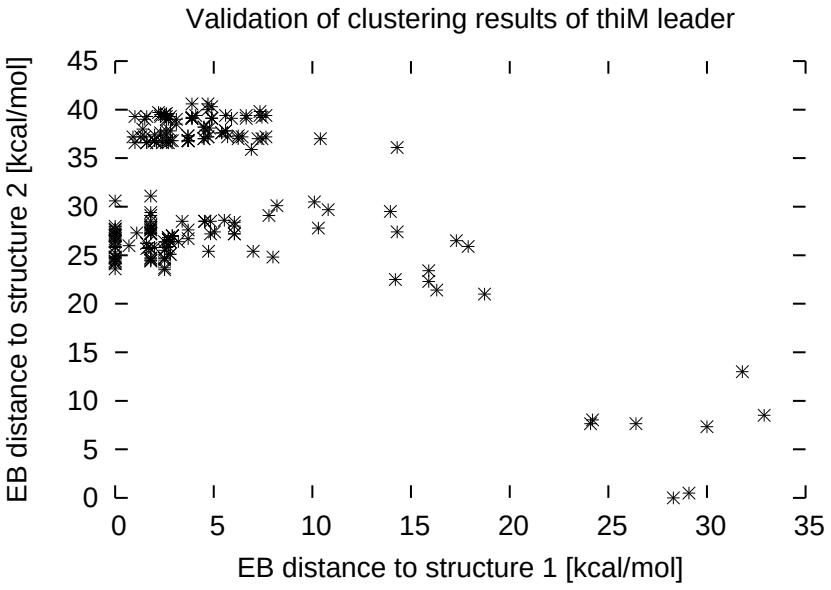
parnac represent the two positions of the thiM leader riboswitch. Despite the fact that the analysis was only successful using d_{MD} I think it is adequate to call this a successful analysis. At least it was able to reveal an alternative set of conformations that could resemble the alternating structures of this switch.

4.5.6 S-box leader of B. subtilis metE

Function In addition to riboswitches regulating vitamin biosynthesis, Epshtein et al. [53] showed that riboswitches are also involved in control of amino acid biosynthesis, especially methionine. At least 11 transcription units in *B. subtilis*, that are mostly involved in cysteine and methionine synthesis, possess a leader element that includes an intrinsic transcription terminator, competing anti-terminator, and a conserved element (S-box) that functions as an anti-antiterminator. S-adenosyl-methionine directly binds S-box RNA to stabilise its anti-antitermination structure, thus causing termination of the leader transcript.

Parameters

- Energy range: $5 kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50



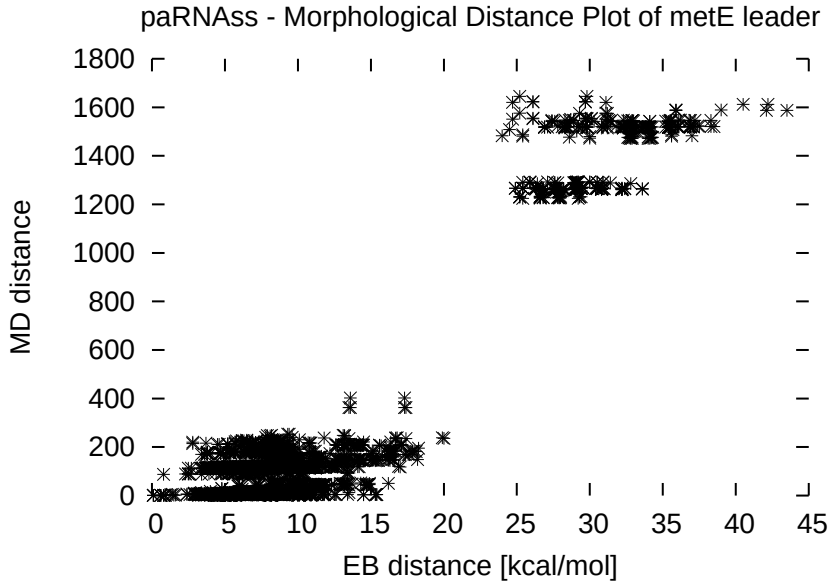
(a) d_{MD}

Figure 4.22: Validation plot for thiM leader.

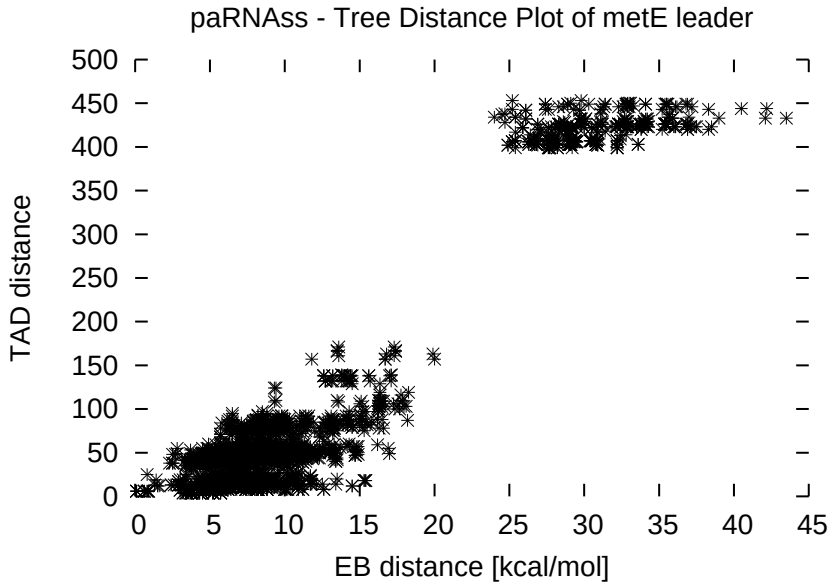
Pairwise Distance Calculation The distance plots in Figure 4.23 show the best separation I ever encountered using `paRNAss`. The structure sample seems to hold structures of two families that are clearly separated by a large energy barrier and are structurally very dissimilar.

Consensus structures The clustering and consensus structure calculation step based on either d_{MD}, d_{TAD} or d_{EB} each delivered the same consensus structures which are given in Figure 4.24. They show significant structural dissimilarity as it is expected from the distances obtained by d_{MD} and d_{TAD}. The problem is that the terminator hairpin at the 3'-end is present in both predicted consensus structures, so that these conformations do not resemble the active and inactive positions of the switch. Nevertheless, such striking results imply a switching capability between these conformations so I continued the `paRNAss` analysis.

Consensus structure validation Up to now the analysis clearly proposes a conformational switch and this is also strongly supported by the results of the validation shown in Figure 4.25. The validation plot as well as a *pkMeasure* of 32 confirm the plausibility of the *parnac*. As both conformations carry the terminator hairpin, the *parnac* does not correspond to the published model. A closer look at the published conformations states two problems: (1) The two proposed conformations differ to about 10 kcal/mol in energy, which could of course be the energy that SAM delivers upon binding, but (2) SAM binding leads to formation of the energetic better conformation which would thereby gain even more energy. This would enlarge the energy difference of the confor-

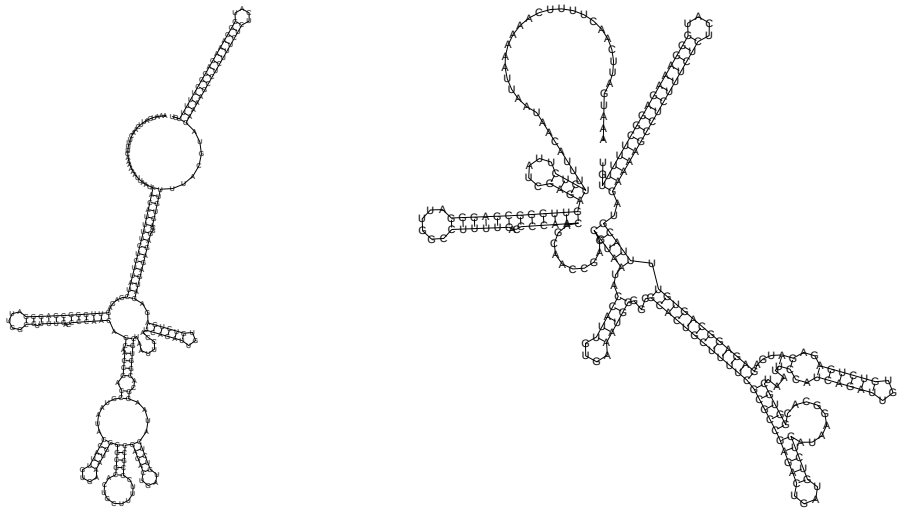


(a) d_{MD}



(b) d_{TAD}

Figure 4.23: Distance plots for metE leader.



(a) c_1 (b) c_2

Figure 4.24: Consensus structures for metE leader.

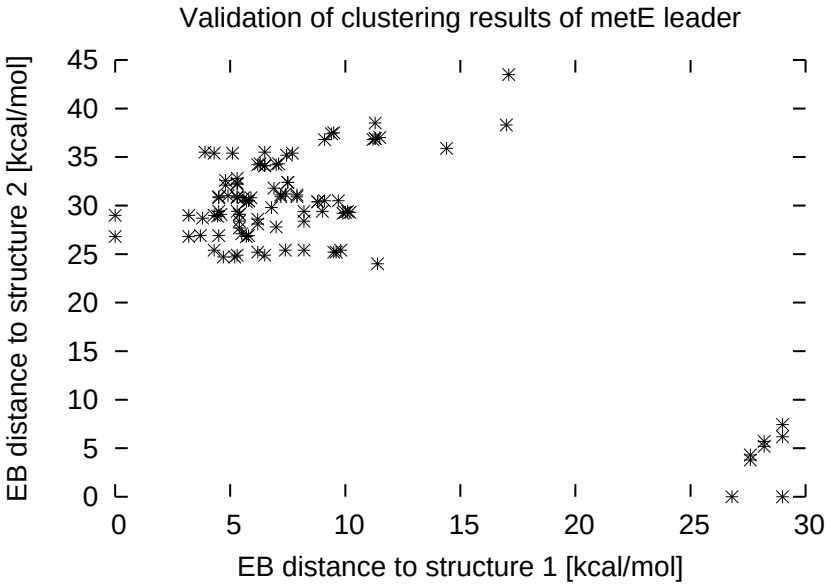


Figure 4.25: Validation plot for metE leader.

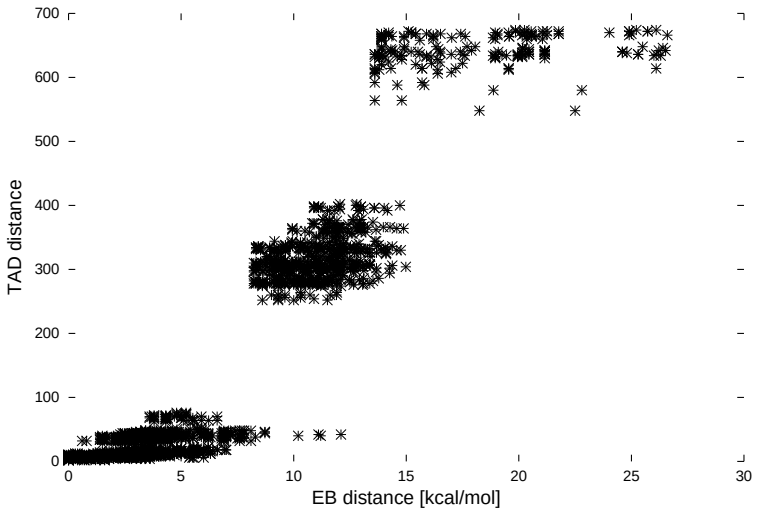
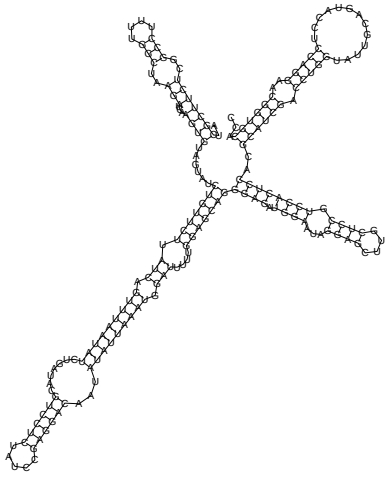


Figure 4.26: Distance plot based on tree alignment and energy barrier distance for human U2 snRNA, showing three separate clouds of points.

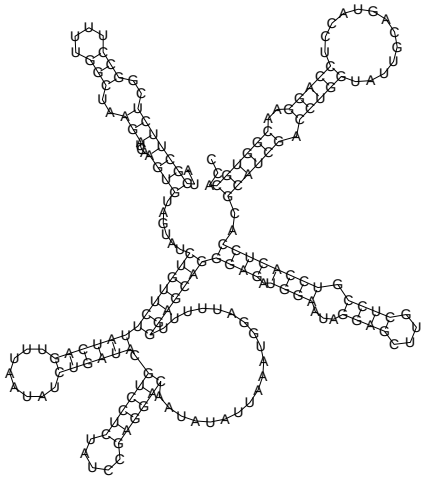
mations additionally. I cannot disprove that the published structures are correct, but I can say that `paRNAss` revealed a conformational switch for this RNA. In my opinion the *parnac* shows the local structural rearrangement in the S-Box which occurs upon binding of S-adenosyl-methionine (SAM).

4.5.7 Human small nuclear RNAs

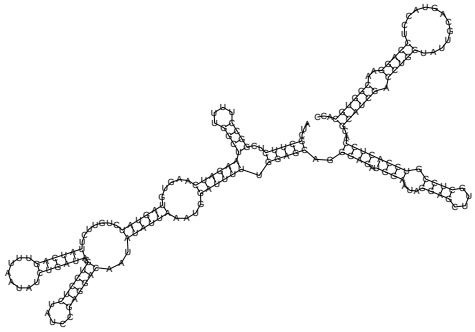
In general the focus of `paRNAss` is on the prediction of structural RNA switching. Other approaches exist to reveal more general properties of the energy landscape of an RNA. Kitagawa et al. [10] studied the conformational landscape of human snRNAs with the method described in Section 2.4.2. I reviewed their analyses of the human spliceosomal snRNAs using the new version of `paRNAss`. The parameters were $P = 5\ kcal/mol$, $F = canonicals(refolded)$, $T = 37$°C and $N = 50$. The experiments on U1, U4, U5 and U6 snRNA delivered comparable results, whereas for U2 snRNA the distance plot (d_{EB}, d_{TAD}) shows three separate clouds of points, as displayed in Figure 4.26. Since each point in the distance plot corresponds to a pair of structures, the appearance of three distinct clouds must be due to three, four or even more families of structures in the sample set. A look at the structures in the sample set revealed three families. Consensus structures of these three families are shown in Figure 4.27. This result is in clear contrast to the result reported by Kitagawa *et al.*, as they propose a "uni-valley" shape of the folding space and therefore only one prominent structure. Possible reasons for this discrepancy may lie with the use of `MFOLD`, producing a heuristic subset of all feasible structures, and the coarse grained "tree representation distance".



(a) c_1, $E = -69.1\ kcal/mol$



(b) c_2, $E = -68.0\ kcal/mol$



(c) c_3, $E = -67.3\ kcal/mol$

Figure 4.27: Consensus structures of the three structure families present in the sample set of the folding space of human U2 snRNA.

4.6 Evaluation of the paRNAss approach

The analyses of Section 4.5 and those presented in Appendix A are very promising, but to be sure about the reliability of `paRNAss` I performed an evaluation procedure based on these examples and additionally included sequences from coding and non-coding regions as well as random sequences. This resulted in the following design: In addition to the published structural RNA switches, a set of 1500 sequences, composed of randomised and native sequences from different sources, was analysed with `paRNAss`. For the randomisation, the percentage of bases of the source sequence mentioned below was changed randomly, allowing an overall deviation in GC content of 5% compared to the source sequence. For each randomisation percentage 30 sequences were produced. As the randomisation should resemble mutational events, and was not performed to test free energy distributions, I did not have to preserve the dinucleotide distribution [61]. In order to get an idea about the impact of prior knowledge on the results, I performed a so called blindtest.

Experiment 1: True positives and false negatives The analyses for this set are shown in detail in Section 4.5 and Appendix A. A summary of the results is shown in Table 4.1. For 16 sequences I successfully revealed a switch by using `paRNAss`. In one case (*E. coli* hok mRNA) the validation results were negative, which may be due to the fact that this switch is triggered by sequence truncation. Altogether, using `paRNAss` I was able to predict 17 out of 21 switches. The switches I failed to recognise, were either bearing pseudoknots (AMV in one structure, *E. coli* α operon mRNA in both) or long distance interactions (LDI), see Table 4.1. The used RNA folding programs are not able to calculate the secondary structure with pseudoknots. Nevertheless, I succeeded in two cases (*E. coli* S15 mRNA and HDV ribozyme), as the pseudoknot conformation shows sufficient dissimilarity even when approximating the pseudoknot by an un-knotted structure. Long distance interactions are difficult to handle with `paRNAss`, because the large contribution of the switching part to the overall distance gets shadowed by numerous small contributions of non-switching parts, thus preventing the outcome of two distinct families in Step 1 of the `paRNAss` approach.

Experiment 2: Switch degradation To get an idea of the stability of switches under mutations (selection pressure), the sequences of the pheS-pheT-operon attenuator from *E. coli* and the Spliced Leader from *Leptomonas collosoma* were randomised 1%, 3%, 4%, 5%, 10%, 15%, 20%, 25%, 30% and 35%. Table 4.2 summarises the outcome of the measurements for the randomised switch sequences. Even the introduction of 1 change (less than 2% rand.) in the sequences is capable of destroying the property of being a switch. Furthermore, the results clearly show that increasing randomisation leads to decreasing amount of predicted switches, as well as rare positive predictions for higher randomisation rates.

Table 4.1: Experiment 1: Summary of the results for known switches. (nt = nucleotides, LDI = long distance interaction involved, pk = pseudoknot present in at least one structure, mRNA = messenger RNA, uRNA = untranslated RNA)

Sequence	nt	type	pk	LDI	prediction
Attenuator	73	mRNA	-	-	+
Spliced Leader	56	uRNA	-	-	+
E. coli DsrA [62]	87	uRNA	-	-	+
E. coli S15	74	mRNA	+	-	+
5'-UTR of *E.coli* btuB mRNA	203	uRNA	-	-	+
5'-UTR of MS2 RNA genome	73	uRNA	-	-	+
Tetrahymena group I intron	108	ribozyme	-	-	+
HDV ribozyme	154	ribozyme	+	-	+
T4 td gene intron	164	uRNA	-	+	+
HIV-1 leader	281	uRNA	-	+	+
HIV-2 leader	390	uRNA	-	+	-
3'-UTR of AMV RNA	145	uRNA	+	-	-
E. coli α operon	140	mRNA	+	-	-
E. coli hok	143	mRNA	-	+	+/-
E. coli thiM leader	165	uRNA	+	-	+
B. subtilis metE leader	247	uRNA	-	+	+
B. subtilis ribD leader	305	uRNA	-	-	+
B. subtilis ypaA leader	344	uRNA	-	+	-
E. coli lysC leader	197	uRNA	-	+	+
S. cerevisiae ITS2	236	uRNA	-	+	+
B. subtilis ptsGHI leader	94	uRNA	-	-	+

Experiment 3: Arbitrary natural sequences, random sequences, human 3'-UTRs A set of native and randomised mRNA sequences was analysed in comparison to sequences from human 3'-UTRs. As well as random sequences, mRNAs, especially the coding regions, are assumed to rarely possess regulatory function, whereas 3'-UTRs are proven to be important regulatory regions. Since switches are regulatory elements, they are supposed to occur more often in such regulatory regions. I restricted the length of the analysed sequences to 80 nucleotides, due to the facts that numerous known switches are in this length range and that the used approach performs well, meaning that the produced plots are easily interpretable in the positive as well as in the negative case.

The set of "arbitrary" natural sequences was build of 110 sequences, corresponding to parts (80 nt) of 26 *E. coli* mRNAs. Therefore each mRNA was cut into pieces of length 80 and 4-6 of these were chosen (equally distributed over the whole sequence). Exon 2 of the *Caenorhabditis elegans* cAMP-dependent protein kinase catalytic subunit C gene (M37114) and bases 1596 to 1675 of the *Drosophila melanogaster* egghead gene (NM_080313) mRNA were randomised to form the test set of random sequences.

Table 4.2: Experiment 2: Switch degradation. Number of positive predictions in the set of 30 sequences for each randomisation step. For the Spliced Leader the number of changes for 3% and 4% randomisation both round to two mutations, so this corresponds to one experiment.

% Randomisation	0	1	3	4	5	10	15	20	25	30	35
Attenuator (73 nt)	30	2	0	1	1	1	1	0	1	0	0
Spliced Leader (56 nt)	30	18	10		5	0	2	0	0	1	0

Table 4.3: Experiment 3: Results for arbitrary sequences, random sequences and human 3'-UTRs. (sequence length = 80 nt)

Source	Nr. of sequences	pred. *parnac*	% positives
E. coli mRNAs	110	1	**0.9**
Random sequences	420	6	**1.4**
Human 3'-UTRs	400	14	**3.5**

For the set of human 3'-UTRs, the first 80 bases of the first 400 entries in the human 3'-UTR section of the UTRdb (version of 2003/02/12, [63]) were extracted.

The results of the analyses under the above mentioned conditions using `paRNAss` are summarised in Table 4.3. One *parnac* was predicted in the set of 110 parts of *E. coli* mRNAs. This means that a *parnac* is predicted with a frequency of 0.9%. For the set of random sequences derived from natural sequences, there were six *parnac* predictions in 420 sequences, leading to a *parnac* frequency of 1.4 %. The analysis of the 400 parts of human 3'-UTRs led to fourteen *parnac* predictions. This corresponds to a *parnac* prediction rate of 3.5%.

Experiment 4: Blind test In the case of proven conformational switches I knew that I had to get a positive prediction and tried different sets of parameters to get the desired results. In order to get an idea about the impact of this prior knowledge I set up a blind test together with Marc Rehmsmeier. For this purpose I gave each of the published switch sequences to Marc, who shuffled each of them preserving the dinucleotide distribution and mixed these shuffled with the original sequences. I received this set of sequences numbered starting at zero, so that I was unable to identify the switch sequences by their name. All of these sequences were analysed with `paRNAss` using different parameters. The result of each analysis was simply the classification as switch or non-switch. The list of these results was then compared to the list assigning each sequence its origin and therefore the property of being a switch or a non-switch. The results were rather disappointing as I only achieved 50% correct predictions (true positives and true negatives) and a p-value of 0.57. This means that the combination `paRNAss` and me is only slightly better than chance when predicting conformational switches in unknown sequences.

```
0        1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890123
AUCCAGGAGGCUAGCGCGUGAGAAGAGAAACGGAAAACAGCGCCUGAAAGCCUCCCAGUGGAGGCUUUUUUUG
...((((..(((..(.(((..........))))).....))).)))).(((((((....)))))))......(1)
.((((((((((.((((((................)).))))).....)))))))...))))..........(2)
```

Figure 4.28: Alternate structures for Attenuator.

4.7 Discussion

The applications in Section 4.5 and the results of the evaluation show that the strategy used by `paRNAss` enables a researcher to reveal conformational switching in RNA. Difficulties arise when pseudoknots or LDIs are involved in one or in both alternate conformations. Current available standard RNA folding programs are not able to calculate the secondary structure with pseudoknots because of computational complexity. Some algorithms exist, which are capable of computing specific types of pseudoknots [64, 65] and could be used to fill the gap. In some cases, e.g. ypaA and ribD, the free energies of the published conformations are $6-10$ kcal/mol above the mfe. Since the number of suboptimal structures for these sequences can only be calculated up to 4 kcal/mol due to computational limitations, these conformations are inaccessible.

Degradation of a switch by mutational events is very fast. The introduction of one nucleotide change is capable of destroying the property of being a switch, notwithstanding the fact that its conformations may persist in the space of suboptimal structures [44]. This is in accordance with a comparable analysis by Schuster et al. [66], where it was shown that there is some probability that even a single mutation substantially alters the secondary structure. Furthermore, the authors state that sequences of length 100 with Hamming distances greater than 3 (corresponding to 3% randomisation) are very unlikely to have identical or closely related structures. In this experiment the switching property gets almost completely lost for 1% (Attenuator), resp. 5% (Spliced Leader), being in the same range as the before stated value. The reason for the immediate appearance of this effect in the case of the Attenuator can be seen when taking a look at the two alternate structures of the native sequence in Figure 4.28. Together, they span about 90% of the sequence, so each mutation is very likely to affect at least one structure. Furthermore, the two structures share 80% of the bases, so both of them are affected in most cases. To give an example: Changing the G-7 to a C and eliminating the base pairs G-7 was involved in, alters the free energy values as follows: The free energy of structure 1 increases from -21.20 kcal/mol to -21.03 kcal/mol, whereas structure 2 loses -7.1 kcal/mol, increasing the free energy from -20.93 kcal/mol to -13.83 kcal/mol. The corresponding structure is still in the folding space and it is still the member with the lowest free energy in its family, but it now falls beyond the threshold of 3 kcal/mol. This seems to be a very drastic example, but shows one possible reason for the observed volatility. Generally, I assume that several mechanisms are accountable for these results: (1) One structure looses free energy and gets lost in the structure space

Table 4.4: Accession numbers of the human 3'-UTRs with positive *parnac* predictions.

3HSA016018	3HSA020013	3HSA012599
3HSA014833	3HSA020034	3HSA012648
3HSA020049	3HSA012739	3HSA012760
3HSA028980	3HSA016147	3HSA012856
3HSA012868	3HSA012922	

(see example above); (2) One of the two prominent structures gains energy, kicking the other out of the threshold; (3) A new structure, which was buried in the suboptimals, gets prominent, revealing a third local minimum. Another fact is, that the Attenuator is a conformational switch as well as a coding RNA. For that reason selective pressure is present in two different ways, and the evolutive possibilities to improve the switch are restricted. In the case of the Spliced Leader, these mechanisms seem to intervene more smoothly. Additionally, the structures contain larger 5' unpaired regions, which could serve as buffers for mutational events. The few *parnac*s that are predicted for higher randomisation rates imply a small possibility to obtain a conformational switch by chance.

Regarding the predictions for the set of mRNAs and random sequences as false positive predictions, implies an overall error rate of 1.3%. This must be seen as an upper bound, as I cannot exclude the possibility that some are true (but unproven) positives. In the case of the human 3'-UTRs the frequency of *parnac* predictions is 3.5%, which is substantially (2.7-fold) higher. Since these regions often possess regulatory function, I assume that these predictions are good candidates for structural RNA switches and that this should encourage experimenters to check for correctness of these *parnac*s (see Table 4.4).

The dampening results of the blind test, where I was not much better than chance, need a closer look. The hardest conclusion from this experiment might be, that the predictions obtained with the help of `paRNAss` are not significant. This is too strong, as it could be, that I was too restrictive and another person would have achieved better results. Additionally, it could be that I did not try enough parameter combinations and that spending more time on this would have improved the results. Other uncertainties lie in the assignment of the shuffled sequences to be non-switches, since these might possess the (unproven) ability to serve as a conformational switch, and in the small sample size the test was based on. Major conclusions from this experiment are that the parameters have a high impact on `paRNAss`, that the results of a `paRNAss` analysis have to be investigated thoroughly and that this interpretation step is susceptible to errors. Therefore, it might not be a good idea to use `paRNAss` for searching conformational switches in large sequences, but it could serve as an early test when a researcher has some evidence that an unknown sequence might be a conformational switch. A method to search for RNA switches in genomic sequences using less parameters might be based on the approach of abstract shapes, which I will present in the next chapter. `paRNAss` could then be used to validate those predictions.

The `paRNAss` results for the human U2 snRNA (Section 4.5.7) document the use of `paRNAss` to reveal more than two competing structures. The results differ from those of

Kitagawa et al. [10] and are contradictory to the biological need for a certain structure of the U2 snRNA. But, one has to bear in mind that the native structure of an RNA might get stabilised by additional factors, like proteins and other RNAs. Additionally, the three consensus structures show reasonable dissimilarity so that I am convinced that the `paRNAss` results are closer to reality than the uni-valley profile proposed by Kitagawa *et al.*.

`paRNAss` is better suited for the prediction of conformational switching than the approach based on `SFOLD` (Section 3.4.1), as this one checks for morphological dissimilarity only and does not take into account that the conformations should be separated by a reasonable energy barrier.

The small number of proven conformational switches can, on the one side, be seen as a drawback to my evaluation procedure, but, on the other side, I think that this expresses the explicit need for a bioinformatics approach in this area of research. `paRNAss` is such an approach and I am convinced that it will be helpful to reveal new conformational RNA switches, and in turn gets improved based on these findings. Further improvements to `paRNAss` can result from new distance measures, more reliable algorithms for structure prediction and methods to make the interpretation of the results easier or even unnecessary by some kind of scoring. I tried to develop a score based on measures for the quality of the clustering, but the results were not conclusive. At the same time Robert Giegerich developed the idea of abstract shapes of RNA, which in my point of view has a high impact on `paRNAss`. Therefore, I decided to work on this first and to postpone the `paRNAss` scoring task to my future work.

CHAPTER 5

Abstract Shapes of RNA

The structure space of an RNA is due to its complexity difficult to imagine and due to its size hard to analyse entirely. Several approaches (see Sections 2.4 and 3.4.3) exist to accomplish this problem. A new approach based on structural abstraction will be introduced in the next section. It enables the user to get a holistic view of the structure space and to extract the "big players", either based on energy or on probability obtained from the partition function. Furthermore, I show applications that point at the possibility to infer structural well-definedness and alternative secondary structures. This work was done in cooperation with Robert Giegerich, who had the initial idea and defined what abstract shapes are. My part was the implementation, testing and evaluation of the method.

5.1 Drawbacks of Current Approaches to RNA Secondary Structure Prediction

In the previous chapter I presented the `paRNAss` approach for the prediction of conformational switching in RNA. It is based on the analyses of the structure space of an RNA and performs pairwise comparisons of suboptimal structures to elucidate the potential of serving as a conformational switch. Hence, the computational effort of this step of the `paRNAss` approach is quadratic in the number of structures that have to be analysed. The problem that emerges is, that the number of suboptimal structures grows exponential with increasing sequence length, even when restricting to a moderate energy range. Therefore, the `paRNAss` approach incorporates a sampling step, which chooses a given number of suboptimal structures equally distributed over the set of suboptimal solutions. This sampling is not free from artefacts, as it might miss entire families. Hence, it is favourable to reduce the number of structures per family to ensure that a member of each family is sampled, or no sampling is needed at all.

Another disadvantage of current RNA folding algorithms is that the computed *mfe*-structure may differ from the native one. For example, for some tRNAs the computed mfe-structure is not the expected "cloverleaf" but a single long hairpin. Looking at subop-

timal structures for these tRNAs reveals the cloverleaf further down in the energy sorted list of suboptimal solutions. For tRNAs this does not state a problem, but for structure elucidation of RNAs with unknown structure it is very laborious to search through the list of possibly 100,000 suboptimal structures to find the one being in correspondence with prior knowledge or experimental data.

Taking a closer look at the list of suboptimal structures reveals a general feature: The structure space contains a large number of similar structures, differing only in a few base pairs or the position of bulges and internal loops. In most cases the researcher focuses on dissimilar structures and is not interested in those of high similarity.

Dissimilarity can be inferred using distance measures, like the ones described in Section 3.2. This always results in a biased sample due to the distance threshold chosen. One could further define dissimilarity as the difference in structural elements that form a structure. Such coarse graining was already mentioned in Section 3.1.2. The basic idea is to switch from the representation of individual base pairs and unpaired bases to the representation of structural elements, such as hairpin loops, bulge loops, internal loops and multiloops. This does not only infer dissimilarity of structures, but also allows for the partitioning of the folding space into different classes of structures, sharing the same structural elements, i.e. having the same shape.

5.2 Defining Abstract Shapes

Classes of structures can be defined in many ways, but a few requirements seem appropriate to catch the intuition of a "shape": When we feel (either intuitively or in some formal sense) that two structures are similar, they should either have the same shape, or their shapes should be similar in the same sense. Within each abstract shape, a concrete structure is designated as its representative, such that looking at all the representatives gives a meaningful overview of what is there in the folding space. Furthermore, each abstract shape should also have an explicit representation, that is not a concrete structure and independent of primary sequence.

The domain of sequences is closed under juxtaposition – concatenating sequences s and t, we obtain the sequence st. The same holds for structures – if x and y are hairpins, and we paste the 3' end of x to the 5' end of y, we obtain a structure which is simply an external loop with two adjacent hairpin structures. In addition to juxtaposition, structures are formed by recursive embedding. For example, implanting three adjacent hairpins into the loop of a fourth, we obtain a cloverleaf structure. Being formed by juxtaposition and embedding, structures are inherently tree-like. Although they can be represented in many ways – such as strings, squiggle plots, or base pair lists – a tree representation is the one that can be used for all purposes without introducing artefacts or losing explicit information.

In data type theory, this is called an initial data type: There is a simple mapping from the initial data type to any other representation, while an inverse mapping may be more complicated or may not even exist.

Shapes should be homomorphic images of structures, which means that when structure x is embedded in structure y, then the shape of x is also embedded in the shape of y. For this reason, the principles of juxtaposition and embedding must apply to the shape domain as well. Before going into technical detail, we arrive at the following definitions:

Definition 5.2.1 *Let $\mathcal{S}$ be the tree-like domain of structures, and $\mathcal{P}$ a tree-like domain of shapes. A shape abstraction is a mapping π from $\mathcal{S}$ to $\mathcal{P}$ that preserves juxtaposition and embedding.*

Two *structures* x and y have the *same* shape when $\pi(x) = \pi(y)$. Two *sequences* s and t have a *common* shape if they have structures x_s and x_t such that $\pi(x_s) = \pi(x_t)$. To compare the shapes of two structures, they need not have the same primary sequence, nor even sequence similarity or same sequence length. Turning now to the folding space $\mathcal{F}(s)$ of a given RNA sequence s, we define the desired classes as inverse images of π:

Definition 5.2.2 *For a given RNA sequence s, its (concrete) folding space $\mathcal{F}(s)$ is the set of all legal structures according to the rules of base pairing. Its (abstract) shape space is $\mathcal{P}(s) = \{\pi(x)|x \in \mathcal{F}(s)\}$. The class of p-shaped structures in $\mathcal{F}(s)$ is $\{x|x \in \mathcal{F}(s), \pi(x) = p\}$*

In other words, the shape class p is $\pi^{-1}(p) \cap \mathcal{F}(s)$. As the inverse image of a function always induces an equivalence relation, unique representatives are defined as follows:

Definition 5.2.3 *The representative structure $\hat{p}$ for shape class p is the element that has minimal free energy among all structures in the class.*

There is the rare case that two structures in a shape also have the same energy, in which case the representative is the smallest one under a lexicographic ordering on trees. The shape representative structures will be called *shreps* for short, to distinguish them from an explicit representation of the shape as a whole, which we will introduce below. The above definitions must be complemented by concrete data structures representing RNA structures and shapes. The different structural components in RNA are single stranded regions, hairpin loops, stacking regions, bulges on the 5' or on the 3' side, internal loops, and multiloops. Furthermore, we have lists of adjacent structures, such as the components of the external loop. These structural components are denoted by node labels SS, HL, SR, BL and BR, IL, ML, and AD respectively. Technically, label E is needed to denote an empty list of adjacent components. Individual nucleotides A, C, G, U, as well as strings thereof, represent themselves. Figure 5.1 shows a particular structure in several notations, including the one defined here. Note that in the tree notation, the primary sequence can be read from the leaves of the tree in left to right (5' to 3') order. In the text and for computer input, trees will be written as formulas, such as $AD(SS(ACGUU), E)$ or $AD(HL(C, UUU, G), E)$.

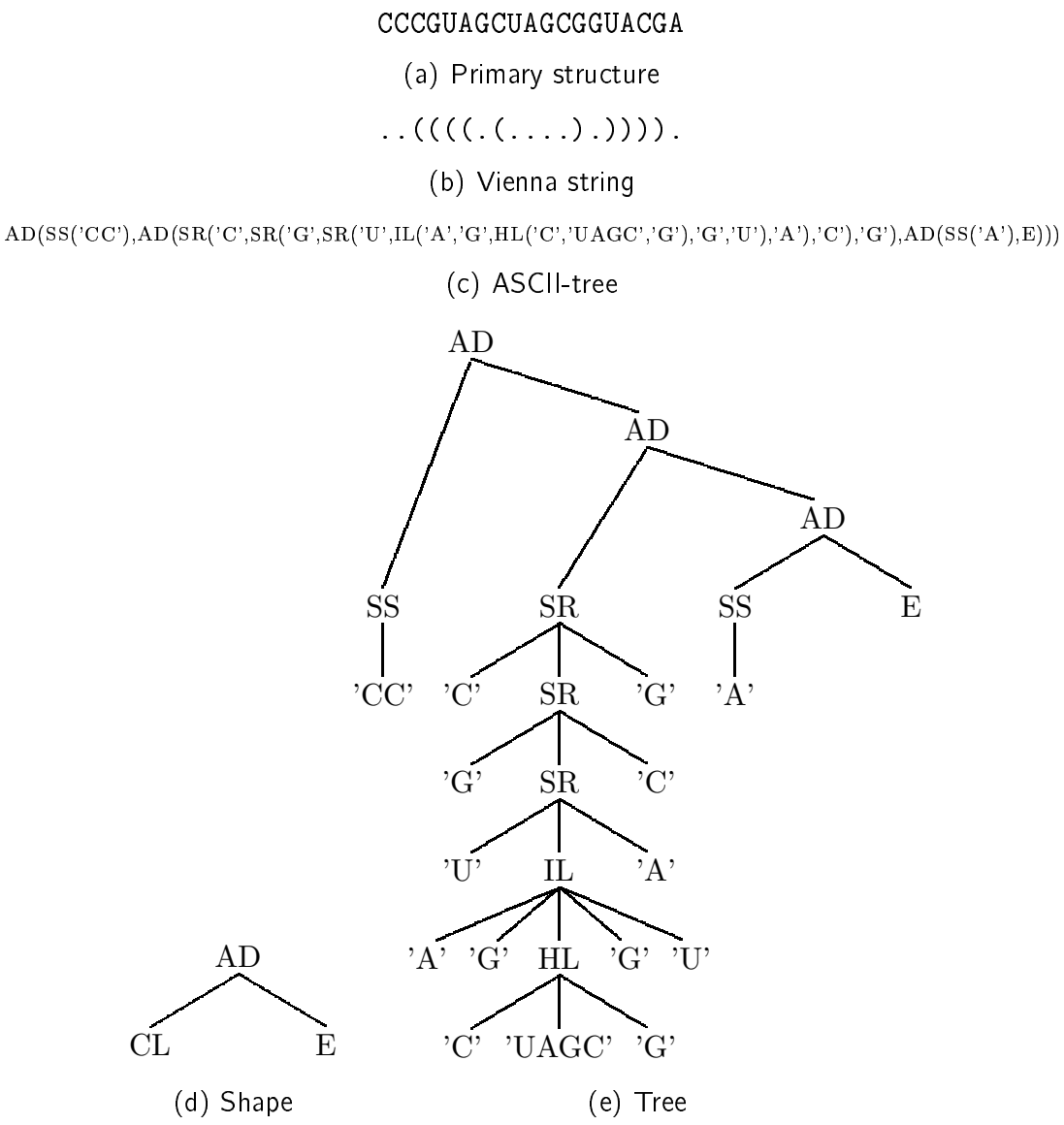


Figure 5.1: Notations of RNA primary ((a)) and secondary structure ((b)–(e)).

A first shape abstraction is to only care about open and closed structures, branching and adjacency. These situations are represented by node labels OP, CL, FK (from "fork"), AD, and E. (Re-using AD and E in the domain of shapes should not give rise to confusion; they are just generic list constructors.)

The abstraction mapping π from structures to shapes is defined by the following equations. We use variables a, b for nucleotides; I, I' for loop sequences; c for a list of adjacent components; and x for arbitrary structures.

$$\begin{aligned}
\pi(SS(I)) &= OP && (5.1)\\
\pi(HL(a,I,b)) &= CL && (5.2)\\
\pi(SR(a,x,b)) &= \pi(x) && (5.3)\\
\pi(BL(a,I,x,b)) &= \pi(x) && (5.4)\\
\pi(BR(a,x,I,b)) &= \pi(x) && (5.5)\\
\pi(IL(a,I,x,I',b)) &= \pi(x) && (5.6)\\
\pi(ML(a,c,b)) &= FK(\pi(c)) && (5.7)\\
\pi(AD(SS(I),c)) &= \pi(c) && (5.8)\\
\pi(AD(x,c)) &= AD(\pi(x),\pi(c)) \text{ for } x \neq SS(I) && (5.9)\\
\pi(E) &= E && (5.10)
\end{aligned}$$

It is easy to see that this abstraction function retains hairpins and multiloops, but abstracts from stack lengths, bulges, internal loops, and single-stranded regions (except for the case of the completely unpaired structure). It completely abstracts from primary sequence. This abstraction might be too strong in some cases, especially with short sequences that do not have much chance to show shape variation on this level of abstraction. In such cases, weaker abstraction functions that retain more structural detail can be defined in a similar way. The ones that are implemented in the program `RNAshapes` are described in Section 5.3.

While all the computational analysis of shapes is based on these tree representations, it is convenient for the human eye to introduce string representations for structures as well as for shapes. A notation for shapes, using homomorphism $\nu_{\mathcal{P}}$ is defined as follows, where $\ldots_k$ means k dots, $|I|$ is the length of string I, and ϵ denotes the empty string.

$$\begin{aligned}
\nu_P(OP) &= _ && (5.11)\\
\nu_P(CL) &= [\,] && (5.12)\\
\nu_P(FK(c)) &= [\,\nu_P(c)\,] && (5.13)\\
\nu_P(AD(x,c)) &= \nu_P(x)\nu_P(c) && (5.14)\\
\nu_P(E) &= \varepsilon && (5.15)
\end{aligned}$$

This is analogous to the familiar "Vienna" notation for structures, here defined as ν_S:

$$\begin{aligned}
\nu_S(SS(l)) &= \ldots_{|l|} && (5.16)\\
\nu_S(HL(a,l,b)) &= (\ldots_{|l|}) && (5.17)\\
\nu_S(SR(a,x,b)) &= (\ \nu_S(x)\ ) && (5.18)\\
\nu_S(BL(a,l,x,b)) &= (\ldots_{|l|}\ \nu_S(x)\ ) && (5.19)\\
\nu_S(BR(a,x,l,b)) &= (\ \nu_S(x)\ \ldots_{|l|}) && (5.20)\\
\nu_S(IL(a,l,x,l',b)) &= (\ldots_{|l|}\ \nu_S(x)\ \ldots_{|l'|}) && (5.21)\\
\nu_S(ML(a,c,b)) &= (\ \nu_S(c)\ ) && (5.22)\\
\nu_S(AD(x,c)) &= \nu_S(x)\nu_S(c) && (5.23)\\
\nu_S(E) &= \epsilon && (5.24)
\end{aligned}$$

Note the simple recursive definitions, whereas a direct definition of the mapping from the Vienna string $\nu_S(x)$ to the corresponding shape's notation $\nu_P(\pi(x))$ requires a parsing function. Such simplicity is the advantage of using a tree representation. Figure 5.3 shows some structures in Vienna notation, together with their shape notation under the abstraction function π.

Any sensible notation function must be injective, i.e. it must not map two distinct objects to the same notation. This appears to be violated, as $\nu_S(IL(a,l,x,l',b)) = \nu_S(ML(a,AD(x,E),b) = (\nu_S(x))$. However, the recurrences that analyse the search space have been designed to be unambiguous [67], and hence a candidate of the form $ML(a,AD(x,E),b)$, a non-branching multiloop, is never considered. With this in mind, it is easy to show that both ν_S and ν_P are injective. Hence, the program `RNAshapes` can keep the tree representations for itself, and faithfully communicate with its users via the string representations of structures and shapes.

5.3 Abstraction Levels

In the analysis of RNA secondary structure different structural elements are in focus of the researcher. In the case of conformational switching it might be the nesting pattern of stems, whereas in the case of protein binding sites it might be the existence of a bulge or internal loop. For this reason I developed five different types of shapes, each abstracting from structural detail to a different degree. The most abstract type 5 is given by the abstraction function shown in Equations 5.1–5.10 and abstracts from bulge and internal loops as well as from singlestrands. More detail about singlestranded regions in multiloops and the external loop is kept using type 4. Shape type 3 again abstracts from all kinds of singlestrands but retains information on the nesting of bulge and internal loops. Re-adding the representation of singlestrands in multiloops and the external loop yields shapes of type 2. The most detailed shape type abstracts only from the lengths of stacking and singlestranded regions and is referred to as type 1. An example for the different abstractions is shown in Figure 5.2. The implementation is shown in Appendix B.2.4.

```
Sequence:      gttaatgtagcttaataacaagatggataattgtatcccataaaca
Structure:     ((((.((.....)).))))....((((((....))))))........
Shape type 1: [_[_]_]_[_]_
Shape type 2: [[]]_[]_
Shape type 3: [[]][]
Shape type 4: []_[]_
Shape type 5: [][]
```

Figure 5.2: Example showing the different abstraction levels.

5.4 Folding Based on Shapes

5.4.1 Algorithm

The program `RNAshapes` was developed using the ADP framework (see Section 3.4.1). Since we want to explore the whole or at least a large part of the folding space of RNA, the program makes use of a grammar describing the folding space of RNA including dangling bases and disallowing isolated base pairs. The evaluation is based on algebras like the ones shown in Equations 5.1–5.10 and 5.16–5.24 for shapes and "Vienna" notation, respectively. Analogous to these examples, an algebra for free energy calculation scores the structural elements with their energy contribution obtained from the thermodynamic energy parameters. The implementation makes use of a combination of these three algebras in a triple-algebra of the form: (energy, shape, "Vienna" notation). The essential part of the implementation is the objective function h which filters the list of (intermediate) solutions and keeps entries with lowest free energy for each distinct shape. It is defined as follows:

$$
\begin{aligned}
&h([s_1,\ldots,s_n]) = h'([],\ (filter(e\_range,\ [s_1,\ldots,s_n]))), \text{ where}\\
&h'([sh_1,\ldots,sh_m],\ [s_1,\ldots,s_n]) = h'(insert\ (s_1,[sh_1,\ldots,sh_m]),\ [s_2,\ldots,s_m]), \text{ where}\\
&insert((x_e,x_s,x_v),\ [\,]) = [(x_e,x_s,x_v)]\\
&insert((x_e,x_s,x_v),\ [(y_e,y_s,y_v)_1,\ldots,(y_e,y_s,y_v)_m])\\
&\quad = \begin{cases}
[(y_e,y_s,y_v)_1,\ldots] & ,\ x_s = y_s\ \&\&\ x_e \geq y_e\\
[(x_e,x_s,x_v),(y_e,y_s,y_v)_2,\ldots] & ,\ x_s = y_s\ \&\&\ x_e < y_e\\
[(y_e,y_s,y_v)_1,(insert((x_e,x_s,x_v),\ [(y_e,y_s,y_v)_2,\ldots])] & ,\ x_s \neq y_s
\end{cases}
\end{aligned}
$$

s_k refers to an (intermediate) solution and sh_k to an (intermediate) "shape-optimal" solution, where "shape-optimal" means that it attains the (so far) lowest free energy for its shape. s_k and sh_k are both of the triple-form (energy, shape, "Vienna" notation). The function $filter$ removes solutions that have higher free energy than the current minimal solution plus the chosen energy range (e_range). The actual implementation of the algebras mentioned in this section can be found in Appendix B.

5.4.2 Applications

If not mentioned explicitly, the following applications are based on the most abstract shape type 5, which gives the nesting of hairpin loops and multiloops.

Transfer RNA

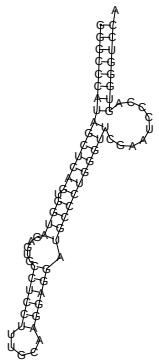
Transfer RNAs (tRNAs) are one of the best analysed RNA families. Various experiments have revealed the biological active structure of tRNAs which is known as the *cloverleaf* structure. In contrast to this, out of 99 tRNA sequences from the *Rfam* database [68], only 30 have a cloverleaf as their predicted *mfe*-structure (data not shown). The biological explanation for this is, that tRNAs possess modified bases which may on the one hand be no longer capable of forming base pairs, or on the other hand are able to interact in a different way. This alters the free energy of the predicted conformation such that it rises above the free energy of the cloverleaf (or vice versa), letting the latter achieve the energetical optimum. For structure prediction, when the modifications are unknown, current practice is to calculate suboptimal structures for a certain energy range and to subsequently search (by eye or by a simple pattern matching algorithm) for the cloverleaf structure in the list of suboptimals. For tRNAs this means that about 50–300 structures have to be checked. To give an example I chose the *Natronobacterium pharaonis* tRNA for alanine (gb: AB003409.1/96-167). The predicted *mfe*-structure is one hairpin with two bulge and one internal loop, as depicted in Figure 5.3(b). The cloverleaf structure, shown in Figure 5.3(d) appears at position 104 in the energy sorted list of 199 suboptimals, produced by `RNAsubopt`. Using `RNAshapes`, gives three shapes in an energy range of 5 kcal/mol above the *mfe* of which the rank 3 *shrep* is the cloverleaf structure. The output of `RNAshapes` and the squiggle plots for the *shreps* are shown in Figure 5.3.

Leader of HIV-1 genome

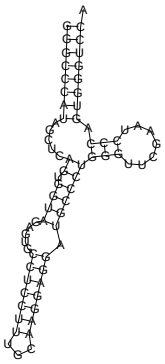
The leader of HIV-1 (see Section 4.5.4) is known to switch between two alternating conformations, a branched structure (S_2) and a more stable structure (S_1) which mainly consists of two adjacent helices. Structure prediction in an energy range of 3 kcal/mol based on the approach of abstract shapes revealed three *shreps* which are shown in Figures 5.4(a)–(c). Figure 5.4(a) shows the *mfe*-structure, which corresponds to the aforementioned structure S_1. The third *shrep* (Figure 5.4(c)) shows good correspondence to the conformation S_2. Further analysis, with a relaxed energy threshold of 6 kcal/mol produced 19 *shreps* and revealed that *shrep* 12 (Figure 5.4(d)) is equal to S_2. Summarising this means that 19 *shreps* had to be checked until both correct conformations could be identified. Performing the same approach based on complete suboptimal folding without considering lonely base pairs would have meant checking approximately 200,000 structures.

```
Shape     GGGCCCAUAGCUCAGUGGUAGAGUGCCUCCUUUGCAAGGAGGAUGCCCUGGGUUCGAAUCCCAGUGGGUCCA
[]        ((((((((((((((.((((.....(((((((...)))))))..)))))))))))).........))))))).  -35.9 kcal/mol
[[]]      ((((((((.....((.((((.....(((((((...)))))))..)))))))(((.......))).))))))).  -32.2 kcal/mol
[[][]]    ((((((...((((.......)))).(((((((...)))))))).....((((.......))))).)))))).  -31.7 kcal/mol
```

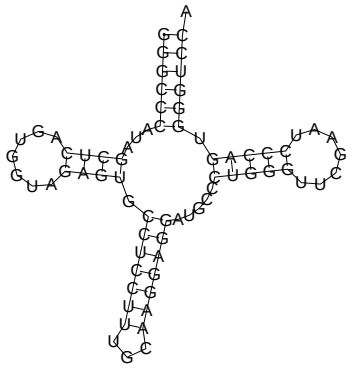
(a) Output of RNAshapes



(b) 1st *shrep*



(c) 2nd *shrep*



(d) 3rd *shrep*

Figure 5.3: Predicted *shreps* for *Natronobacterium pharaonis* tRNA-ala in an energy range of 5 kcal/mol above the *mfe*. This energy range holds 199 structures.

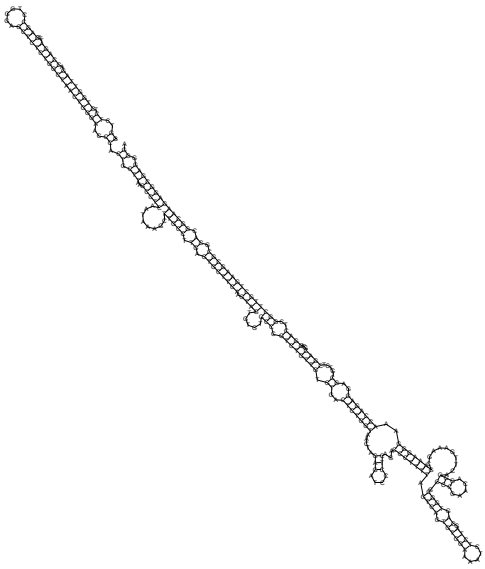
Human small nuclear RNA U2

Human small nuclear (sn)RNA U2 is an essential part of the spliceosome and forms five stem-loops, of which four are present in the predicted energy-optimal structure (Figure 5.5(a)). The second *shrep* possesses all five stems and an additional central helix (Figure 5.5(b)). As shown in Figure 5.5, a third shape is present in the near-optimal structure space which implies structural variability. The same three conformations have been predicted based on the `paRNAss` approach (Section 4.5.7). Conversely, the structure of U2 snRNA is supposed to be important for its correct function and, therefore, it should have evolved to exclude equally stable but dissimilar conformations in which it could get trapped and thereby inactivated. The spliceosome is a dynamic assembly of snRNAs (U1,U2,U4,U5,U6) and numerous associated proteins [69]. Hence, a solution to the above contradiction could be, that the active conformation of U2 snRNA gets stabilised by these RNAs and proteins.

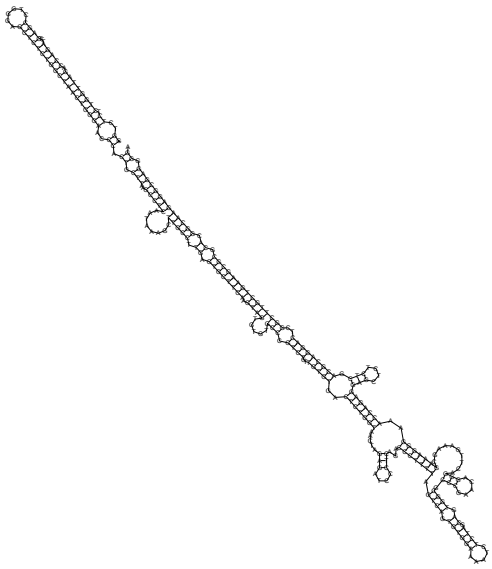
5.5 The Shape Space

5.5.1 Size

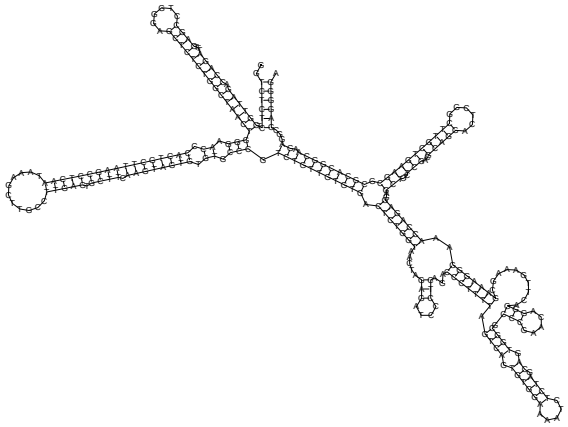
For any RNA sequence s, the number of suboptimal structures grows exponential with the sequence length N [5] as well as with the considered energy range above the *mfe*. Referring to Section 5.4.2, this means that for a sequence of moderate length, such as the leader of HIV-1 (281 nt), and a reasonable energy threshold of 6 kcal/mol the



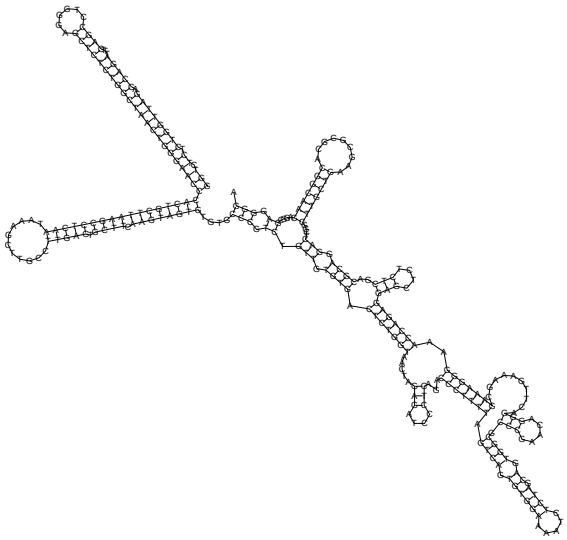
(a) 1st *shrep*: −108.3 kcal/mol, `[][[][[][]]]`, $\mathbf{S}_1$



(b) 2nd *shrep*: −107.9 kcal/mol, `[][[[][[][]]][]]`

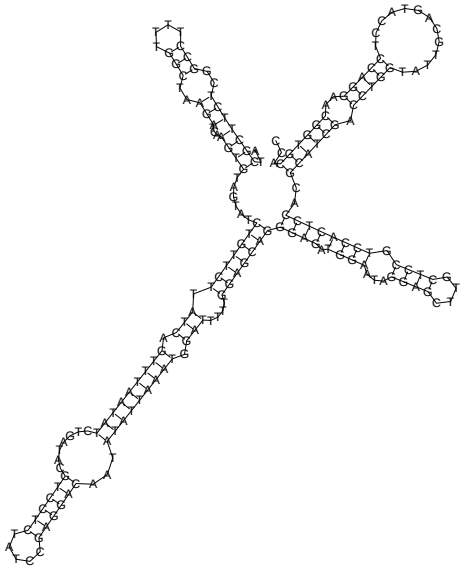


(c) 3rd *shrep*: −106.8 kcal/mol, `[[][][[[][[][]]][]]]`

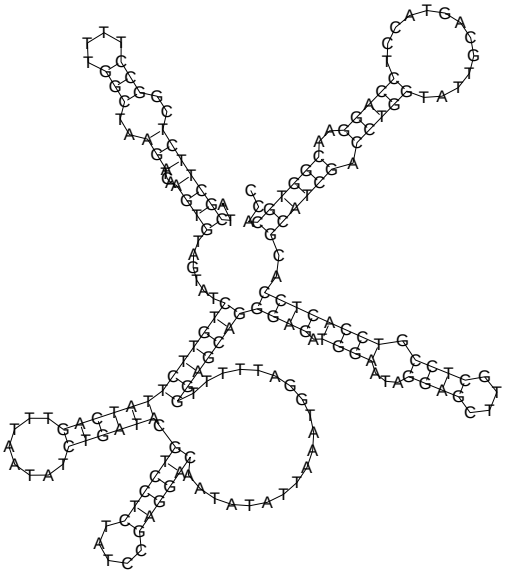


(d) 12th *shrep*: −102.8 kcal/mol, `[][][[[[][[][]]][]][]]`, $\mathbf{S}_2$

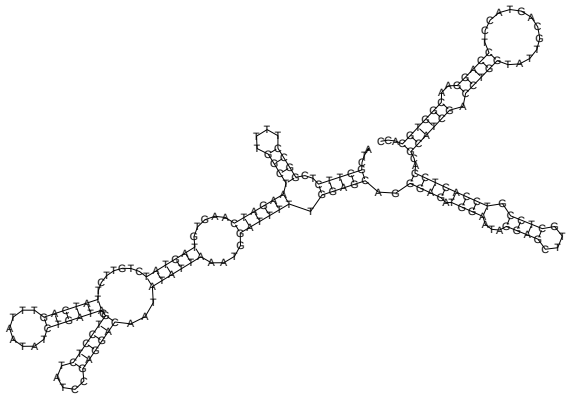
Figure 5.4: Subset of the 19 predicted *shreps* for HIV1-leader in an energy range of 6 kcal/mol above the *mfe*.



(a) 1st *shrep*: −69.12 *kcal/mol*, []⌊⌋[]⌊⌋[]⌊⌋[]



(b) 2nd *shrep*: −68.02 *kcal/mol*, []⌊⌋[]⌊⌋[[]⌊⌋[]]⌊⌋[]



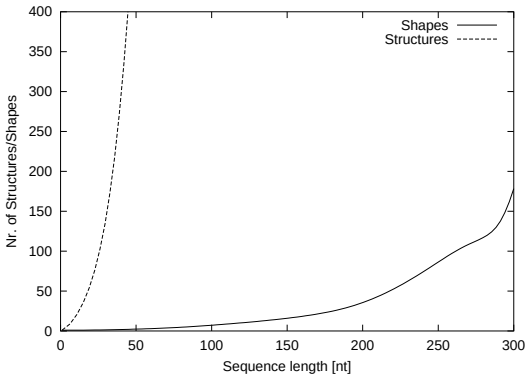
(c) 3rd *shrep*: −67.32 *kcal/mol*, []⌊⌋[]⌊⌋[[]⌊⌋[]]⌊⌋[]

Figure 5.5: Predicted *shreps* for human U2 snRNA in an energy range of $3kcal/mol$ above the *mfe*.

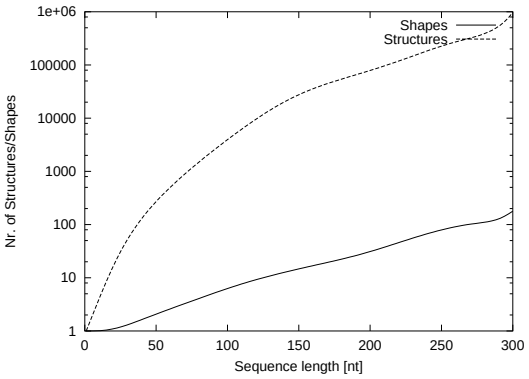
number of suboptimal structures exceeds 200,000 even when restricting to structures without isolated base pairs. In contrast to this, the number of *shreps* is 19, and thus stays significantly smaller. In order to reveal more general properties about the growth behaviour of the folding space $\mathcal{F}(s)$ and the shape space $\mathcal{P}(s)$, sequences from the *Rfam* data base [68] were analysed with lengths ranging from 20–300 nt in an energy range of 5 kcal/mol using shape type 5. Additionally, sequences of length ≈ 100 nt for energy ranges from 0–10 kcal/mol were examined to reveal the influence of the energy range. As a last experiment, I estimated the base of the exponential expression relating the number of structures (without isolated base pairs) and shapes (type 5), respectively, to the sequence length N ($size(\mathcal{F}(s)) = c_{\mathcal{F}} * a^N$, $size(\mathcal{P}(s)) = c_{\mathcal{P}} * b^N$). For this purpose I computed the number of all possible structures and shapes for random sequences of various lengths. I chose 30 sequences for each length; for the shape analysis at length 120 only one data point was calculated due to computational constraints. Figures 5.6(a) and (b) illustrate the slower (but still exponential) growth of $\mathcal{P}(s)$ compared to $\mathcal{F}(s)$ with growing sequence length in an energy range of 5 $kcal/mol$ above the *mfe*. For a growing energy range but fixed sequence length, $\mathcal{P}(s)$ grows slower (but still exponential) than $\mathcal{F}(s)$, too (data not shown). The ratio of shapes to structures is decreasing (asymptotically) with growing sequence length as well as with growing energy range (see Figures 5.6(c) and (d)). This also expresses the differences in growth rates between $\mathcal{P}(s)$ and $\mathcal{F}(s)$ for either sequence length or energy range. Figure 5.6(e) shows the overall number of structures and shapes for random sequences of increasing length. Their approximation by functions exponential in sequence length N gives estimates for $size(\mathcal{F}(s))$ and $size(\mathcal{P}(s))$. The analyses resulted in $size(\mathcal{F}(s)) \approx 0.04 * 1.4^N$ and $size(\mathcal{P}(s)) \approx 0.21 * 1.1^N$.

5.5.2 Morphology

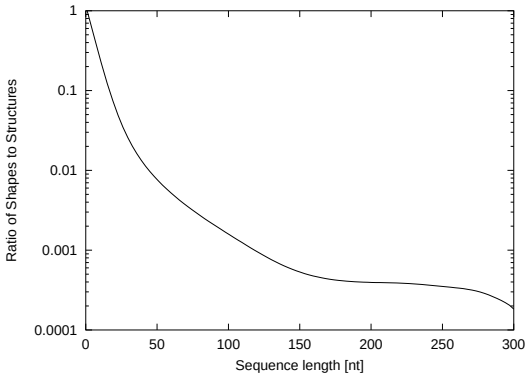
The structure space of RNA is described by the notion of neighbourhood. It is common practice to use the opening and closing of base pairs as the neighbourhood relation, meaning that neighbouring structures differ in exactly one base pair. As aforementioned, this can be used to classify structures as local (or global) minima, members of a valley, or saddle points in the energy landscape. The neighbourhood relations of the shape space are addition and removal of the structural elements that are considered by the actual shape type. For this reason, the shape abstraction somehow compresses the structure landscape in terms of replacing a valley of (many) structures with a valley of (not so many) shapes. Note that many, even non-neighbouring structure-valleys might get combined in one shape-valley. The shapes of two neighbouring structures are either the same, belong to the same shape valley or belong to neighbouring shape valleys. There is no possibility that their shapes belong to non-neighbouring shape valleys, whereas non-neighbouring structures may have neighbouring shapes.



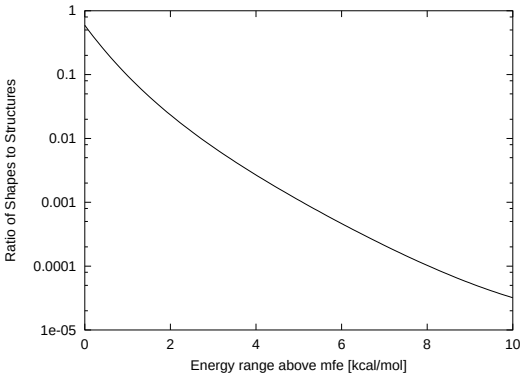
(a) Growth of structure space, resp. shape space with sequence length (energy range 5 $kcal/mol$).



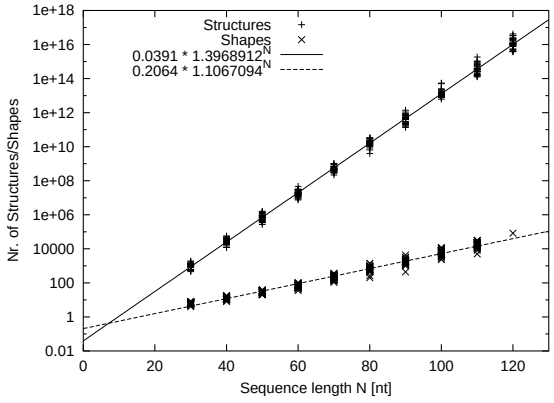
(b) Growth of structure space, resp. shape space with sequence length (energy range 5 $kcal/mol$, log-scale).



(c) Shape/Structure ratio for growing sequence length (energy range 5 $kcal/mol$).



(d) Shape/Structure ratio for growing energy range ($n \approx 100$).



(e) Overall number of structures, resp. shapes (log-scale) with growing sequence length.

Figure 5.6: Comparison of folding space and shape space (based on shape type 5). All graphs show the slower growth of the shape space compared to the structure space.

5.6 Probabilities of Shapes

5.6.1 Computation

In Section 3.4.1, it is shown how the partition function can be used to calculate the probability of an individual structure. As already mentioned, the *mfe*-structure always has the highest probability. The probabilities of individual structures decrease with longer sequences, as the total and near-optimal number of structures increases. Furthermore, a scientist might not be interested in the probability of one specific structure, but in the probability of a class of similar structures, such as those having the same shape. This suggests to implement the calculation of the partition function and embed it in the approach of abstract shapes. Generally spoken, the probability of shape p is the sum of the probabilities of all structures $j \in p$.

$$P(p) = \sum_{j \in p} P(j) \tag{5.25}$$

A naive algorithm based on this equation would calculate the probability of each individual structure and sum these. Hence, this algorithm would suffer from the exponential explosion we want to circumvent with the approach of abstract shapes. A closer look at Equation 3.7 shows: The partition function is the sum of the Boltzmann weighted free energies of all structures of the structure space $\mathcal{F}(s)$. The free energy of an individual structure j is the sum of the energy contributions $E(x)$ of all structural elements $x \in j$. Thus, Equation 3.7 can be re-written as:

$$Q = \sum_{j \in \mathcal{F}(s)} e^{\frac{-\sum_{x \in j} E(x)}{RT}} \tag{5.26}$$

$$= \sum_{j \in \mathcal{F}(s)} \prod_{x \in j} e^{\frac{-E(x)}{RT}} \tag{5.27}$$

where R is the universal gas constant (0.00198717 kcal/K) and T the temperature in Kelvin. This equation shows how the partition function is computed recursively as the sum of the products of the Boltzmann weighted energy contributions of the structural elements over all structures.

Implemented in ADP this yields an algebra which multiplies the Boltzmann weighted energy contributions of the structural elements and uses summation as the objective function. The goal is to compute shapes together with their probabilities. The probability of shape p is its contribution Q_p to the partition function divided by the full partition function Q.

$$P(p) = \frac{Q_p}{Q} \tag{5.28}$$

How can we compute Q_p? The answer is to combine the shape notation algebra with the partition function algebra in such a way that, each time intermediate solutions have

```
((((((((((((((.((((.....(((((((...))))))).)))))))))).........))))))))).
```

(a) *Shrep* for shape 1: `[]`, $-35.9kcal/mol$, $P = 0.6430508$

```
(((((((.....((.((((.....(((((((...))))))).))))))(((......))).))))))).
```

(b) *Shrep* for shape 2: `[[][]]`, $-32.2kcal/mol$, $P = 0.0094475$

```
((((((...((((......)))).(((((((...))))))).....(((((......))))).)))))).
```

(c) *Shrep* for shape 3: `[[][][]]`, $-31.7kcal/mol$, $P = 0.34262794$

Figure 5.7: *Shreps* of the three most probable shapes of the *N. pharaonis* tRNA-ala together with the probabilities of the shapes (sorted by increasing energy).

the same shape, their Boltzmann weighted energies are added up. This combination is achieved using the so called product algebra [70], which is a general method to combine two algebras. The resulting algebra computes all feasible shapes together with their respective contributions to the partition function. The probability of an individual shape, which is calculated according to Equation 5.28, is the sum of the probabilities of all structures that are members of the same shape, exactly like stated in the beginning of this section. Note that this approach analyses the structure space and not the shape space, because the probabilities of shapes are computed from those of structures and structures are not present in the shape space. The implementation of the partition function and the product algebra is given in Appendix B.

5.6.2 Applications

Transfer RNA

Applying the shape probability approach to the tRNA example from Section 5.4.2 gives the results shown in Figure 5.7. The shape holding the mfe-structure is also the most probable one (shape 1), but approx. $\frac{1}{3}$ of the structures of the ensemble reside in the cloverleaf shape (shape 3). This is much more than for the second shape which holds an energetically better *shrep*. In terms of the structure space this means that three valleys are present in the near-optimal energy range, one for each shape. The valley holding structures of the first shape is the deepest (minimum free energy) and also the broadest (most structures reside in this shape). The valley with structures from the second shape is the second deepest but it is very narrow. The structures of the third shape are represented by a not so deep but rather broad valley.

Attenuator

The Attenuator, already described in Section 4.5.1, is known to switch from a translational inactive to a translational active conformation under specific conditions. These two conformations correspond to two valleys in the structure landscape that are separated by a mountain (energy barrier). For shape analysis this means that two shapes

```
...((((..(((....(((.........)))......))).)))).(((((((....)))))))......
```

(a) *Shrep* for shape 1: `[][]`, $-21.2kcal/mol$, $P = 0.00307785$

```
.((((((((((.((((((.................)).))))....)))))))...))))..........
```

(b) *Shrep* for shape 2: `[]`, $-20.93kcal/mol$, $P = 0.00185678$

```
...(((((.....((((((.................)).)))))...(((((((((....)))))))))))))
```

(c) *Shrep* for shape 3: `[[][]]`, $-19.33kcal/mol$, $P = 0.99506193$

Figure 5.8: *Shreps* of the three most probable shapes of the Attenuator together with the probabilities of the shapes (sorted by increasing energy).

with reasonable probability should be present. The corresponding experiment delivers the results summarised in Figure 5.8. The first striking result of this analysis is, that the shape holding the *mfe*-structure is not the most probable one. In fact, a shape with a best energy which is $\sim 2kcal/mol$ above the *mfe* has the highest probability. Shape 1 and 3 seem to be rather similar with respect to their *shreps*, so their probabilities can be added. This means that the shape with two hairpins, which may be embedded in a multiloop, has a probability of 0.998 and the shape with one hairpin has a probability of 0.002. In other words the Attenuator occurs to 99.8% in shape 1 or 3 and to 0.2% in shape 2. Shape 1/3 corresponds to the "off" position of the switch and the high probability resembles that this is its native position. The "on" position (shape 2) is very unlikely which might express that the attenuator is a tight regulator. The difference in the probability can also give an impression on the effort needed to trigger the switch.

dsrA

In case of dsrA, the most abstract shape is not applicable as both alternate conformations of this conformational switch would belong to the shape `[][][]`. An analysis using the less abstract shape type 3 had the outcome shown in Figure 5.9. The *shreps* imply that it is reasonable to combine shapes 1 and 4 and shapes 2 and 3. Based on this, shape 1/4 has a probability of 0.49 and shape 2/3 of 0.42, which is in good correspondence to the switching nature of this RNA.

Precursor of microRNA lin-4

microRNAs (miRNAs) are small (~22nt) regulatory RNAs that are processed from larger precursors, for which the secondary structure is assumed to play an important role. A common feature of all known precursors is, that they form a hairpin whose energy is significantly lower than for random sequences of the same dinucleotide distribution [71]. This is a hint to a well-defined secondary structure, which implies that the corresponding shape should have a very high probability. An analysis revealed, that for the precursor of *C. elegans* lin-4 [72] the shape `[]` has a probability of 0.9999995, which means that only 1 in 20,000 molecules has a different shape or that each molecule is 99.99995%

..(((((((.....)))))))...((.........(((...(((((....)))))...)))(((((((.....)))))))).))

(a) *Shrep* for shape 1: `[][[[]][]]`, $-27.9kcal/mol$, $P = 0.3830028$

..(((((((.....)))))))...((((((((((((.......)))))))..))))))..((((((((((.....))))))))))..

(b) *Shrep* for shape 2: `[][[]][]`, $-32.3kcal/mol$, $P = 0.30127695$

..(((((((.....)))))))((.((((((((((((.......)))))))..))).))).((((((((((.....))))))))))

(c) *Shrep* for shape 3: `[][[[[]]][]]`, $-25.1kcal/mol$, $P = 0.12366786$

..(((((((.....)))))))...................((((.....))))....((((((((((.....))))))))))..

(d) *Shrep* for shape 4: `[][][]`, $-32.1kcal/mol$, $P = 0.10630956$

Figure 5.9: *Shreps* of the four most probable shapes of dsrA, together with the shape probabilities (sorted by decreasing probability).

..(((.((((.((((.(((((((.(((.......)))))))))).))))..))))..)))...

(a) *Shrep* for shape 1: `[[[[[]]]]]`, $-24.9kcal/mol$, $P = 0.88655585$

..(((.((((((.((.(((((((.(((.......)))))))))).))))..))))..)))...

(b) *Shrep* for shape 2: `[[[[[[]]]]]]`, $-23.6kcal/mol$, $P = 0.10814524$

..(((.((((.((((.(((((((.............)))))))..))))..))))..)))...

(c) *Shrep* for shape 3: `[[[[]]]]`, $-22.34kcal/mol$, $P = 0.00505367$

..(((.((((((.((.(((((((.((.......)).)).)))))..))))..))))..)))...

(d) *Shrep* for shape 4: `[[[[[[[]]]]]]]`, $-20.3kcal/mol$, $P = 0.00024395$

Figure 5.10: *Shreps* of the four most probable shapes of the *C. elegans* lin-4 precursor, together with the shape probabilities (sorted by decreasing probability).

of its lifetime in the single hairpin shape. For the output a probability cut-off of 10^{-6} was used, which might be the reason that no further shapes appear. Another fact that has to be considered is that the shape abstraction might have been too strong. For this reason, I performed an analysis with a shape retaining more structural detail (type 3), giving the results shown in Figure 5.10. All *shreps* are very similar so it is reasonable to combine them in the single hairpin shape. Their summed up probability is 0.99999871 which is besides from rounding inaccuracies the same as the probability of the single hairpin shape.

5.7 Discussion

In this chapter I presented the concept of abstract shapes and their efficient computation. With this method the number of near-optimal solutions in RNA secondary structure prediction is reduced from several hundreds or thousands to only a few, which enables the researcher to get an overview of the structure space. Furthermore, this reduction makes comparative studies much faster and easier. For example, the identification of a

regulatory element in co-regulated RNAs can be done by multiple structure alignment of suboptimal structures. If only two such sequences are analysed, the number of necessary pairwise comparisons grows already quadratic with the number of suboptimal solutions. Thus, the computational burden can be reduced with the computation of abstract shapes in a quadratic way, too.

The introduction of the partition function approach to calculate the probabilities of shapes gives further insights into the structure space of RNA. First, the examples show that the shape holding the *mfe*-structure is not automatically the most probable one. Second, in the case of competing structures the probabilities express the frequency of each shape in the ensemble. Additionally, when searching the common shape of two sequences a promising approach would be to search for the one with the highest combined probability.

The remarkable result for the lin-4 precursor of *C. elegans*, together with the facts that all known miRNA precursors have a hairpin as their *mfe*-structure and that they are rather small ($\sim$ 100 nt), leads to the idea of using `RNAshapes` to search whole genomes for hairpins of this size having a very high probability. These should be good candidates for new miRNA precursors and hence also for new miRNAs.

A general problem in structure analysis is to choose a reasonable cut-off for suboptimal structures. Uncertainties in the energy parameters imply that structures whose energy is P% above the mfe have to be considered, whereas structural variability by molecular kinetics is bound by a certain energy range, e.g. 5 $kcal/mol$. With the probabilities of shapes it is now possible to choose a probability cut-off. This, though based on energy parameters, is independent of the actual *mfe* value and, due to the summation over all structures belonging to a shape, more robust against slight errors in the energy parameters. The actual value of this cut-off seems to be reasonable at about 10^{-4} as this would mean that only 1 in 10,000 molecules would attain this shape. Normally, only up to a few thousand copies of an RNA are present in the cell. In a kinetic point of view, a cut-off at 10^{-4} means that a relevant shape has to be present for at least 0.36 seconds if the RNA is stable for one hour.

The complexity of the algorithms is, as for the known folding algorithms using DP, $O(n^3)$ in time and $O(n^2)$ in space. The algorithm for the computation of shape probabilities uses 5 tables and 3 lists ($5n^2 + 3n$), whereas the algorithm to compute shapes and *shreps* in a certain energy range makes use of 3 tables and 1 list ($3n^2 + 1n$). The reason for the larger memory requirements of the algorithm for shape probabilities lies in the use of the unambiguous grammar. This one has extra productions, some of which have to be tabulated to ensure that the computation finishes in reasonable time.

Due to computational limitations, which partly result from the Haskell implementation, the algorithm for calculating abstract shapes is currently limited to sequences of length up to 350 nt and the algorithm for calculating shape probabilities to sequences of length up to 150 nt. For a reasonable number of functional RNAs this is sufficient, but it is of course desirable to be able to analyse longer sequences as well. Currently, I am working together with Peter Steffen to improve the performance by making use of his ADP compiler, which directly produces fast C-Code. Some extensions to the standard compiler are necessary to satisfy the specific needs of shape analysis.

CHAPTER 6

Summary and Outlook

In this thesis I have presented two tools for the analysis of the secondary structure space of RNA. The `paRNAss` approach is intended to reveal conformational switching in RNA and the evaluation showed that it is useful for this purpose. Furthermore, due to the improvements I introduced, it can be used to discover more general properties, such as the presence of three alternative and dissimilar structures. A major problem remaining is, that a user has to interpret the results, which is the crucial step for correct predictions. But, without `paRNAss` the detection of conformational switching would be much harder. A favourable improvement would be to switch from user to automated interpretation and even better to some kind of scoring. First attempts to score conformational switches were not conclusive. In some cases they delivered reasonable results, so that I am convinced that it is possible to design such a scoring.

Especially in the case of long sequences the results are often difficult to interpret. The main reason for this is that long sequences can form more dissimilar structures even when restricting to local minima (by refolding). A solution could be to define some measure for local dissimilarity, which to my knowledge was not tried before. Another helpful fact would be to know the relevance of each structure and to use this as a weight during the analysis. Two structures with large distance but low relevance could then be neglected and would not disturb the distance plots. But how should relevance be measured? Perhaps the approach of abstract shapes leads to an answer in the near future.

The idea of abstract shapes is implemented in the program `RNAshapes`. Its aim is to give a representative overview of the complete or the near-optimal structure space and to simplify subsequent analyses by reasonably reducing the number of objects. The benefits of `RNAshapes` are the low number of shapes and *shreps* the user is confronted with in the output, the possibility to adjust the abstraction level to fit to the problem, and the modular implementation which allows the easy introduction of up-coming enhancements, such as further types of shapes. Especially the implementation of the partition function approach for the computation of probabilities of shapes is a major improvement in RNA secondary structure analysis. As beforementioned, the longer the sequence the more dissimilar structures and also shapes are present, even in the near-optimal structure/shape

space. By looking at the shape probabilities, the researcher is able to decide which of the shapes are relevant and which not. Additionally, the number of shapes with reasonable probability hints at properties such as structural well-definedness, kinetic traps in the folding pathway, or alternating conformations. All this might make the approach of abstract shapes one of the standard tools in RNA structure analysis.

Another domain where `RNAshapes` could be helpful is in the classification of unknown sequences. Based on its shapes a sequence could be assigned to known families of RNAs, such as those collected in Rfam. This could be used as a prefiltering step to decrease the number of families under consideration and to speed up the classification process. One can even think of calculating a mean probability for the common shape of each family and to use this in combination with the shape probabilities of the new sequence to score the hits. The mean probability of the common shape or some other statistical measure could also serve as a measure for the structural diversity in this family.

Incorporation of the approach of abstract shapes into `paRNAss` should also be done in near future. This could improve the speed and the predictive power of `paRNAss` and would also allow to define one probability cut-off for all RNAs, which would reduce the number of parameters and make predictions more reliable. It is of course not trivial to choose a reasonable cut-off value. Therefore, a thorough investigation of the approach of shape probabilities is necessary, which is currently in progress. One can also think of using a modified version of `RNAshapes` to search whole genomes for sequences having two shapes with similar probability and to validate these with `paRNAss` to discover unknown conformational switches. Furthermore, searching for sequences having one shape with very high probability and, hence, a well-defined structure, might reveal new members of functional non-coding RNA. In order to achieve this, the performance of `RNAshapes` has to be improved, which is carried out at the moment.

Both tools are available on the Bielefeld University Bioinformatics Server (BiBiServ, http://bibiserv.techfak.uni-bielefeld.de/{parnass,rnashapes}). The program `RNAshapes`, as well as its source code, is also available for download for various platforms.

Bibliography

[1] K. Krueger, P. Grabowski, A. Zaug, J. Sands, D. Gottschling, and T. Cech. Self-splicing RNA - auto-excision and auto-cyclization of the ribosomal-RNA intervening sequence of *Tetrahymena*. *Cell*, 31:147–157, 1982.

[2] D.P. Bartel. MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell*, 116:281–297, 2004.

[3] M. Ridanpää, H. van Eenennaam, K. Pelin, R. Chadwick, C. Johnson, B. Yuan, W. vanVenrooij, G. Pruijn, R. Salmela, S. Rockas, O. Makitie, I. Kaitila, and A. de la Chapelle. Mutations in the RNA component of RNase MRP cause a pleiotropic human disease, cartilage-hair hypoplasia. *Cell*, 104:195–203, 2001.

[4] D.H. Mathews, J. Sabina, M. Zuker, and H. Turner. Expanded Sequence Dependence of Thermodynamic Parameters Provides Robust Prediction of RNA Secondary Structure. *J. Mol. Biol.*, 288:911–940, 1999.

[5] M.S. Waterman. *Introduction to Computational Biology*. Chapman & Hall / CRC, 1995.

[6] M. Zuker. On Finding All Suboptimal Foldings of an RNA Molecule. *Science*, 244: 48–52, 1989.

[7] S. Wuchty, W. Fontana, I.L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49:145–165, 1999.

[8] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast Folding and Comparison of RNA Secondary Structures (The Vienna RNA Package). *Chemical Monthly*, 125:167–188, 1994.

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

[10] J. Kitagawa, Y. Futamura, and K. Yamamoto. Analysis of the conformational energy landscape of human snRNA with a metric based on tree representation of RNA structures. *Nucleic Acids Res.*, 31(7):2006–2013, 2003.

[11] R. Giegerich, D. Haase, and M. Rehmsmeier. Prediction and visualization of structural switches in RNA. In *Proc. 1999 Pacific Symposium on Biocomputing*, pages 126–137. World Scientific, 1999.

[12] C. Flamm, I.L. Hofacker, P.F. Stadler, and M.T. Wolfinger. Barrier Trees of Degenerate Landscapes. *Z. Phys. Chem*, 216:155–173, 2002.

[13] P. Hogeweg and B. Hesper. Energy directed folding of RNA sequences. *Nucl. Acids Res.*, 12:67–74, 1984.

[14] B.A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *CABIOS*, 4:381–393, 1988.

[15] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local Similarity in RNA Secondary Structures. In *Proc. of the IEEE Computer Society Bioinformatics Conference (CSB)*, in press, 2003.

[16] B.A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *CABIOS*, 6:309–318, 1990.

[17] M. Rehmsmeier. Klassifikation von RNA-Sequenzen durch Analyse ihres Strukturraums. Diploma thesis, Bielefeld University, Faculty of Technology, 1996.

[18] J.H. Ward. Hierarchical grouping to optimize an objective function. *JASA*, 58: 236–244, 1963.

[19] S.B. Needleman and Wunsch C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

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

[21] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithm for loop matchings. *SIAM J. Appl. Math.*, 35:68–82, 1978.

[22] G. Chen, B.M. Znosko, X. Jiao, and D.H. Turner. Factors Affecting Thermodynamic Stabilities of RNA 3 x 3 Internal Loops. *Biochemistry*, 43:12865–12876, 2004.

[23] D. Evers and R. Giegerich. Reducing the Conformation Space in RNA Structure Prediction. In *German Conference on Bioinformatics*, pages 118–124, 2001.

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

[25] M. Zuker, D.H. Mathews, and D.H. Turner. Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide. In J. Barciszewski and B.F.C. Clark, editors, *RNA Biochemistry and Biotechnology*. NATO ASI Series, Kluwer Academic Publishers, 1999.

[26] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucl. Acids. Res.*, 31:3406–3415, 2003.

[27] G. Steger, H. Hofmann, J. Fortsch, H.J. Gross, J.W. Randles, H.L. Sanger, and D. Riesner. Conformational transitions in viroids and virusoids: comparison of results from energy minimization algorithm and from experimental data. *J. Biomol. Struct. Dyn.*, 2:543–571, 1984.

[28] J.S. McCaskill. The equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure. *Bioplymers*, 29:1105–1119, 1990.

[29] R. Giegerich. A Systematic Approach to Dynamic Programming in Bioinformatics. *Bioinformatics*, 16:665–677, 2000.

[30] R. Giegerich and C. Meyer. Algebraic Dynamic Programming. In Kirchner, H. and Ringeissen, C., editor, *Algebraic Methodology And Software Technology, 9th International Conference, AMAST 2002*, pages 349–364, Saint-Gilles-les-Bains, Reunion Island, France, 2002. Springer LNCS 2422.

[31] R. Giegerich, C. Meyer, and P. Steffen. Towards a discipline of dynamic programming. In Schubert, S. and Reusch, B. and Jesse, N., editor, *Informatik bewegt*, Lecture Notes in Informatics, pages 3–44. Springer, 2002.

[32] R. Giegerich and P. Steffen. Implementing Algebraic Dynamic Programming in the Functional and the Imperative Paradigm. In Boiten, E.A. and Möller, B., editor, *Mathematics of Program Construction*, volume 2386 of *Springer Lecture Notes in Computer Science*, pages 1–20. Springer, 2002.

[33] I. Tinoco, J.D. Puglisi, and J.R. Wyatt. RNA folding. *Nucleic Acids Mol. Biol.*, 4: 205–226, 1990.

[34] R Nussinov and G Pieczenik. Structural and combinatorial constraints on base pairings in large nucleotide sequences. *J. Theor. Biol.*, 106:244–259, 1984.

[35] H.M. Martinez. An RNA folding rule. *Nucl. Acids Res.*, 12:323–334, 1984.

[36] J.P. Abrahams, M. van den Berg, E. van Batenburg, and C.W.A. Pleij. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucl. Acids Res.*, 18:3035–3044, 1990.

[37] A.P. Gultyaev, F.H.D. van Batenburg, and C.W.A. Pleij. The Computer Simulation of RNA Folding Pathways Using a Genetic Algorithm. *J. Mol. Biol.*, 250:37–51, 1995.

[38] I.I. Titov, V.A. Ivanisenko, and N.A. Kolchanov. FITness - a WWW-resource for RNA folding simulation based on genetic algorithm with local minimization. *Comput. Techn.*, 5:48–56, 2000.

[39] H. Isambert and E.D. Siggia. Modeling RNAfolding paths with pseudoknots: Application to hepatitis delta virus ribozyme. *Proc. Natl. Acad. Sci. USA*, 97:6515–6520, 2000.

[40] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. RNA folding at elementary step resolution. *RNA*, 6:325–338, 2000.

[41] K. Kawasaki. Diffusion constants near the critical point for time-dependent Ising models. *Phys Rev*, 145:224–230, 1966.

[42] K.A. LeCuyer and D.M. Crothers. The Leptomonas collosoma Spliced Leader RNA can switch between two Alternate Structural Forms. *Biochemistry*, 32:5301–5311, 1993.

[43] P. Bengert and T. Dandekar. Riboswitch finder–a tool for identification of riboswitch RNAs. *Nucl. Acids Res.*, 32(Web Server issue):154–159, 2004.

[44] C. Flamm, I.L. Hofacker, S. Maurer-Stroh, P.F. Stadler, and M. Zehl. Design of multistable RNA molecules. *RNA*, 7:254–265, 2001.

[45] D. Barash. Second Eigenvalue of the Laplacian Matrix for Predicting RNA Conformational Switch by Mutation. *Bioinformatics*, Advance Access, 2004.

[46] G. Fayat, J.F. Mayaux, C. Sacerdot, M. Formant, M. Springer, M. Grunberg-Manago, and S. Blanquet. Escherichia coli Phenylalanyl-tRNA Synthetase Operon region. *J. Mol. Biol.*, 171:239–261, 1983.

[47] C. Philippe, L. Benard, C. Portier, E. Westhof, B. Ehresmann, and C. Ehresmann. Molecular dissection of the pseudoknot governing the translational regulation of Escherichia coli ribosomal protein S15. *Nucleic Acids Res.*, 23(1):18–28, 1995.

[48] D.W. Lazinski and J.M. Taylor. Regulation of the hepatitis delta virus ribozymes: To cleave or not to cleave? *RNA*, 1:225–233, 1995.

[49] I.G. Wool, A. Glück, and Y. Endo. Ribotoxin recognition of ribosomal RNA and a proposal for the mechanism of translocation. *Trends Biochem. Sci.*, 17:266–269, 1992.

[50] H.D. Madhani and C. Guthrie. A novel base-pairing interaction between U2 and U6 snRNAs suggests a mechanism for the catalytic activation of the spliceosome. *Cell*, 71(5):803–817, 1992.

[51] W. Winkler, A. Nahvi, and R.R. Breaker. Thiamine derivatives bind messenger RNAs directly to regulate gene expression. *Nature*, 419:952–956, 2002.

[52] W.C. Winkler, S. Cohen-Chalamish, and R.R. Breaker. An mRNA structure that controls gene expression by binding FMN. *Proc. Natl. Acad. Sci. USA*, 99(25): 15908–15913, 2002.

[53] V. Epshtein, A.S. Mironov, and E. Nudler. The riboswitch-mediated control of sulfur metabolism in bacteria. *Proc. Natl. Acad. Sci. USA*, 100(9):5052–5056, 2003.

[54] A. Walter, D. Turner, J. Kim, M. Lyttle, P. Müller, D. Mathews, and M. Zuker. Coaxial Stacking of Helixes Enhances Binding of Oligoribonucleotides and Improves Predictions of RNA Folding. *Proc. Natl. Acad. Sci. USA*, 91:9218–9222, 1994.

[55] M.M. Konarska, Padgett R.A., and P.A. Sharp. Trans splicing of mRNA precursors in vitro. *Cell*, 42:165–171, 1985.

[56] W.J. Murphy, K.P. Watkins, and Agabian N. Identification of a novel Y branch structure as an intermediate in trypanosome mRNA processing: evidence for trans splicing. *Cell*, 47:517–525, 1986.

[57] R.E. Sutton and J.C. Boothroyd. Evidence for trans splicing in trypanosomes. *Cell*, 47:527–535, 1986.

[58] R.E. Davis and Hodgson S. Gene linkage and steady state RNAs suggest trans-splicing may be associated with a polycistronic transcript in Schistosoma mansoni. *Mol Biochem Parasitol.*, 89:25–39, 1997.

[59] Y. Cao and S.A. Woodson. Destabilizing effect of an rRNA stem-loop on an attenuator hairpin in the 5' exon of the Tetrahymena pre-rRNA. *RNA*, 4:901–914, 1998.

[60] H. Huthoff and B. Berkhout. Two alternating structures of the HIV-1 leader RNA. *RNA*, 7:143–157, 2001.

[61] C. Workman and A. Krogh. No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nuc. Ac. Res.*, 27(24):4816–4822, 1999.

[62] R.A. Lease and M. Belfort. A trans-acting RNA as a control switch in Escherichia coli: DsrA modulates function by forming alternative structures. *Proc. Natl. Acad. Sci. USA*, 97(18):9919–9924, 2000.

[63] G. Pesole, S. Luini, G. Grillo, F. Licciulli, F. Mignone, C. Gissi, and C. Saccone. UTRdb and UTRsite: specialized database of sequences and functional elements of 5' and 3' untranslated regions of eukaryotic mRNAs. Update 2002. *Nucleic Acids Res.*, 30(1):335–340, 2002.

[64] R. Giegerich and J. Reeder. From RNA folding to thermodynamic matching, including pseudoknots. Technical Report 2003–03, Technische Fakultät, Universität Bielefeld, 2003.

[65] E. Rivas and S.R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068, 1999.

[66] P. Schuster, W. Fontana, P.F. Stadler, and I.L. Hofacker. From sequences to shapes and back: A case study in RNA secondary structures. *Proc. R. Soc. Lond. B.*, 255: 279–284, 1994.

[67] R. Giegerich. Explaining and Controlling Ambiguity in Dynamic Programming. *Proc. Combinatorial Pattern Matching*, pages 46–59, 2000.

[68] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31:439–441, 2003.

[69] C.B. Burge, T. Tuschl, and P.A. Sharp. *The RNA world.*, chapter Splicing Precursors to mRNAs. Cold Spring Harbor Laboratory Press, second edition, 1999.

[70] R. Giegerich and P. Steffen. Versatile and declarative dynamic programming using pair algebras. in preparation, 2004.

[71] E. Bonnet, J. Wuyts, P. Rouzé, and Y. Van de Peer. Evidence that microRNA precursors, unlike other non-coding RNAs, have lower folding free energies than random sequences. *Bioinformatics*, 2004. doi:10.1093/bioinformatics/bth374.

[72] P.H. Olsen and V. Ambros. The lin-4 regulatory RNA controls developmental timing in Caenorhabditis elegans by blocking LIN-14 protein synthesis after the initiation of translation. *Dev. Biol.*, 216:671–680, 1999.

[73] A. Nahvi, N. Sudarsan, M.S. Ebert, X. Zou, K.L. Brown, and R.R. Breaker. Genetic Control by a Metabolite Binding mRNA. *Chem. Biol.*, 9:1043–1049, 2002.

[74] H. Groeneveld, K. Thimon, and J. van Duin. Translational control of maturation-protein synthesis in phage MS2: a role for the kinetics of RNA folding? *RNA*, 1 (1):79–88, 1995.

[75] K. Semrad and R. Schroeder. A ribosomal function is necessary for efficient splicing of the T4 phage thymidylate synthase intron in vivo. *Genes. Dev.*, 12(9):1327–1337, 1998.

[76] A.M.G. Dirac, H. Huthoff, J. Kjems, and B. Berkhout. Regulated HIV-2 RNA dimerization by means of alternative RNA conformations. *Nucleic Acids Res.*, 30 (12):2647–2655, 2002.

[77] R.C.L. Olsthoorn, S. Mertens, F.T. Brederode, and J.F. Bol. A conformational switch at the 3' end of a plant virus RNA regulates viral replication. *EMBO J.*, 18 (17):4856–4864, 1999.

[78] P.J. Schlax, K.A. Xavier, T.C. Gluick, and D.E. Draper. Translational Repression of the Escherichia coli α Operon mRNA. *J. Biol. Chem.*, 276(42):38494–38501, 2001.

[79] T. Franch, A.P. Gultyaev, K.J. Öistämö, K. Gerdes, and C.W.A. Pleij. Programmed cell death by hok/sok of plasmid R1: processing at the hok mRNA 3'-end triggers structural rearrangements that allow translation and antisense RNA binding. *J. Mol. Biol.*, 273(1):38–51, 1997.

[80] D.A. Rodionov, A.G. Vitreschak, A.A. Mironov, and M.S. Gelfand. Regulation of lysine biosynthesis and transport genes in bacteria: yet another RNA riboswitch? *Nucl. Acids Res.*, 31:6748–6757, 2003.

[81] C.A. Côtè, C.L. Greer, and Peculis B.A. Dynamic conformational model for the role of ITS2 in pre-rRNA processing in yeast. *RNA*, 8:786–797, 2002.

[82] O. Schilling, I. Langbein, M. Müller, M.H. Schmalisch, and J. Stülke. A protein-dependent riboswitch controlling *ptsGHI* operon expression in *Bacillus subtilis*: RNA structure rather than sequence provides interaction specificity. *Nucl. Acids Res.*, 32: 2853–2864, 2004.

Appendices

APPENDIX A

paRNAss Analyses

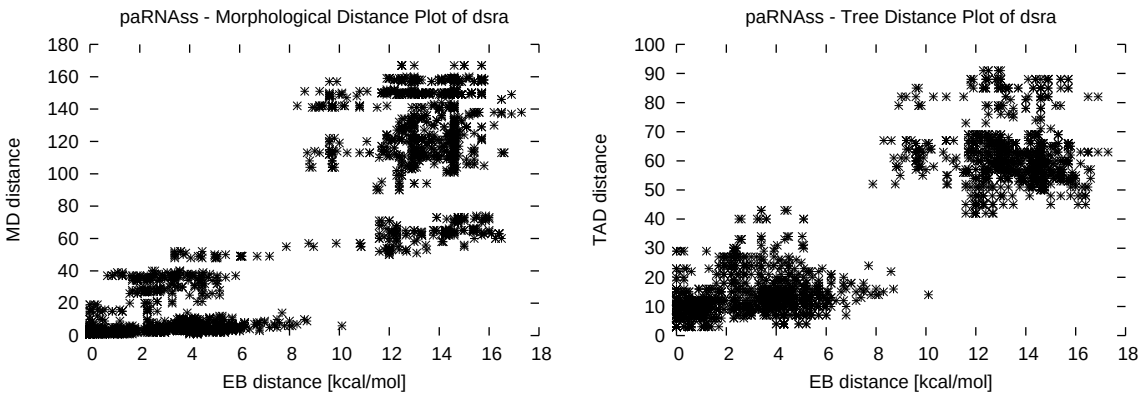
A.1 E. coli DsrA

Function DsrA is an 87-nucleotide regulatory RNA that acts in trans with two different mRNAs, *hns* and *rpoS*. DsrA has opposite effects on these transcriptional regulators. DsrA interacts with the *hns* mRNA start and stop codon regions to form a coaxial stack and thereby leads to degradation of *hns* mRNA. In contrast, DsrA base pairs in a discrete fashion with *rpoS* mRNA translational operator and enables translation. Thus, different conformations of DsrA lead to opposite regulatory effects on target RNAs.

Parameters

- Energy range: $3kcal/mol$
- Search space: feasibles
- Temperature: 37 °C
- Max. Structures: 50

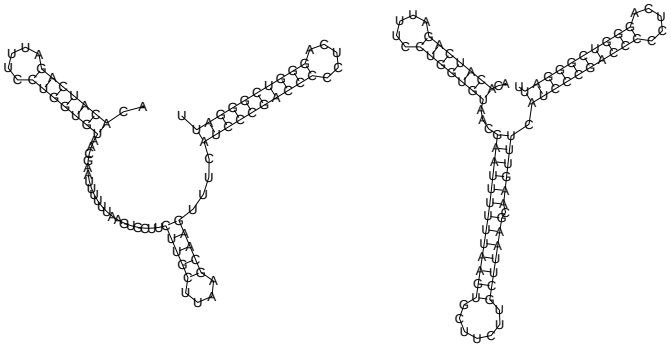
Distance plots



(a) d_{MD} (b) d_{TAD}

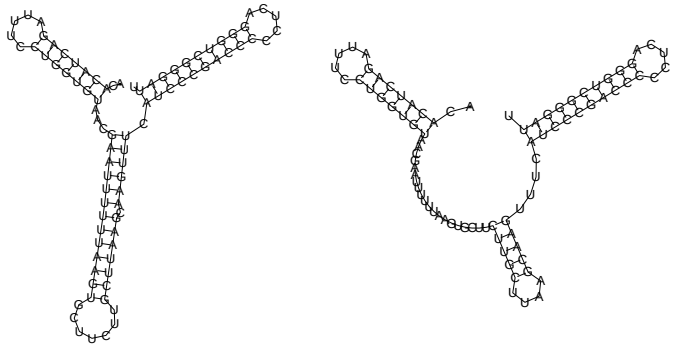
Figure A.1: Distance plots for dsrA.

Consensus structures



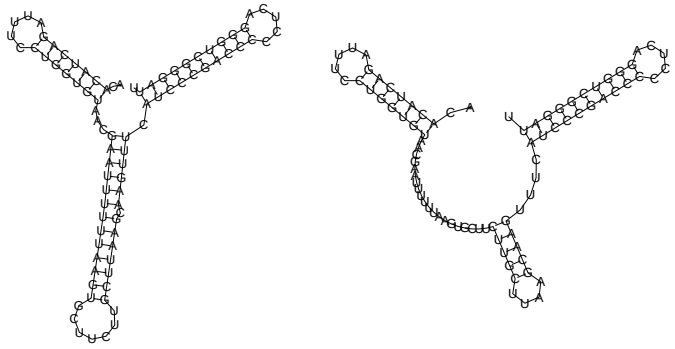
(a) c_1 (b) c_2

Figure A.2: Consensus structures for dsrA based on d_{MD}.



(a) c_1 (b) c_2

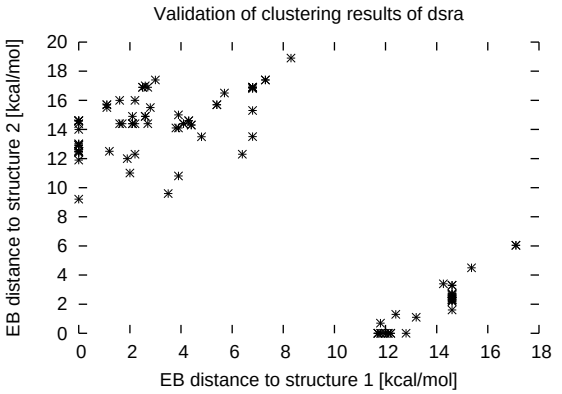
Figure A.3: Consensus structures for dsrA based on d_{TAD}.



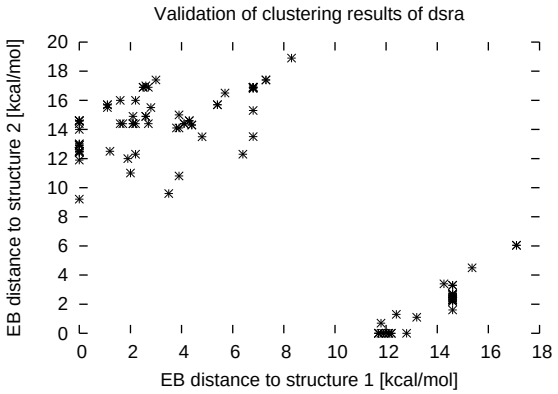
(a) c_1 (b) c_2

Figure A.4: Consensus structures for dsrA based on d_{EB}.

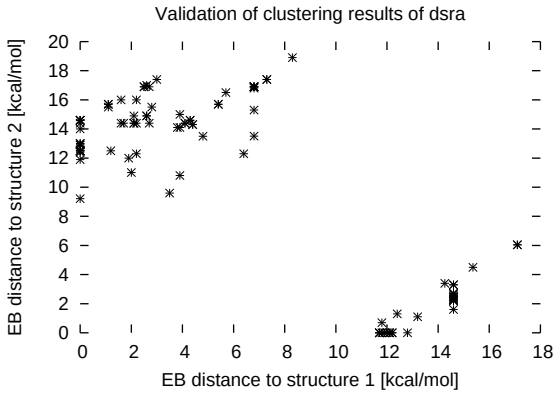
Consensus structure validation



(a) d_{MD}, $pkMeasure = 6$



(b) d_{TAD}, $pkMeasure = 6$



(c) d_{EB}, $pkMeasure = 6$

Figure A.5: Validation plots for dsrA.

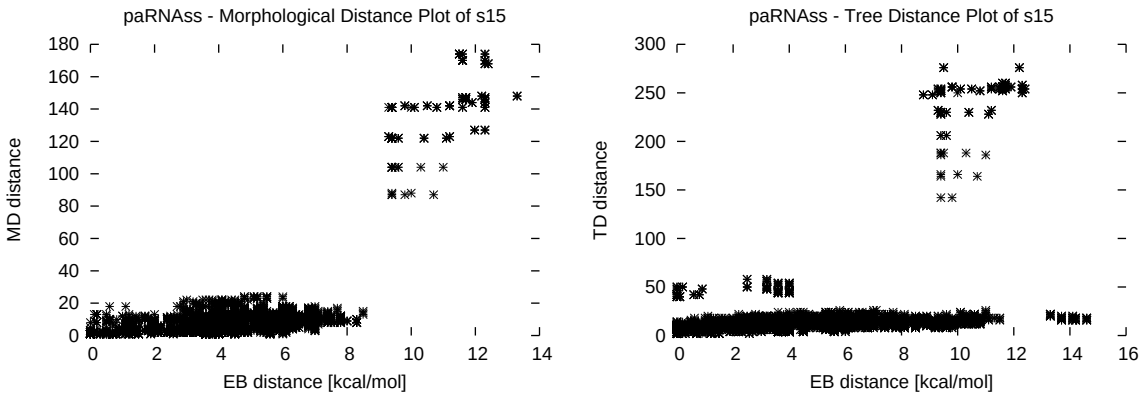
A.2 E. coli S15

Function The ribosomal protein S15 controls its own translation by binding to a region of the mRNA overlapping the ribosome binding site. That region of the mRNA can fold into two mutually exclusive conformations that are in dynamic equilibrium: a structure with two hairpins and a pseudoknot [47]. S15 protein binds to the pseudoknot conformation and allows ribosome binding, but it traps the ribosome on its loading site, preventing the formation of the active ternary 30S/mRNA/initiator-tRNA complex.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 100

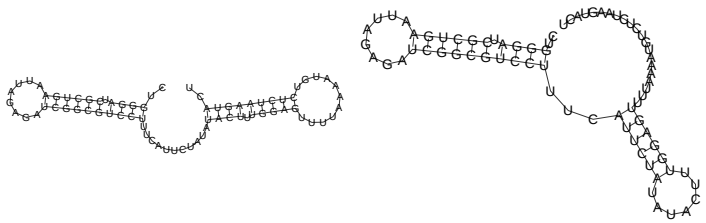
Distance plots



(a) d_{MD} (b) d_{TAD}

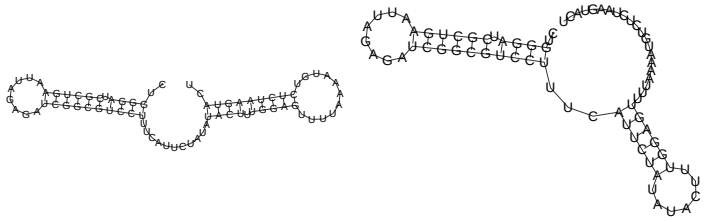
Figure A.6: Distance plots for s15.

Consensus structures



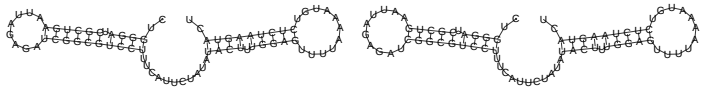
(a) c_1 (b) c_2

Figure A.7: Consensus structures for s15 based on d_{MD}.



(a) c_1 (b) c_2

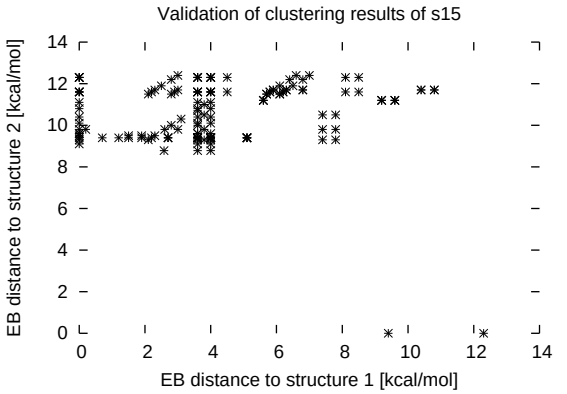
Figure A.8: Consensus structures for s15 based on d_{TAD}.



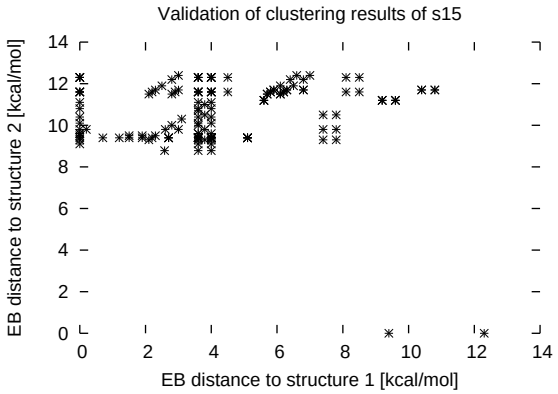
(a) c_1 (b) c_2

Figure A.9: Consensus structures for s15 based on d_{EB}.

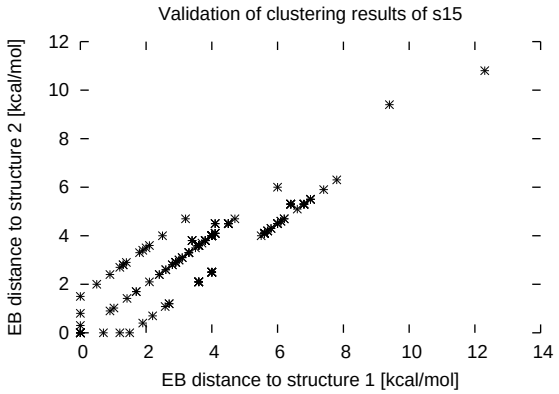
Consensus structure validation



(a) d_{MD}, $pkMeasure = 5$



(b) d_{TAD}, $pkMeasure = 5$



(c) d_{EB}, $pkMeasure = -1$

Figure A.10: Validation plots for s15.

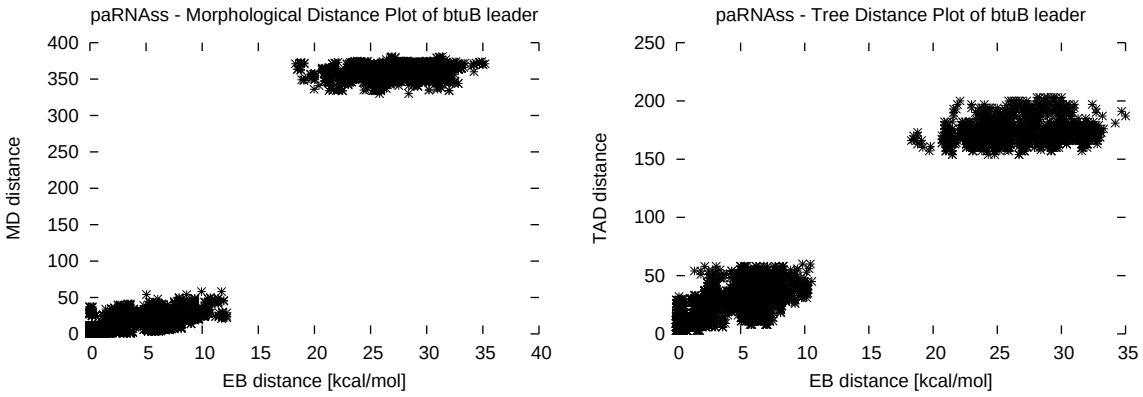
A.3 E.coli btuB leader

Function The *btuB* gene encodes the outer membrane cobalamin transporter in *Escherichia coli*. Its expression is strongly reduced on growth with cobalamins, which is primarily due to changes in translation. Vitamin B_{12} regulates expression of this gene by binding either directly or via a protein to the target site on the mRNA. Binding of vitamin B_{12} leads to the formation of a hairpin in the leader of the mRNA that sequesters the ribosomal binding site. The release of vitamin B_{12} causes a structural transition in the RNA, which opens up the Shine–Dalgarno (SD) sequence for ribosomes. Once enough vitamin B_{12} is produced , it reoccupies its binding site, refolding the SD enclosing hairpin [73]. This was the first example for the class of so called riboswitches, which directly bind to metabolites.

Parameters

- Energy range: $2kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 129

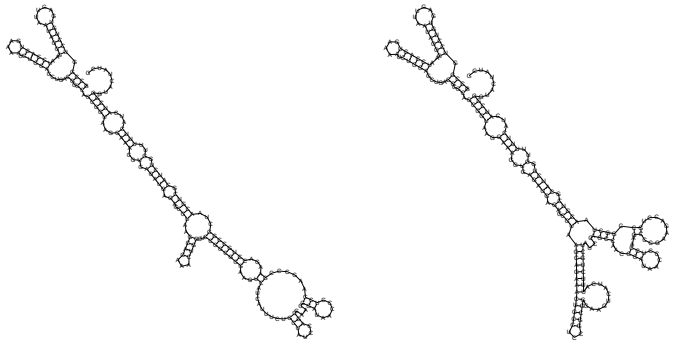
Distance plots



(a) d_{MD} (b) d_{TAD}

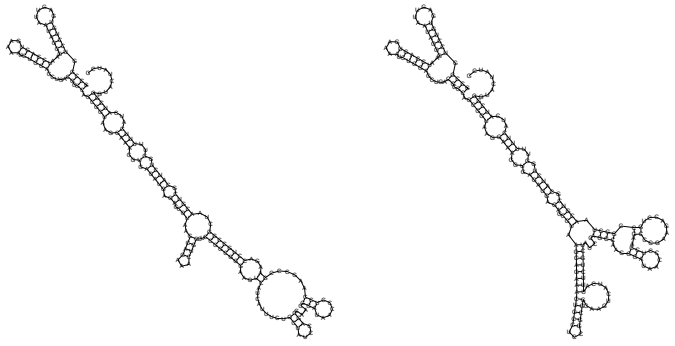
Figure A.11: Distance plots for btuB leader.

Consensus structures



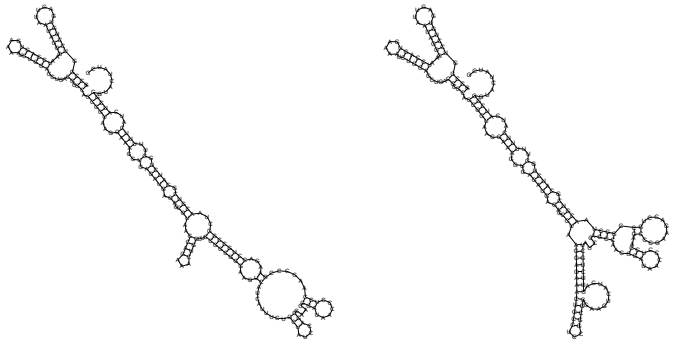
(a) c_1 (b) c_2

Figure A.12: Consensus structures for btuB leader based on d_{MD}.



(a) c_1 (b) c_2

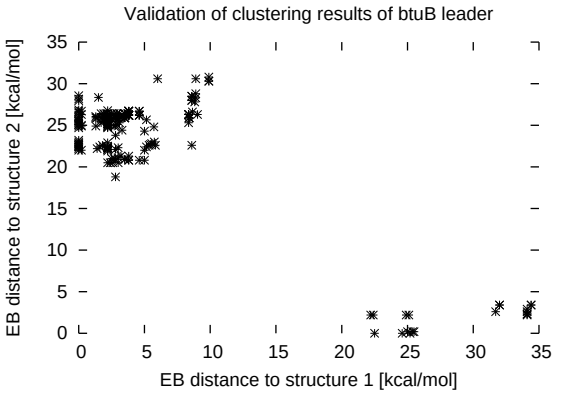
Figure A.13: Consensus structures for btuB leader based on d_{TAD}.



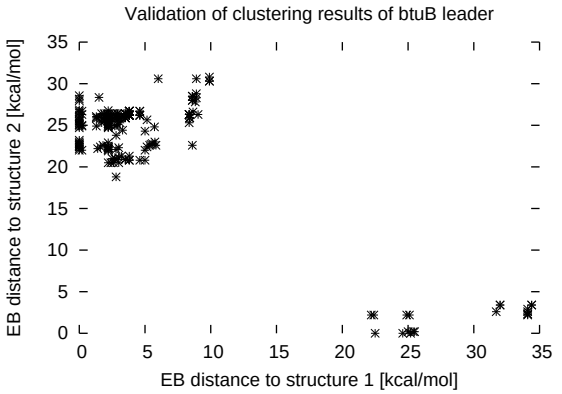
(a) c_1 (b) c_2

Figure A.14: Consensus structures for btuB leader based on d_{EB}.

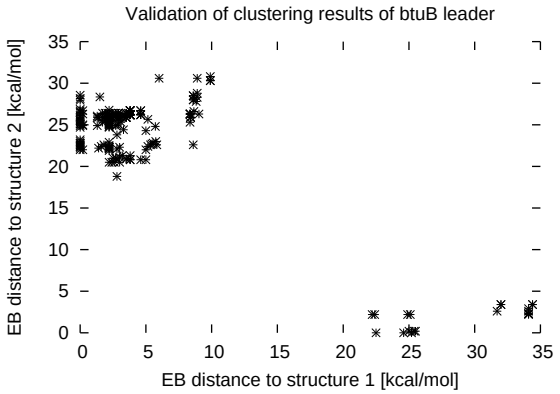
Consensus structure validation



(a) d_{MD}, $pkMeasure = 23$



(b) d_{TAD}, $pkMeasure = 23$



(c) d_{EB}, $pkMeasure = 23$

Figure A.15: Validation plots for btuB leader.

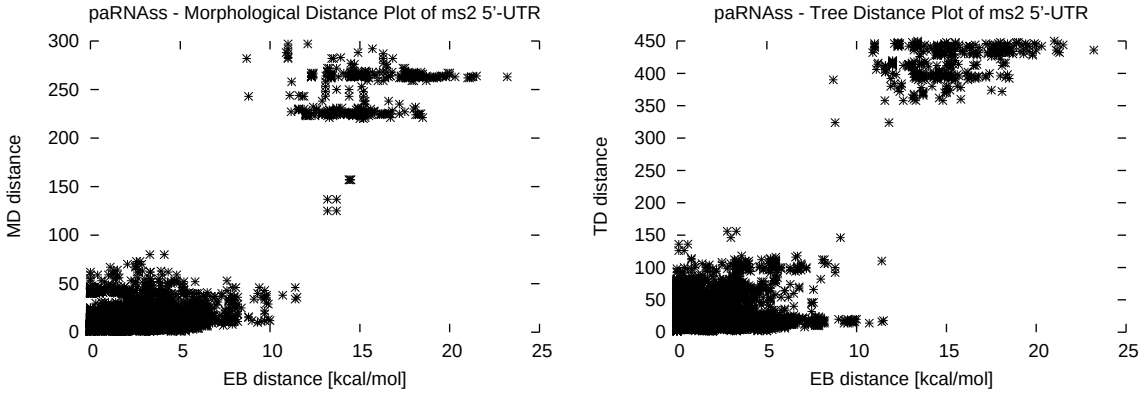
A.4 5'-UTR of MS2 RNA genome

The genomic RNA of bacteriophage MS2 codes for four genes. The regulation of translation of these four different genes is controlled at the level of RNA folding and unfolding, where the first gene is regulated independent from the others. In the *mfe*-structure of the leader, the Shine-Dalgarno (SD) sequence is base paired and inaccessible to ribosomes. During transcription, a metastable hairpin is formed first, sequestering part of the SD-containing hairpin. This keeps the SD sequence accessible for ribosomes. After some time this metastable structure is disrupted in favour of the translationally silent *mfe*-structure [74].

Parameters

- Energy range: $6kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 100

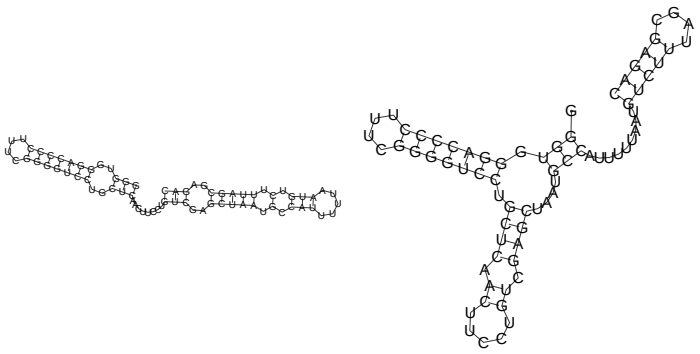
Distance plots



(a) d_{MD} (b) d_{TAD}

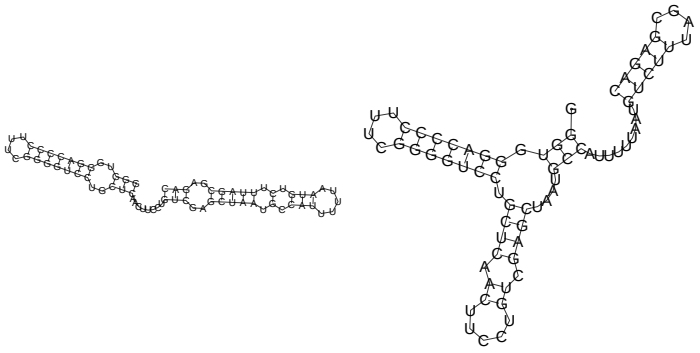
Figure A.16: Distance plots for ms2 5'-UTR.

Consensus structures



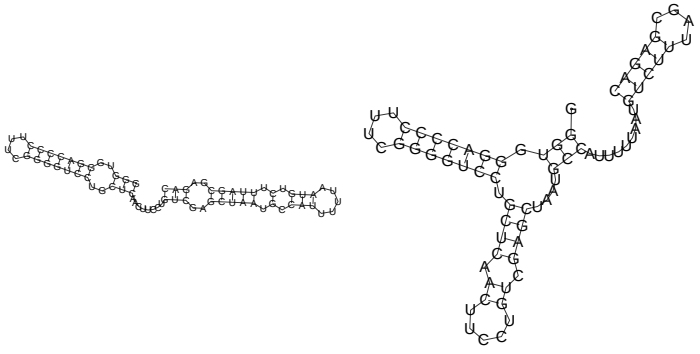
(a) c_1 (b) c_2

Figure A.17: Consensus structures for ms2 5'-UTR based on d_{MD}.



(a) c_1 (b) c_2

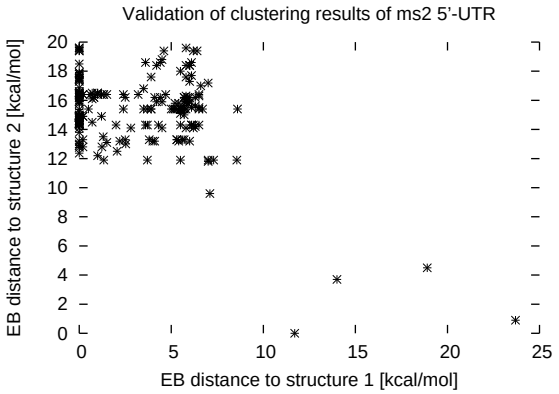
Figure A.18: Consensus structures for ms2 5'-UTR based on d_{TAD}.



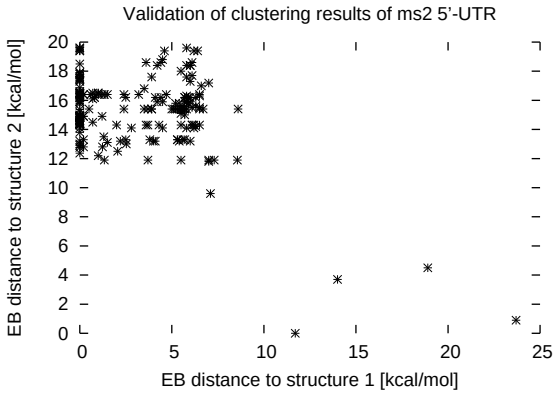
(a) c_1 (b) c_2

Figure A.19: Consensus structures for ms2 5'-UTR based on d_{EB}.

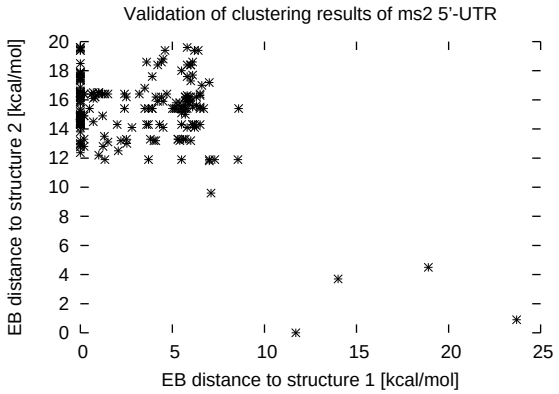
Consensus structure validation



(a) d_{MD}, $pkMeasure = 15$



(b) d_{TAD}, $pkMeasure = 15$



(c) d_{EB}, $pkMeasure = 15$

Figure A.20: Validation plots for ms2 5'-UTR.

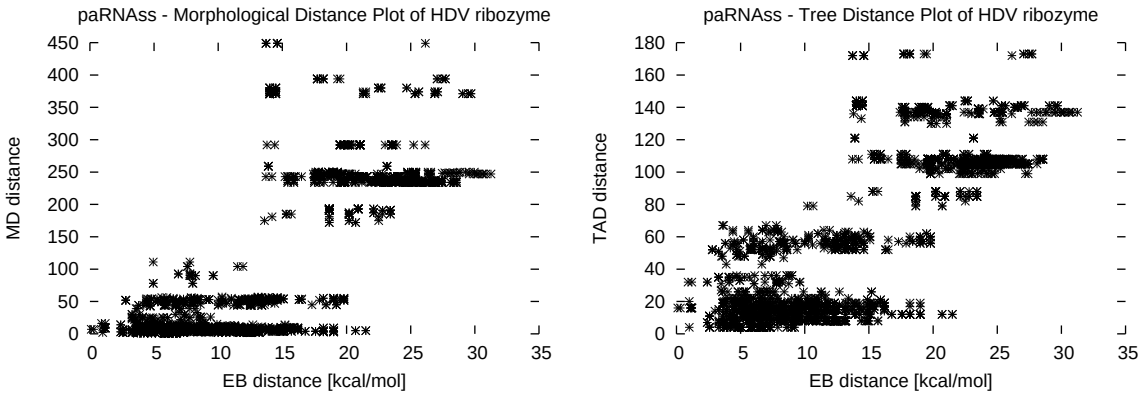
A.5 HDV ribozyme

Function Hepatitis delta virus (HDV) is a small single-stranded RNA satellite of hepatitis B virus. During the rolling-circle replication of the genome multimers are formed, which need to be cleaved into monomers. This cleavage is performed by the RNA itself, more precisely by the HDV ribozyme.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

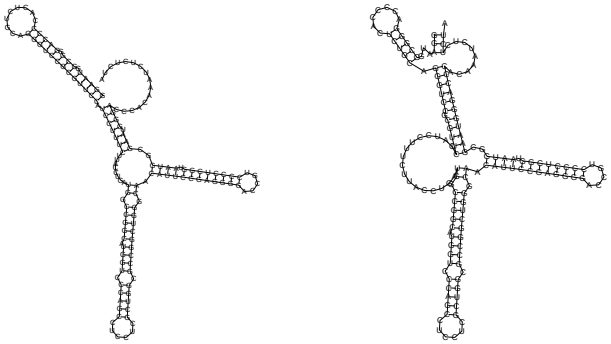
Distance plots



(a) d_{MD} (b) d_{TAD}

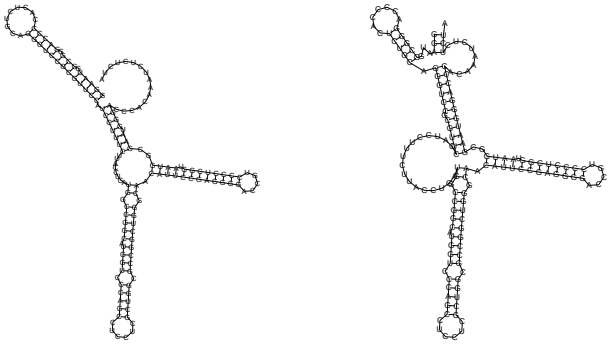
Figure A.21: Distance plots for HDV ribozyme.

Consensus structures



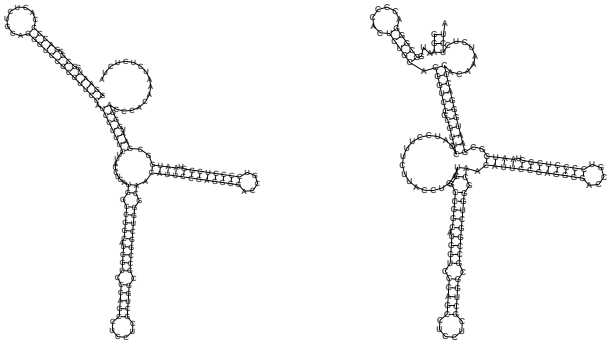
(a) c_1 (b) c_2

Figure A.22: Consensus structures for HDV ribozyme based on d_{MD}.



(a) c_1 (b) c_2

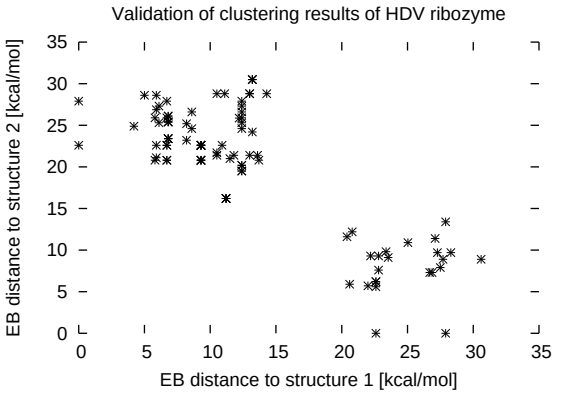
Figure A.23: Consensus structures for HDV ribozyme based on d_{TAD}.



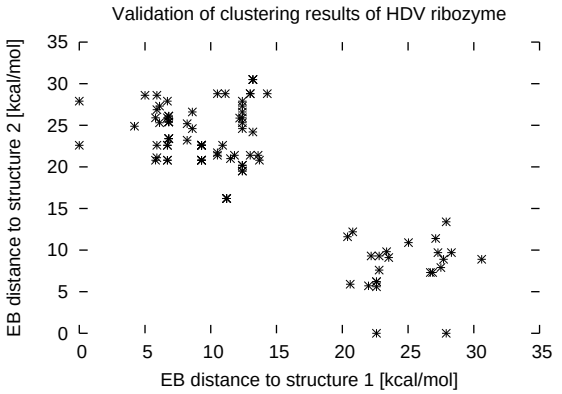
(a) c_1 (b) c_2

Figure A.24: Consensus structures for HDV ribozyme based on d_{EB}.

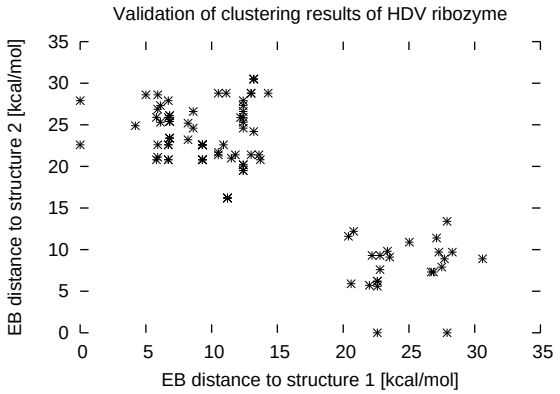
Consensus structure validation



(a) d_{MD}, $pkMeasure = 25$



(b) d_{TAD}, $pkMeasure = 25$



(c) d_{EB}, $pkMeasure = 25$

Figure A.25: Validation plots for HDV ribozyme.

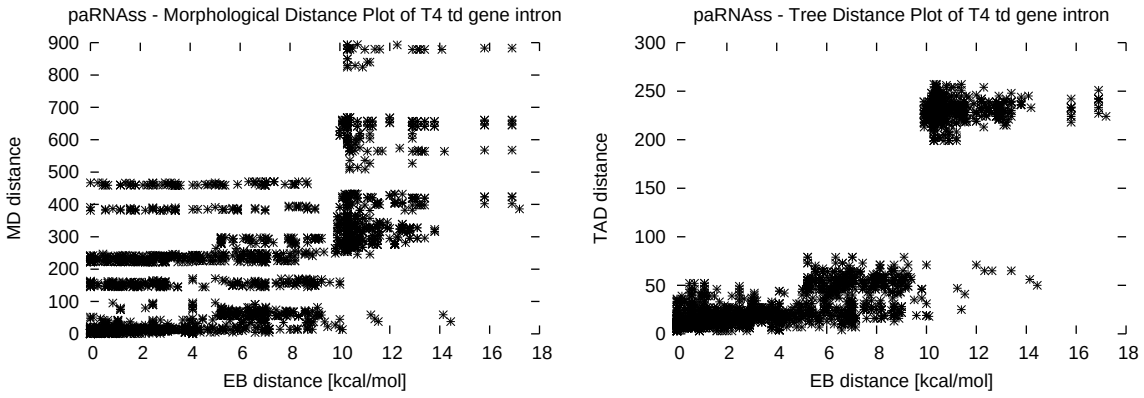
A.6 T4 td gene intron

Function Another member of the family of group I introns is the T4 *thymidylate synthase* (td) gene intron. In this system the ribosome or several RNA chaperones function as a trans-acting factor by disrupting a splicing preventing interaction between the exon and intron. Now the intron can refold into its active conformation, thus leading to an efficient splicing reaction [75].

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

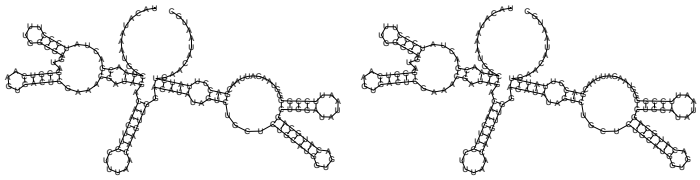
Distance plots



(a) d_{MD} (b) d_{TAD}

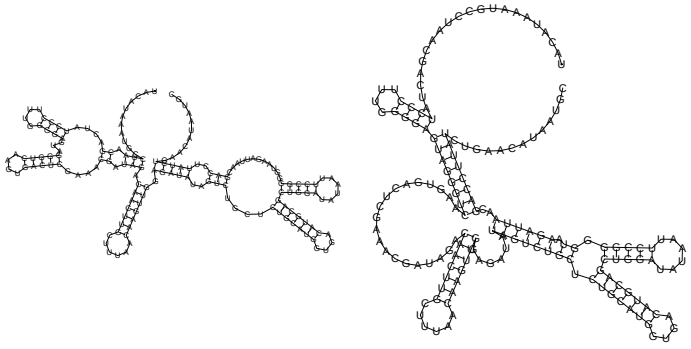
Figure A.26: Distance plots for T4 td gene intron.

Consensus structures



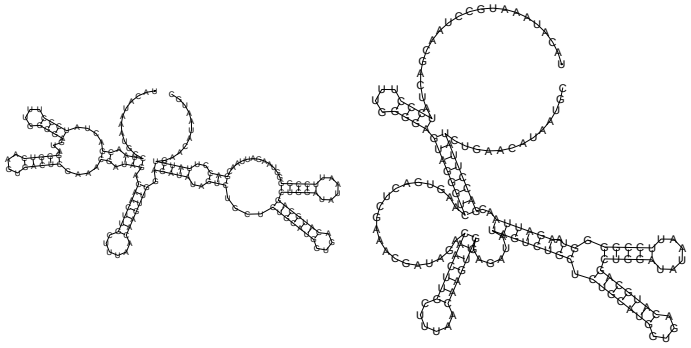
(a) c_1 (b) c_2

Figure A.27: Consensus structures for T4 td gene intron based on d_{MD}.



(a) c_1 (b) c_2

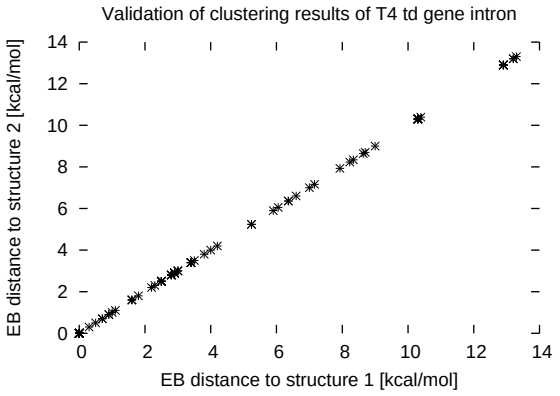
Figure A.28: Consensus structures for T4 td gene intron based on d_{TAD}.



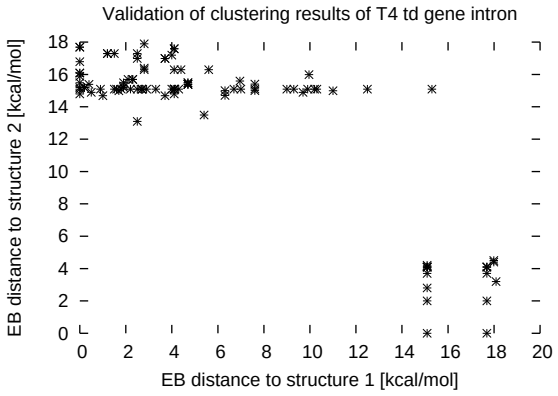
(a) c_1 (b) c_2

Figure A.29: Consensus structures for T4 td gene intron based on d_{EB}.

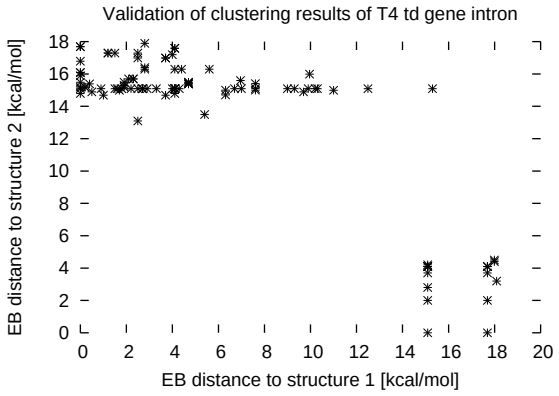
Consensus structure validation



(a) d_{MD}, $pkMeasure = -1$



(b) d_{TAD}, $pkMeasure = 14$



(c) d_{EB}, $pkMeasure = 14$

Figure A.30: Validation plots for T4 td gene intron.

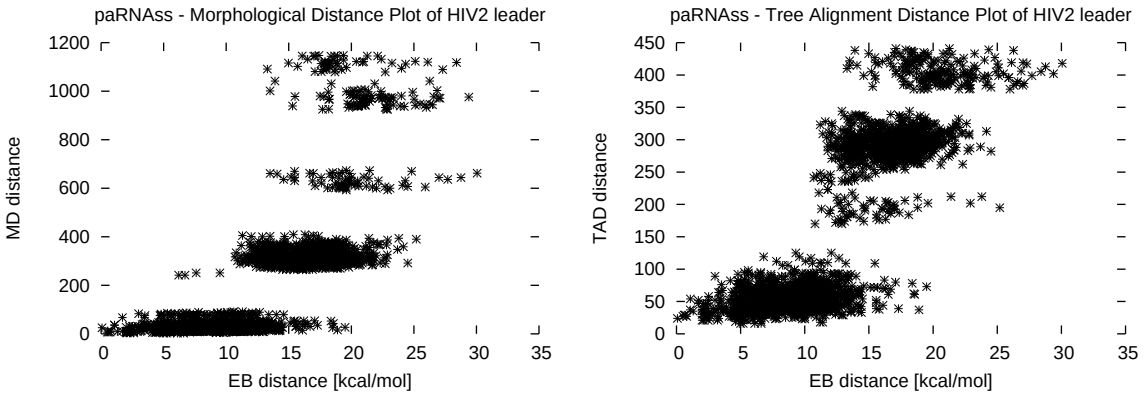
A.7 HIV-2 leader

Function Similar to the previously described HIV-1 leader, also the HIV-2 leader is capable of attaining two different conformations, which reflect different functions. One conformation enables translation of the viral genes, whereas the other is needed for dimerization of the viral genome and subsequent packaging [76].

Parameters

- Energy range: $2kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

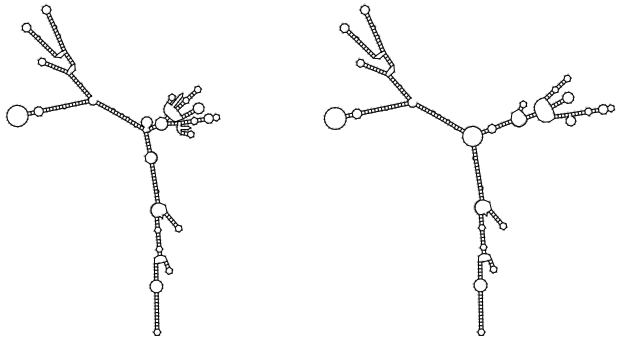
Distance plots



(a) d_{MD} (b) d_{TAD}

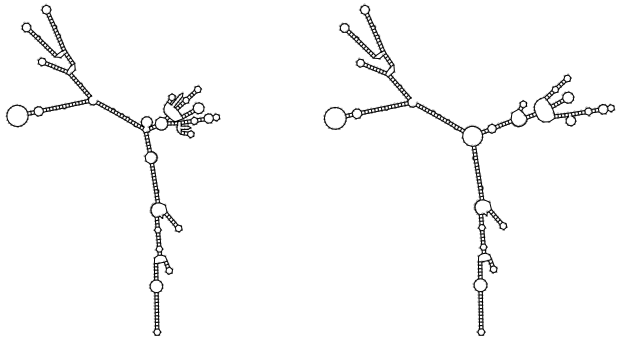
Figure A.31: Distance plots for HIV2 leader.

Consensus structures



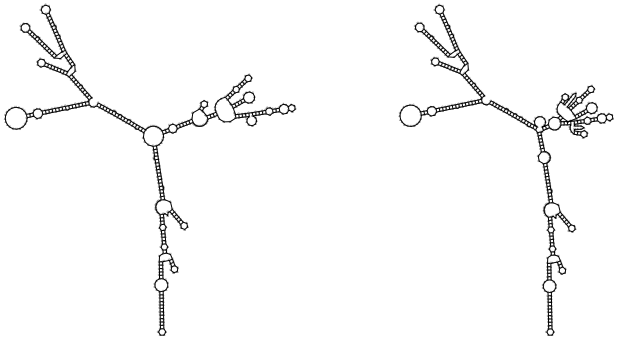
(a) c_1 (b) c_2

Figure A.32: Consensus structures for HIV2 leader based on d_{MD}.



(a) c_1 (b) c_2

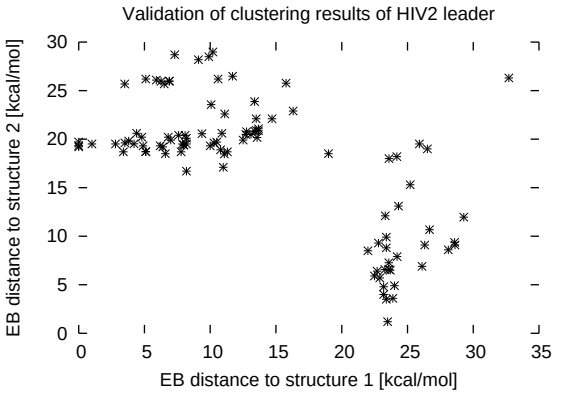
Figure A.33: Consensus structures for HIV2 leader based on d_{TAD}.



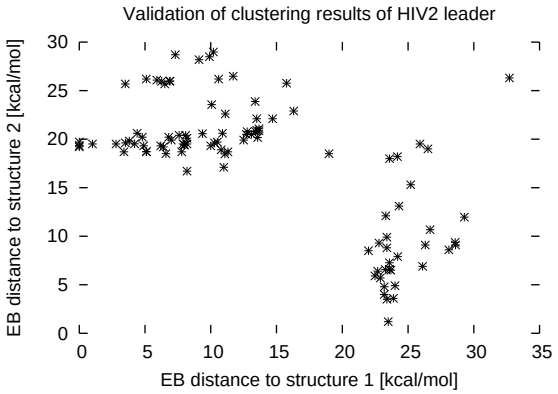
(a) c_1 (b) c_2

Figure A.34: Consensus structures for HIV2 leader based on d_{EB}.

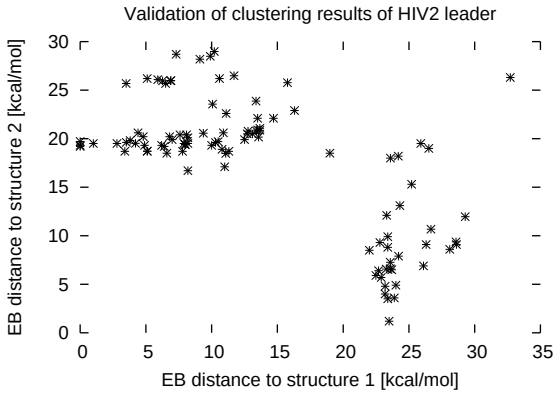
Consensus structure validation



(a) d_{MD}, $pkMeasure = 35$



(b) d_{TAD}, $pkMeasure = 35$



(c) d_{EB}, $pkMeasure = 35$

Figure A.35: Validation plots for HIV2 leader.

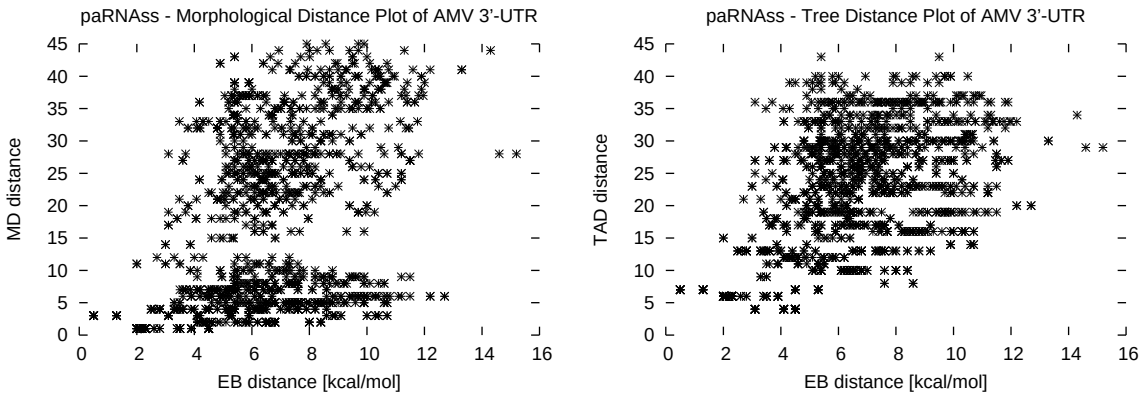
A.8 3'-UTR of alfalfa mosaic virus (AMV) RNA

3'-UTRs of alfamo-virus RNAs fold into a series of stem-loop structures to which the coat protein binds with high affinity. This binding plays a role in initiation of infection and has been thought to substitute for a tRNA-like structure that is found at the 3' termini of related plant viruses. Olsthoorn et al. [77] propose the existence of an alternative conformation of the 3' ends of alfamo-virus RNAs, including a pseudoknot, and that these two conformations could enable the virus to switch from translation to replication, like mentioned before for the HIV-1 and HIV-2 leaders.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

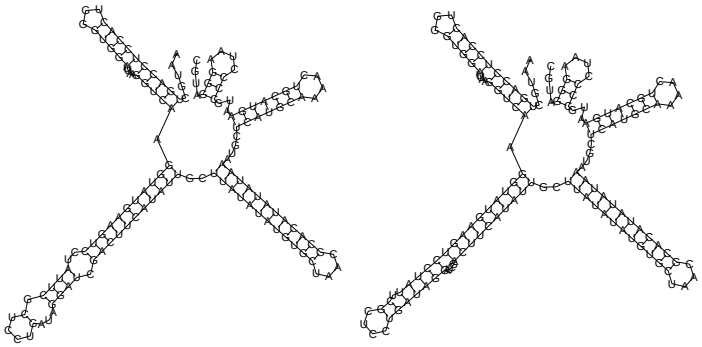
Distance plots



(a) d_{MD} (b) d_{TAD}

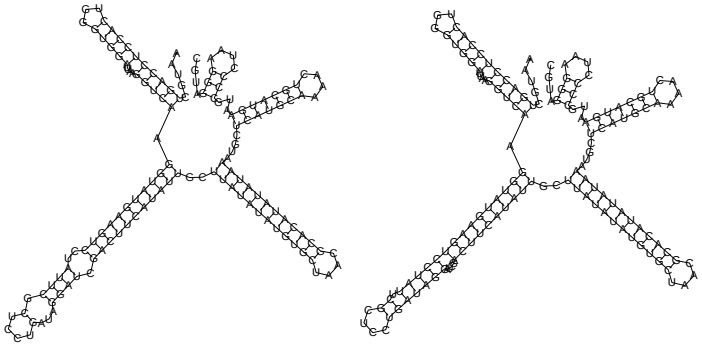
Figure A.36: Distance plots for AMV 3'-UTR.

Consensus structures



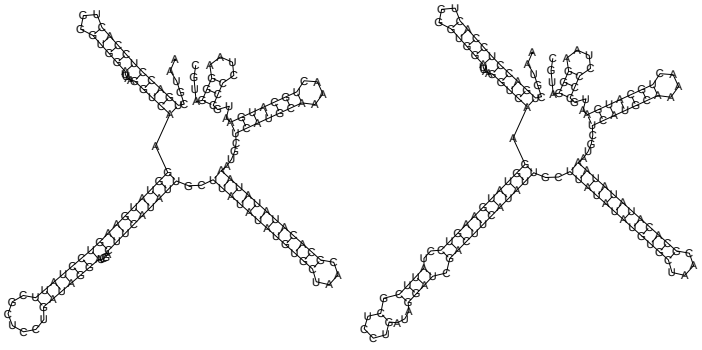
(a) c_1 (b) c_2

Figure A.37: Consensus structures for AMV 3'-UTR based on d_{MD}.



(a) c_1 (b) c_2

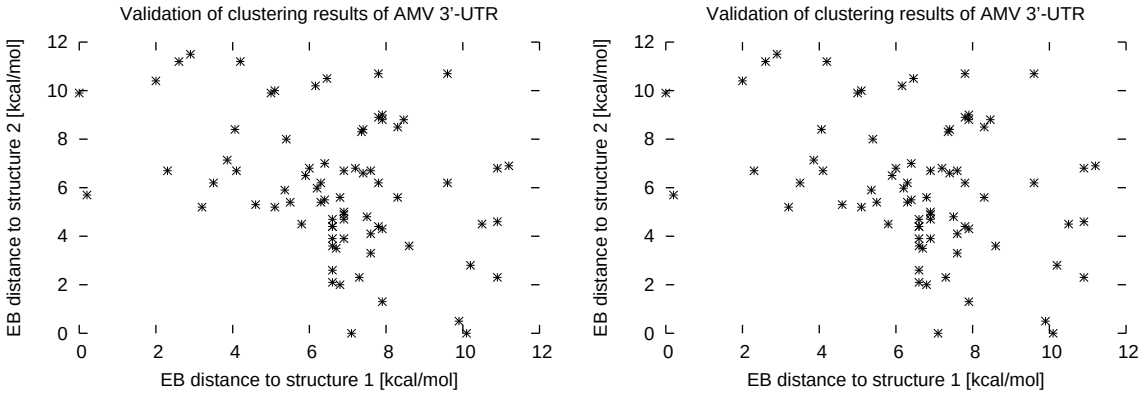
Figure A.38: Consensus structures for AMV 3'-UTR based on d_{TAD}.



(a) c_1 (b) c_2

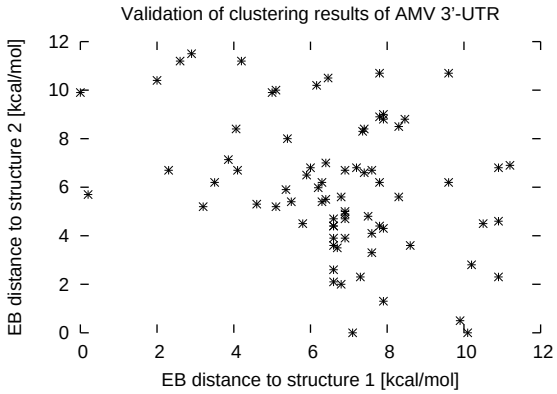
Figure A.39: Consensus structures for AMV 3'-UTR based on d_{EB}.

Consensus structure validation



(a) d_{MD}, $pkMeasure = 7$

(b) d_{TAD}, $pkMeasure = 7$



(c) d_{EB}, $pkMeasure = 7$

Figure A.40: Validation plots for AMV 3'-UTR.

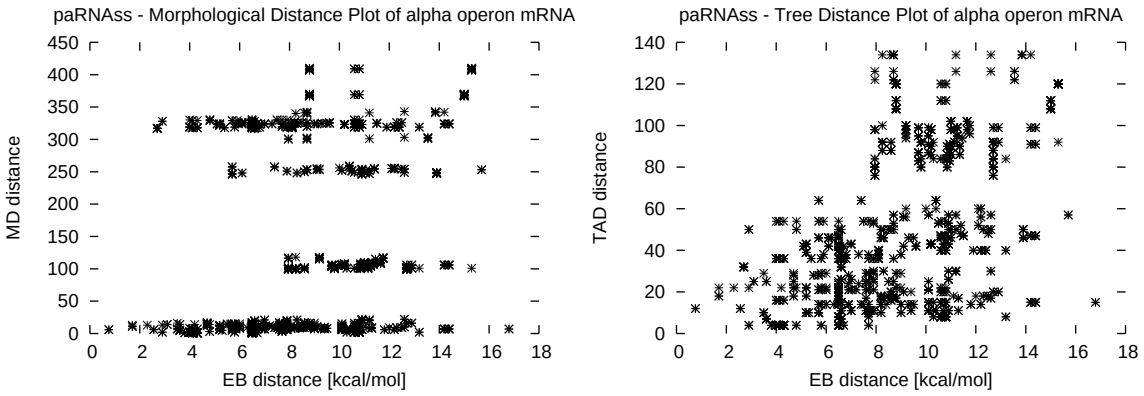
A.9 E. coli α operon

Function Ribosomal protein S4 represses synthesis of the four ribosomal proteins (including itself) in the *Escherichia coli* α operon by binding to a pseudoknot structure that spans the ribosome binding site. One feature of the repression mechanism is, that the mRNA switches between conformations that are "active" or "inactive" in translation, with S4 as an allosteric effector of the inactive form. Such bound S4 is capable of holding the 30S ribosomal subunit in an unproductive complex on the mRNA [78].

Parameters

- Energy range: $4kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

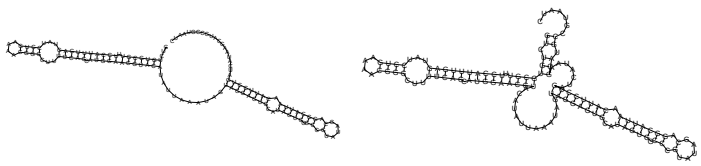
Distance plots



(a) d_{MD} (b) d_{TAD}

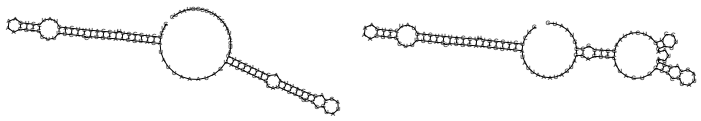
Figure A.41: Distance plots for α operon mRNA.

Consensus structures



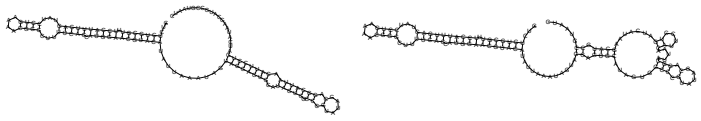
(a) c_1 (b) c_2

Figure A.42: Consensus structures for α operon mRNA based on d_{MD}.



(a) c_1 (b) c_2

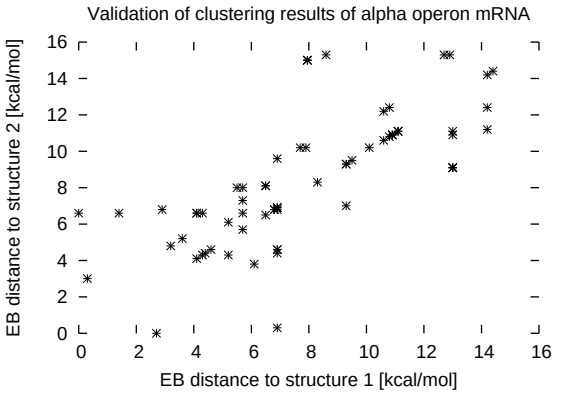
Figure A.43: Consensus structures for α operon mRNA based on d_{TAD}.



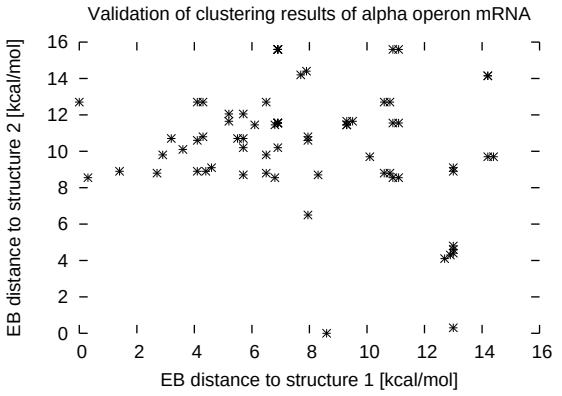
(a) c_1 (b) c_2

Figure A.44: Consensus structures for α operon mRNA based on d_{EB}.

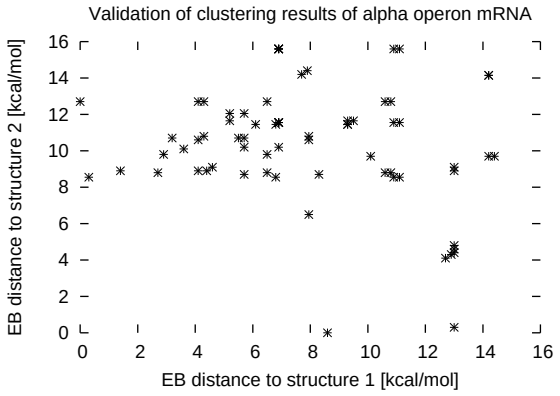
Consensus structure validation



(a) d_{MD}, $pkMeasure = 3$



(b) d_{TAD}, $pkMeasure = 10$



(c) d_{EB}, $pkMeasure = 10$

Figure A.45: Validation plots for α operon mRNA.

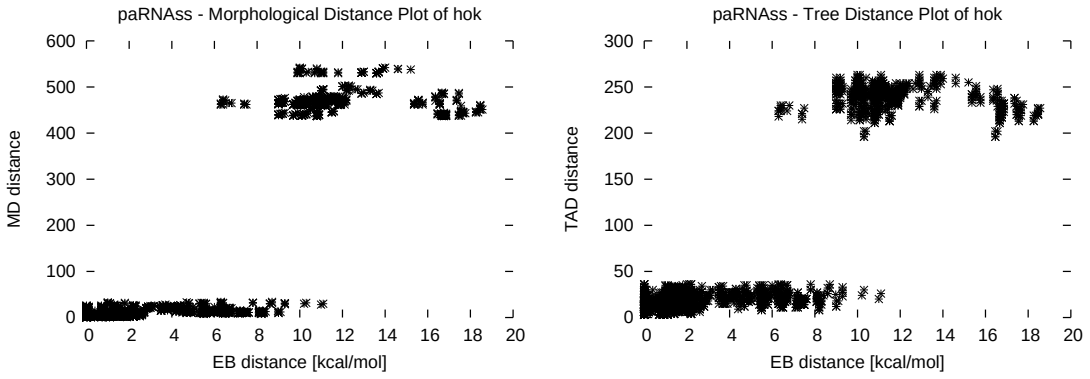
A.10 E. coli hok

Function The R1 plasmid is maintained in the *E. coli* cell by expressing a toxin, which kills the cell if the plasmid is lost. The expression of the *host killing* toxin (Hok) must be controlled in all stages of the lifecycle of the *hok* mRNA. This is achieved in part by the antisense RNA *suppression of killing* (Sok), which is transcribed from the same plasmid R1. The antisense Sok RNA is labile, so that the pool is quickly depleted when the plasmid is lost, which leads to the killing of plasmid-free cells. This also implies that the *hok* mRNA must be considerably more stable than the Sok RNA. Furthermore, it should not be target of the Sok RNA in plasmid-containing cells to permit the formation of a pool of the *hok* mRNA, which is large enough to kill plasmid-free cells. And, last but not least, it should not be translationally active in plasmid-containing cells to avoid premature killing of cells and degradation by duplex formation with Sok RNA. This is achieved by forming a stable and highly structured mRNA, which is translationally inactive. This inactive *hok* mRNA then gets slowly activated by a 3'-end processing, which causes a structural rearrangement. This leads to the formation of the *translation activator hairpin* (*tac*) at the 5'-end. This partially opens the ribosomal entry site and, simultaneously, the target site for the antisense Sok RNA. In plasmid-containing cells, the Sok RNA binds to the active messenger forming a duplex, which gets degraded by RNase III, while in plasmid-free cells, the toxic Hok protein gets expressed [79].

Parameters

- Energy range: $2 kcal/mol$
- Search space: canonicals
- Temperature: 37 °C
- Max. Structures: 50

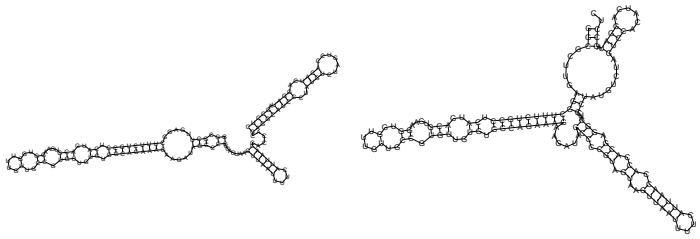
Distance plots



(a) d_{MD} (b) d_{TAD}

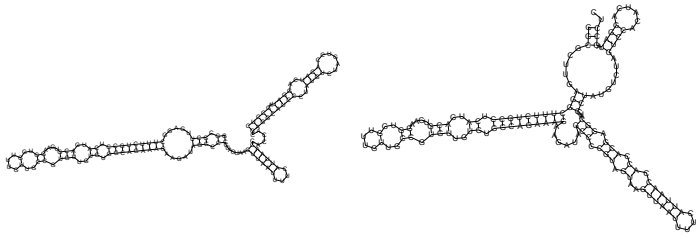
Figure A.46: Distance plots for hok.

Consensus structures



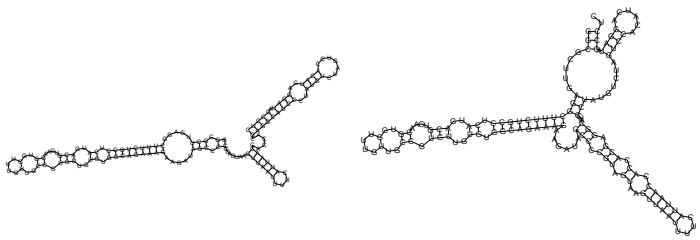
(a) c_1 (b) c_2

Figure A.47: Consensus structures for hok based on d_{MD}.



(a) c_1 (b) c_2

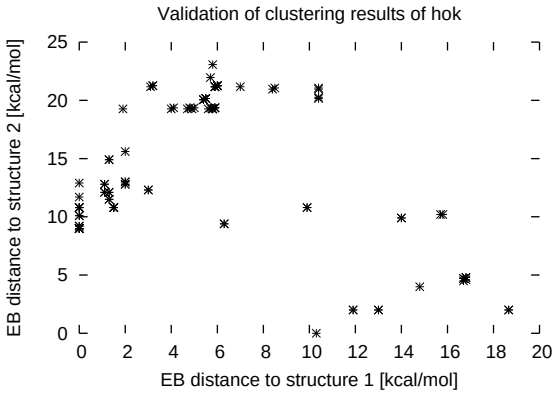
Figure A.48: Consensus structures for hok based on d_{TAD}.



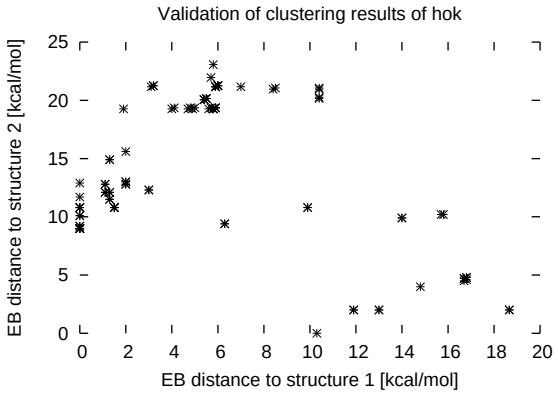
(a) c_1 (b) c_2

Figure A.49: Consensus structures for hok based on d_{EB}.

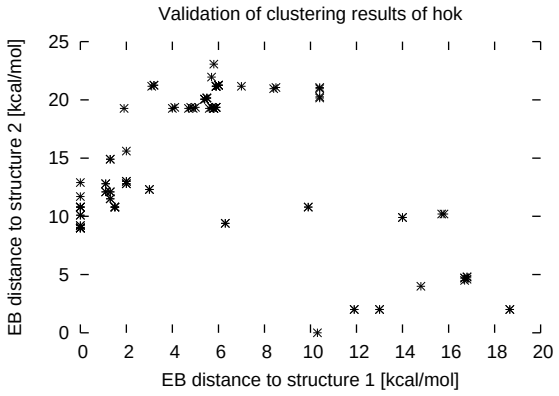
Consensus structure validation



(a) d_{MD}, $pkMeasure = 17$



(b) d_{TAD}, $pkMeasure = 17$



(c) d_{EB}, $pkMeasure = 17$

Figure A.50: Validation plots for hok.

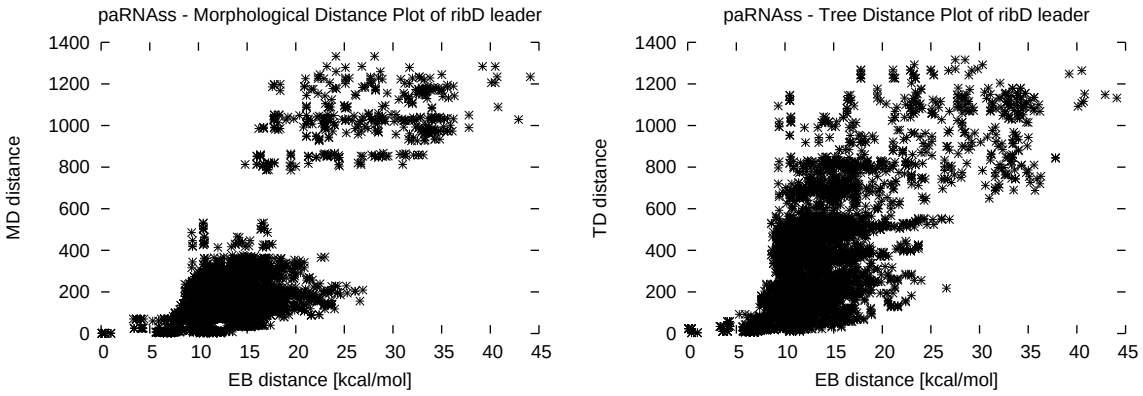
A.11 B. subtilis ribD leader

Function mRNAs of prokaryotic genes required for the biosynthesis of riboflavin and flavin mononucleotide (FMN) share a highly conserved but distinct RNA domain, termed the *RFN* element. FMN is required for down-regulation of the ribD operon of *Bacillus subtilis*, which encodes several FMN biosynthetic enzymes. Winkler et al. [52] showed that FMN directly binds to the *RFN* element of the *ribD* leader and causes transcription termination.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 100

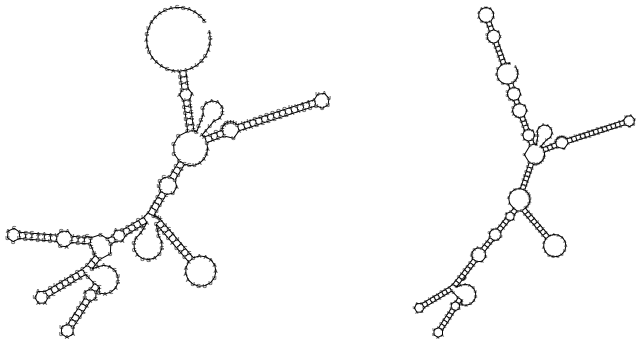
Distance plots



(a) d_{MD} (b) d_{TAD}

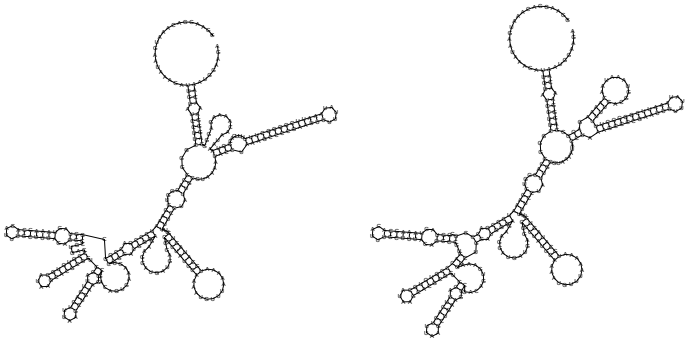
Figure A.51: Distance plots for ribD leader.

Consensus structures



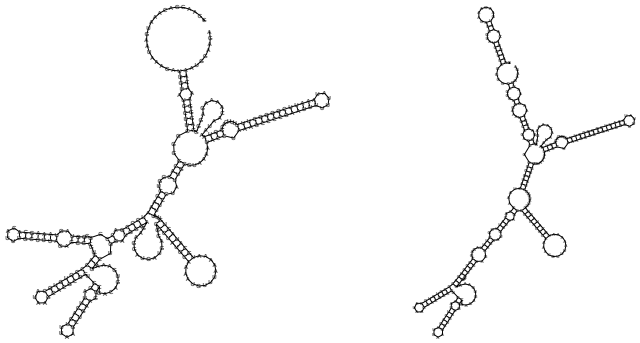
(a) c_1 (b) c_2

Figure A.52: Consensus structures for ribD leader based on d_{MD}.



(a) c_1 (b) c_2

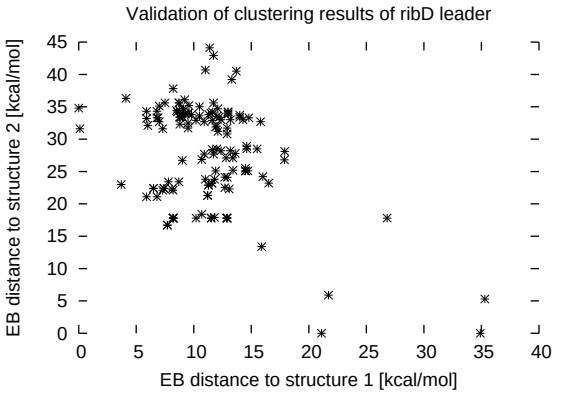
Figure A.53: Consensus structures for ribD leader based on d_{TAD}.



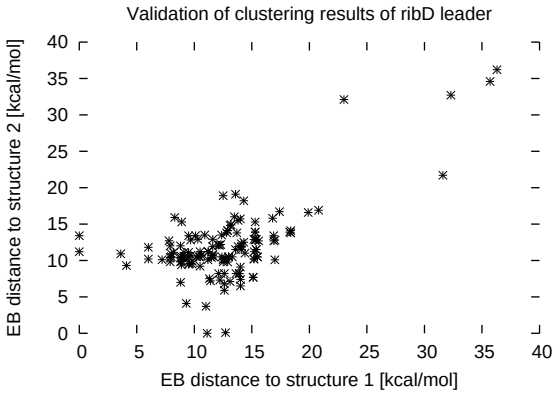
(a) c_1 (b) c_2

Figure A.54: Consensus structures for ribD leader based on d_{EB}.

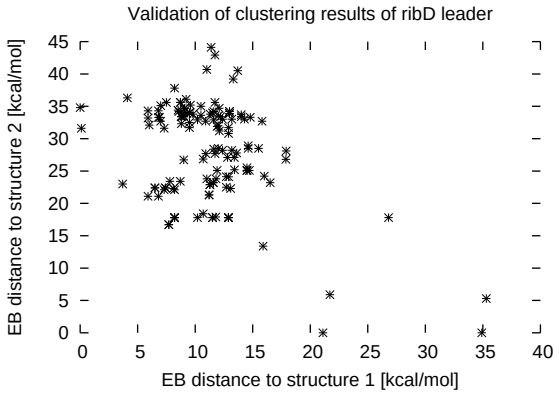
Consensus structure validation



(a) d_{MD}, $pkMeasure = 45$



(b) d_{TAD}, $pkMeasure = 5$



(c) d_{EB}, $pkMeasure = 45$

Figure A.55: Validation plots for ribD leader.

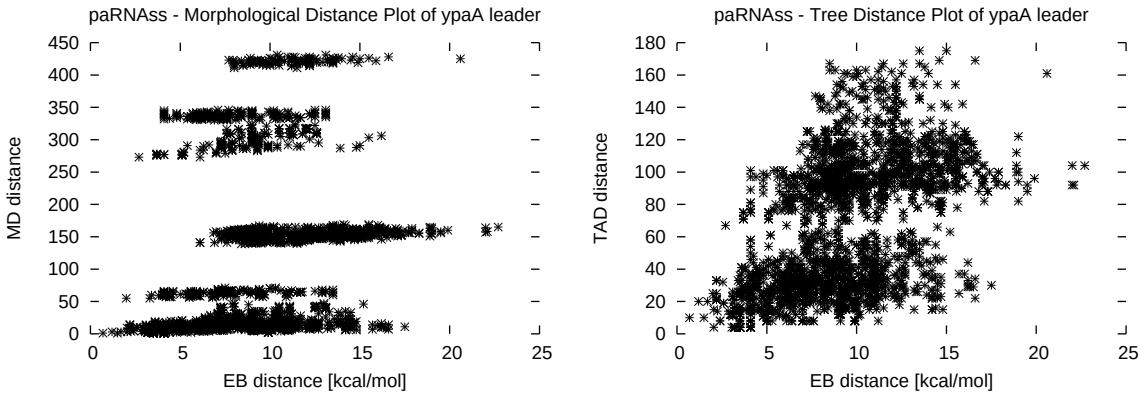
A.12 B. subtilis ypaA leader

Function In the same analysis as in the previous section the authors also showed that the mRNA of *B. subtilis ypaA* (a putative riboflavin transport protein) carries an *RFN* element. Binding of flavinmononucleotide to the *RFN*-containing leader of *ypaA* mRNA leads to down-regulation. This time the mechanism is not transcriptional termination as for the *ribD* leader but sequestration of the ribosome-binding site.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

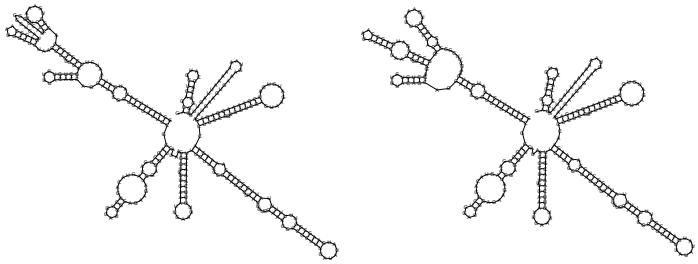
Distance plots



(a) d_{MD} (b) d_{TAD}

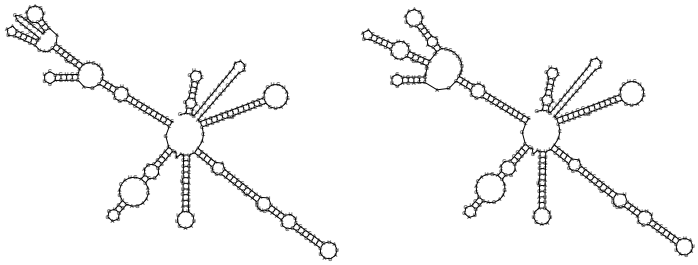
Figure A.56: Distance plots for ypaA leader.

Consensus structures



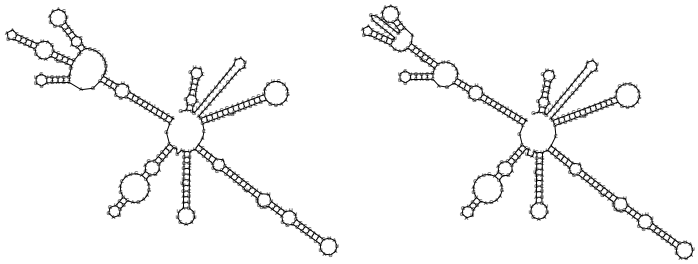
(a) c_1 (b) c_2

Figure A.57: Consensus structures for ypaA leader based on d_{MD}.



(a) c_1 (b) c_2

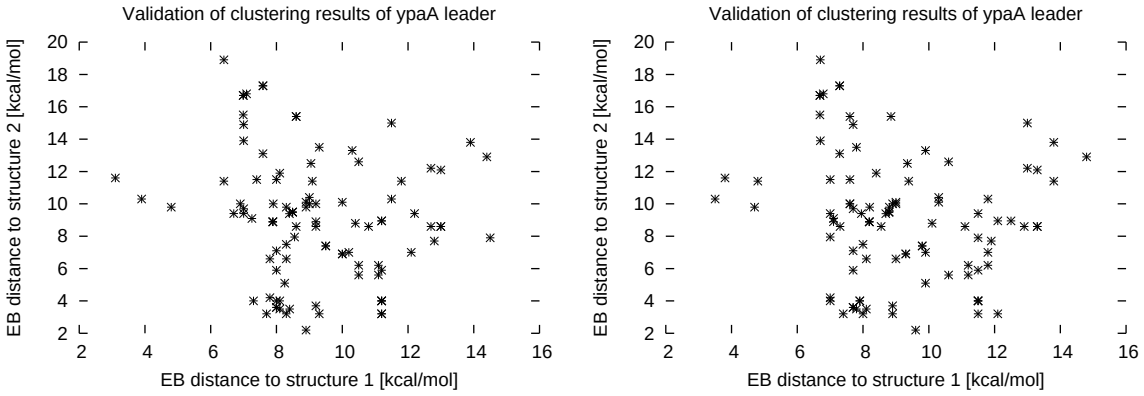
Figure A.58: Consensus structures for ypaA leader based on d_{TAD}.



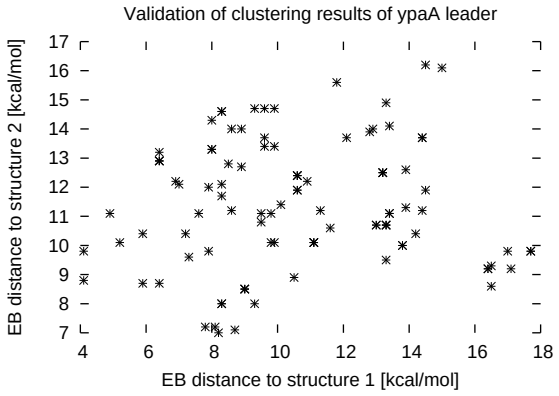
(a) c_1 (b) c_2

Figure A.59: Consensus structures for ypaA leader based on d_{EB}.

Consensus structure validation



(a) d_{MD}, $pkMeasure = 8$ (b) d_{TAD}, $pkMeasure = 8$



(c) d_{EB}, $pkMeasure = 8$

Figure A.60: Validation plots for ypaA leader.

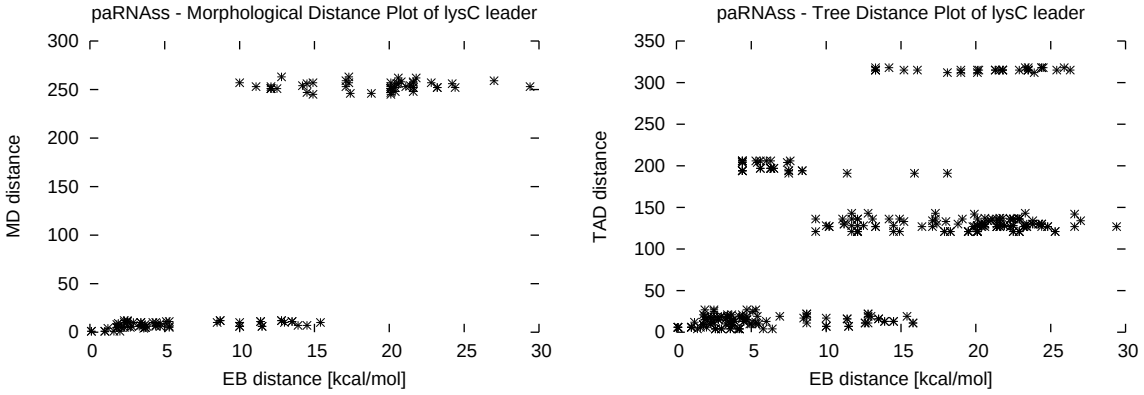
A.13 E. coli lysC leader

Function Rodionov et al. [80] propose another regulatory RNA element involved in amino acid biosynthetic pathways. They identified a lysine-specific RNA element, named the *LYS* element, in the regulatory regions of bacterial genes involved in biosynthesis and transport of lysine. The authors propose that lysine regulates expression in Gram-positive bacteria by premature termination of transcription in the leader. Additionally, they propose that for Gram-negative (*E. coli*) bacteria, binding of lysine to the *LYS* element leads to sequestration of the ribosome binding site by a hairpin structure which represses initiation of translation.

Parameters

- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

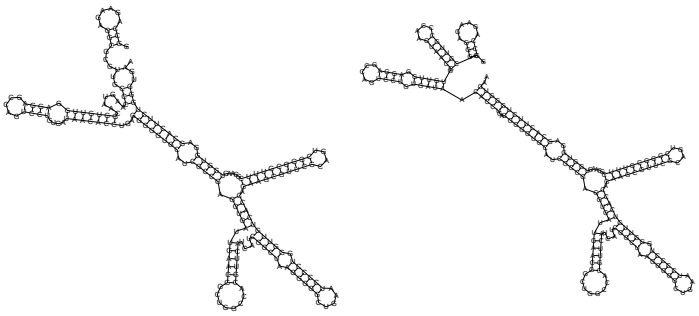
Distance plots



(a) d_{MD} (b) d_{TAD}

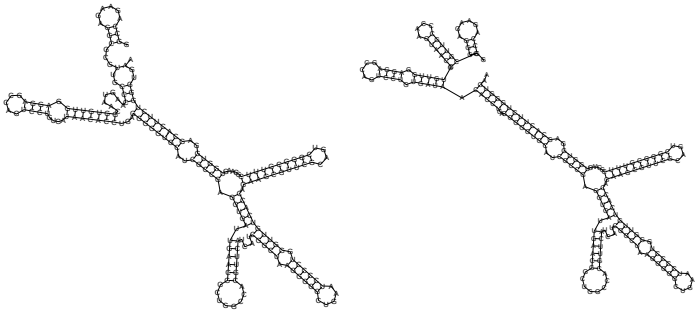
Figure A.61: Distance plots for lysC leader.

Consensus structures



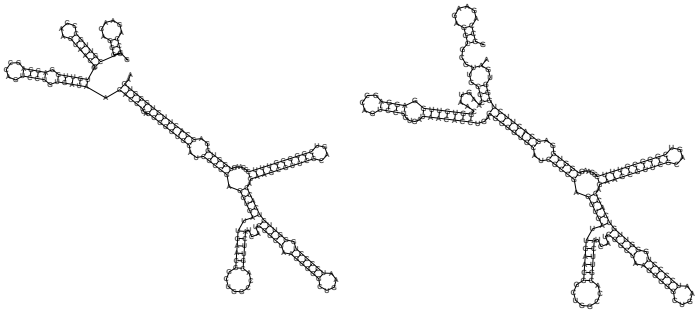
(a) c_1 (b) c_2

Figure A.62: Consensus structures for lysC leader based on d_{MD}.



(a) c_1 (b) c_2

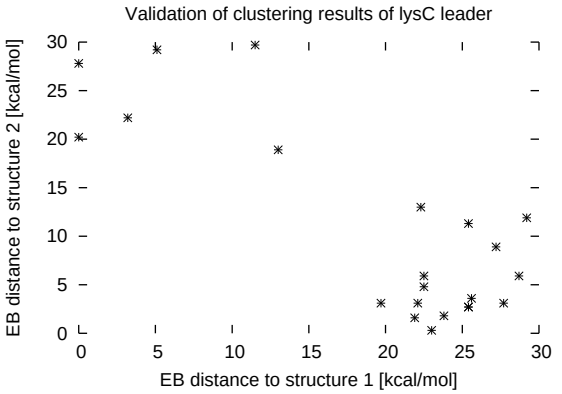
Figure A.63: Consensus structures for lysC leader based on d_{TAD}.



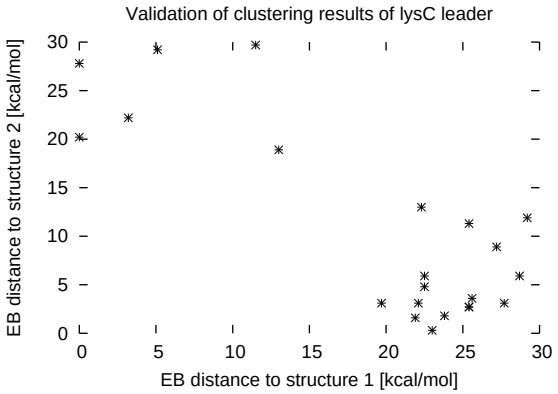
(a) c_1 (b) c_2

Figure A.64: Consensus structures for lysC leader based on d_{EB}.

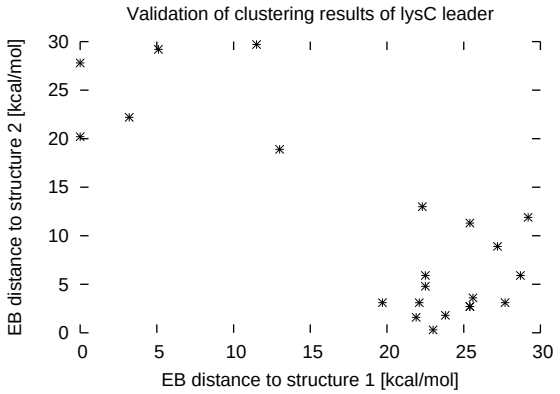
Consensus structure validation



(a) d_{MD}, $pkMeasure = 17$



(b) d_{TAD}, $pkMeasure = 17$



(c) d_{EB}, $pkMeasure = 17$

Figure A.65: Validation plots for lysC leader.

A.14 Internal transcribed spacer of pre-rRNAs in yeast

Function Maturation of the large subunit rRNAs requires a series of cleavages that result in removal of the internal transcribed spacer (ITS2), which separates mature 5.8S and 25/28S rRNAs. Formation of higher order secondary structure is a prerequisite for accurate and efficient pre-rRNA processing. Two alternative secondary structure models exist for *Saccharomyces cerevisiae* ITS2, namely the "hairpin model" and the "ring model". Côtè et al. [81] examined the significance of both models in efficient processing and propose a dynamic conformational model for the role of ITS2: ITS2 initially folds into the "ring" structure. This may promote the association of proteins that further stabilise the structure and initiate formation of the preprocessing complex. Once successfully assembled, the preprocessing complex may induce a conformational transition resulting in formation of the "hairpin" structure. This structure could be shown to be a prerequisite for maturation of both 5.8S and 25S rRNAs.

Parameters

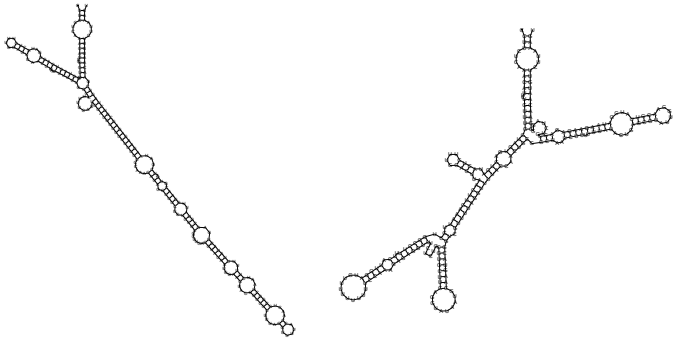
- Energy range: $3kcal/mol$
- Search space: canonicals (refolded)
- Temperature: 37 °C
- Max. Structures: 50

Distance plots

(a) d_{MD} (b) d_{TAD}

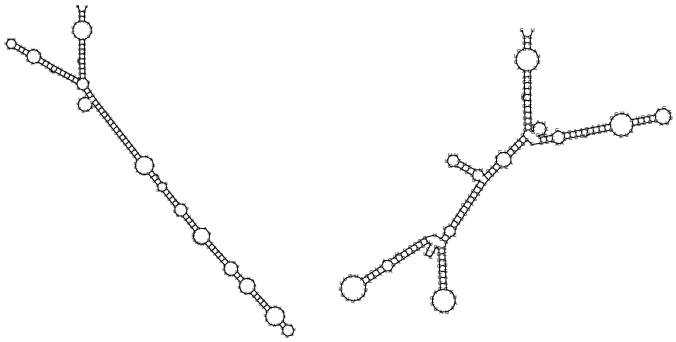
Figure A.66: Distance plots for yeast ITS2.

Consensus structures



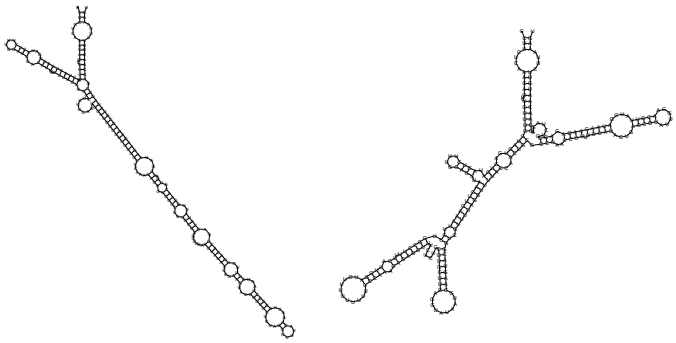
(a) c_1 (b) c_2

Figure A.67: Consensus structures for yeast ITS2 based on d_{MD}.



(a) c_1 (b) c_2

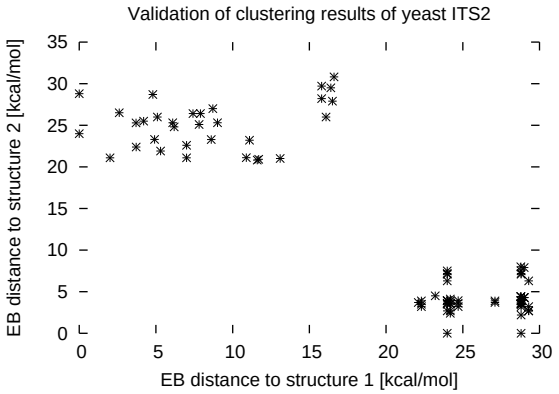
Figure A.68: Consensus structures for yeast ITS2 based on d_{TAD}.



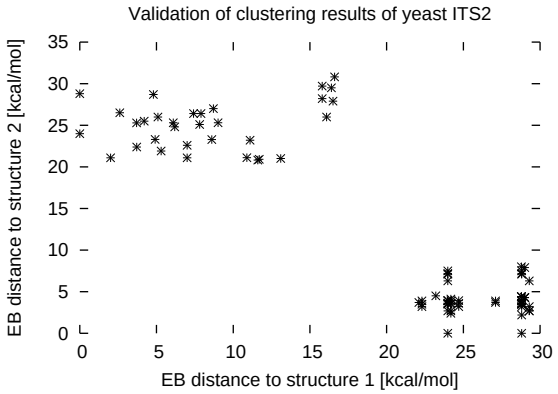
(a) c_1 (b) c_2

Figure A.69: Consensus structures for yeast ITS2 based on d_{EB}.

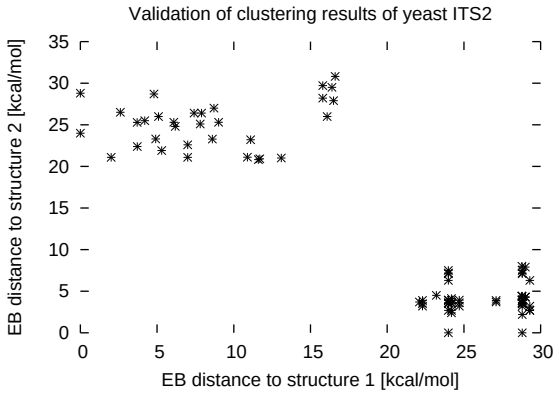
Consensus structure validation



(a) d_{MD}, $pkMeasure = 76$



(b) d_{TAD}, $pkMeasure = 76$



(c) d_{EB}, $pkMeasure = 76$

Figure A.70: Validation plots for yeast ITS2.

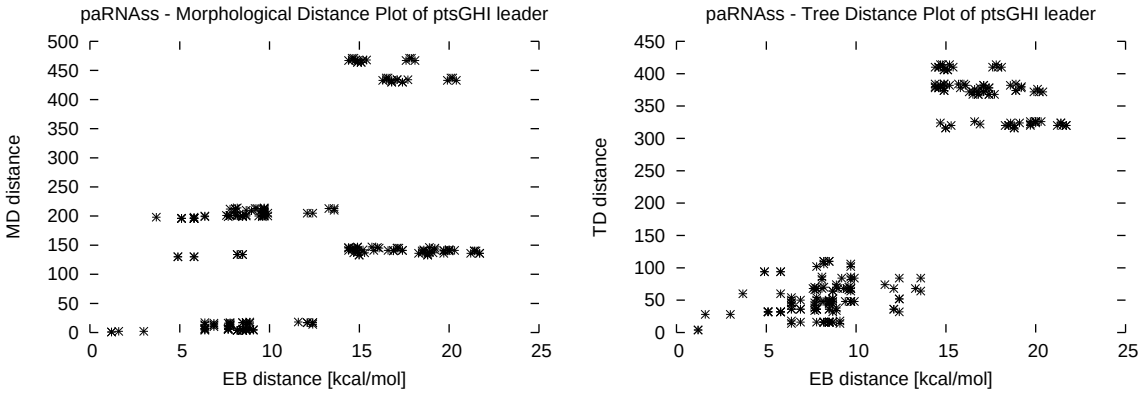
A.15 Leader of ptsGHI operon in B. subtilis

Function The *ptsGHI* operon in *B. subtilis* encodes for the genes involved in glucose transport by the phosphotransferase system. Schilling et al. [82] showed that expression of this operon is controlled at the level of transcript elongation by a protein-dependent riboswitch. In the absence of glucose a transcriptional terminator prevents elongation into the structural genes. In the presence of glucose, the GlcT protein is activated and binds and stabilises an alternative structure that overlaps the terminator and prevents termination.

Parameters

- Energy range: $2kcal/mol$
- Search space: feasibles (refolded)
- Temperature: 37 °C
- Max. Structures: 50

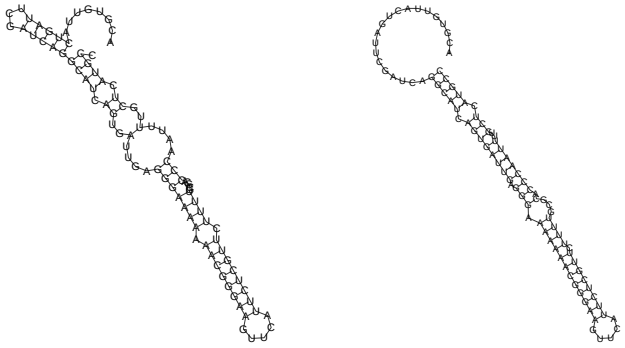
Distance plots



(a) d_{MD} (b) d_{TAD}

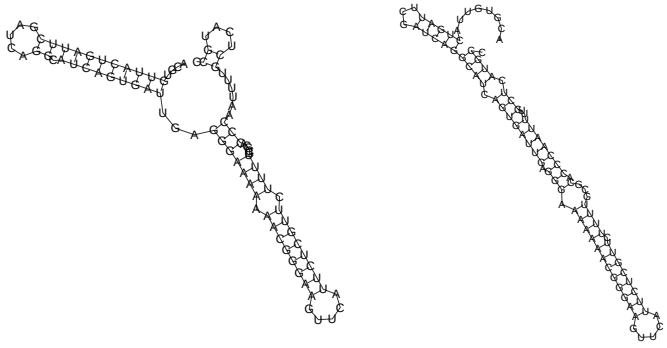
Figure A.71: Distance plots for leader of ptsGHI.

Consensus structures



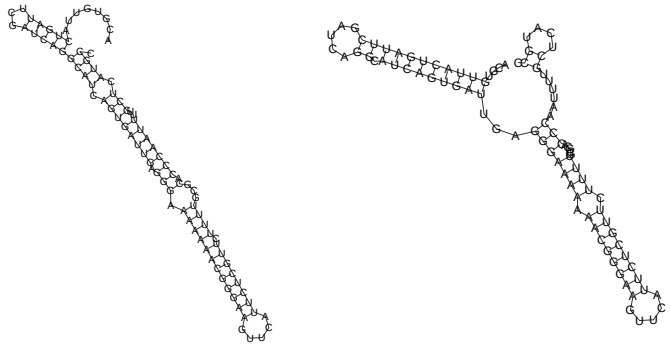
(a) c_1 (b) c_2

Figure A.72: Consensus structures for leader of ptsGHI based on d_{MD}.



(a) c_1 (b) c_2

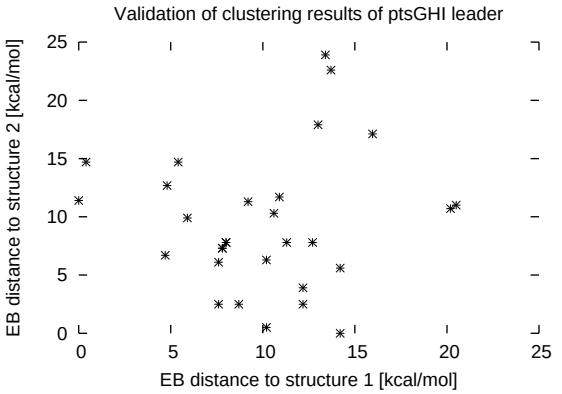
Figure A.73: Consensus structures for leader of ptsGHI based on d_{TAD}.



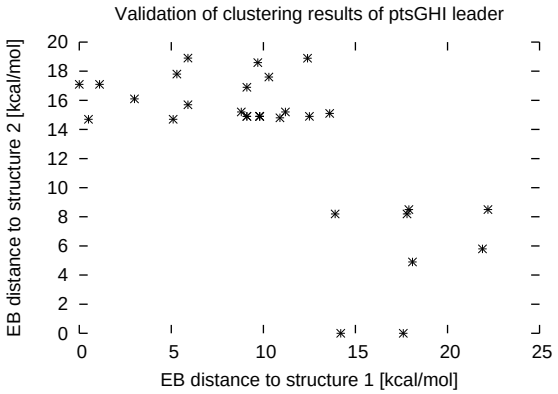
(a) c_1 (b) c_2

Figure A.74: Consensus structures for leader of ptsGHI based on d_{EB}.

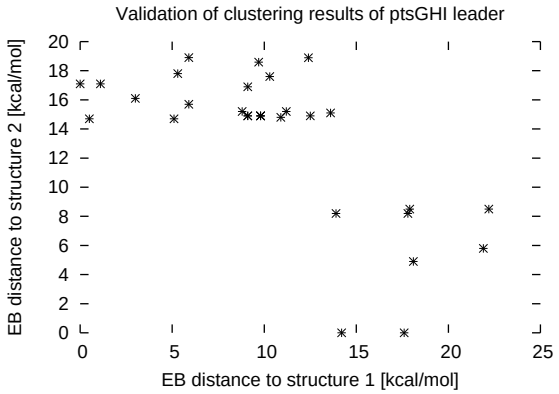
Consensus structure validation



(a) d_{MD}, $pkMeasure = 15$



(b) d_{TAD}, $pkMeasure = 32$



(c) d_{EB}, $pkMeasure = 32$

Figure A.75: Validation plots for leader of ptsGHI.

APPENDIX B

Grammars and Algebras for RNA Folding

B.1 Grammars for Prediction of Canonical RNA Structures

The following grammar, termed `canonicals`, describes the folding space of RNA with dangling bases and is restricted to canonical structures, i.e. structures without isolated base pairs.

```
> canonicals takes alg inp_tr = axiom struct where
>
>   (sadd,cadd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>    ml,mldr,mldlr,mldl,addss,ssadd,cons,ul,combine,h,h_i,
>    h_l,h_s) = alg baseArray takes

>   struct        = listed (
>                   sadd <<< base    -~~ struct |||
>                   cadd <<< edangle ~~~ struct |||
>                   nil  <<< empty              ... h_s)

>   edangle       = edl  <<< base -~~ initstem          |||
>                   edr  <<<          initstem ~~- base |||
>                   edlr <<< base -~~ initstem ~~- base |||
>                   drem <<<          initstem          ... h_l

>   initstem      = is <<< closed

>   closed        = tabulated (
>                   stack ||| hairpin ||| leftB ||| rightB |||
>                   iloop ||| multiloop ... h)

>   multiloop     = (mldl  <<< base -~~ base ~~- base ~~!! ml_components
>                           ~~- base ~~- base                            |||
```

```
>                   mldr  <<< base -~~ base ~~!          ml_components
>                         ~~- base ~~- base ~~- base                   |||
>                   mldlr <<< base -~~ base ~~- base ~~!! ml_components
>                         ~~- base ~~- base ~~- base                   |||
>                   ml    <<< base -~~ base ~~!          ml_components
>                         ~~- base ~~- base )
>                   `with` stackpairing ... h

>   ml_components = combine <<< block ~~~ comps ... h

>   comps         = tabulated (
>                   cons  <<< block ~~~ comps  |||
>                             block            |||
>                   addss <<< block ~~~ region ... h)

>   block         = tabulated (
>                   ul    <<< edangle           |||
>                   ssadd <<< region ~~~ edangle ... h)

>   stack         = (sr  <<< base -~~ closed ~~- base) `with` basepairing

>   hairpin       = (hl  <<< base -~~ base ~~! (region `with` minloopsize 3)
>                         ~~- base ~~- base)
>                   `with` stackpairing

>   leftB         = (sp  <<< base -~~ base ~~! (bl <<< region  ~~~ initstem)
>                         ~~- base ~~- base)
>                   `with` stackpairing ... h

>   rightB        = (sp  <<< base -~~ base ~~! (br <<< initstem ~~~ region)
>                         ~~- base ~~- base)
>                   `with` stackpairing ... h

>   iloop         = (sp  <<< base -~~ base ~~!
>                           (il <<< (region `with` (maxsize 30))
>                                       ~~~ closed ~~~
>                                   (region `with` (maxsize 30)))
>                         ~~- base ~~- base)
>                   `with` stackpairing ... h
```

The `canonicals` grammar is unambiguous in the sense that a dangling base is treated as a structural element. This means that a stem with a dangling base is a different structure than the same stem with the base not dangling. Furthermore, one base in between two stems can either dangle to the left or the right stem (or it does not dangle at all), which means that these are again different structures. A major problem is that each stem with an adjacent base on each side can be interpreted in four ways: no dangling bases, dangle left and no dangle right, no dangle left and dangle right, dangle

left and dangle right. It is debatable if this treatment of dangling bases is correct, since traditionally base pairs are the fundamental building blocks of RNA secondary structure.

For this reason, I developed a grammar which handles dangling bases in a unique way. To achieve this the following general rules were transfered to the above grammar: (I) A singlestranded region has to be followed by a structural element with a dangling base on the left; (II) A structural element with a dangling base on the right is followed by a singlestrand or a structural element with a dangling base on the left; (III) A structural element with no dangling base on the right has to be followed by an element with no dangling base on the left; (IV) One unpaired base in between two structural elements or in between the closing base pair of a multiloop and a structural element (inside the multiloop) is handled explicitly, to be able to decide (in the evaluation algebra) to which stem the base should dangle. In the following transformed grammar each production a rule applies to is annotated with the corresponding number (I - IV):

```
> canonicals_unique_dangles takes alg inp_tr = axiom struct where
>
>  (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>   ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,
>   ssadd,cons,ul,combine,acomb,h,h_i,h_l,h_s) = alg baseArray takes

>  struct        = helixtype1   |||
>                  helixtype2   ... h

>  helixtype1    = listed(
>      (IV)        ambd <<< edanglel  ~~- base ~~~ noleft_dangle |||
>      (IV)        ambd <<< nodangle  ~~- base ~~~ noleft_dangle |||
>      (III)       cadd <<< edanglel  ~~~
>                           (noleft_dangle ||| (nil <<< empty))   |||
>      (III)       cadd <<< nodangle  ~~~
>                           (noleft_dangle ||| (nil <<< empty))   |||
>      (II)        cadd <<< edangler  ~~~
>                           (left_dangle ||| helixtype2)          |||
>      (II)        cadd <<< edanglelr ~~~
>                           (left_dangle ||| helixtype2)          |||
>                  nil  <<< empty                                ... h)

>  helixtype2    =
>      (I)         sadd <<< base    -~~ helixtype2     |||
>      (I)         sadd <<< base    -~~ left_dangle    ... h

>  left_dangle   = listed (
>      (IV)        ambd <<< edanglel  ~~- base ~~~ noleft_dangle |||
>      (III)       cadd <<< edanglel  ~~~
>                           (noleft_dangle ||| (nil <<< empty))   |||
>      (III)       cadd <<< edanglelr ~~~
>                           (left_dangle ||| helixtype2)          |||
```

```
>                 nil  <<< empty                                  ... h)

>  noleft_dangle = listed (
>      (I)         cadd <<< edangler ~~~
>                          (left_dangle  ||| helixtype2)         |||
>      (III)       cadd <<< nodangle ~~~
>                          (noleft_dangle ||| (nil <<< empty))   |||
>      (IV)        ambd <<< nodangle  ~~- base ~~~ noleft_dangle ... h)

>  edanglel      = edl  <<< base -~~ initstem            ... h_l
>  edangler      = edr  <<<          initstem ~~- base   ... h_l
>  edanglelr     = edlr <<< base -~~ initstem ~~- base   ... h_l
>  nodangle      = drem <<<          initstem            ... h_l

>  initstem      = is <<< closed

>  closed        = tabulated (
>                  stack ||| hairpin ||| multiloop ||| leftB |||
>                  rightB ||| iloop ... h)

>  multiloop     = (mldl   <<< base -~~ base ~~- base ~~!! ml_comps1
>                                   ~~- base ~~- base                |||
>      (IV)         mladl  <<< base -~~ base ~~- base ~~!! ml_comps2
>                                   ~~- base ~~- base                |||
>                   mldr   <<< base -~~ base ~~!           ml_comps3
>                          ~~- base ~~- base ~~- base                |||
>      (IV)         mladr  <<< base -~~ base ~~!           ml_comps2
>                          ~~- base ~~- base ~~- base                |||
>                   mldlr  <<< base -~~ base ~~- base ~~!! ml_comps4
>                          ~~- base ~~- base ~~- base                |||
>      (IV)         mladlr <<< base -~~ base ~~- base ~~!! ml_comps2
>                          ~~- base ~~- base ~~- base                |||
>      (IV)         mldladr<<< base -~~ base ~~- base ~~!! ml_comps1
>                          ~~- base ~~- base ~~- base                |||
>      (IV)         mladldr<<< base -~~ base ~~- base ~~!! ml_comps3
>                          ~~- base ~~- base ~~- base                |||
>                   ml     <<< base -~~ base ~~!           ml_comps2
>                          ~~- base ~~- base                           )
>                  `with` stackpairing  ... h

>  ml_comps1 = tabulated (
>      (III)   combine <<< block_dl  ~~~ no_dl_no_ss_end          |||
>      (II)    combine <<< block_dlr ~~~ dl_or_ss_left_no_ss_end  |||
>      (IV)    acomb   <<< block_dl  ~~- base ~~~ no_dl_no_ss_end ... h_i)
```

```
>  ml_comps2 = tabulated (
>      (III)   combine <<< (ul <<< nodangle) ~~~ no_dl_no_ss_end          |||
>      (II)    combine <<< (ul <<< edangler) ~~~ dl_or_ss_left_no_ss_end  |||
>      (IV)    acomb   <<< (ul <<< nodangle) ~~- base ~~~ no_dl_no_ss_end
>                                                                  ... h_i)

>  ml_comps3 =
>      (II)    combine <<< (ul <<< edangler) ~~~ dl_or_ss_left_ss_end  |||
>      (III)   combine <<< (ul <<< nodangle) ~~~ no_dl_ss_end          |||
>      (IV)    acomb   <<< (ul <<< nodangle) ~~- base ~~~ no_dl_ss_end
>                                                            ... h_i

>  ml_comps4 =
>      (III)   combine <<< block_dl  ~~~  no_dl_ss_end        |||
>      (II)    combine <<< block_dlr ~~~ dl_or_ss_left_ss_end |||
>      (IV)    acomb   <<< block_dl ~~- base ~~~ no_dl_ss_end ... h_i

>  block_dl  =
>      (I)     ssadd <<< region ~~~ edanglel |||
>              ul    <<< edanglel            ... h_i

>  block_dlr =
>      (I)     ssadd <<< region ~~~ edanglelr |||
>              ul    <<< edanglelr            ... h_i

>  no_dl_no_ss_end =
>                   ul <<< nodangle  |||
>                   ml_comps2

>  dl_or_ss_left_no_ss_end =
>                           ml_comps1 |||
>                           block_dl

>  no_dl_ss_end = tabulated (
>                 ul    <<< edangler                    |||
>                 ml_comps3                             |||
>      (II)       addss <<< (ul <<< edangler) ~~~ region ... h_i)

>  dl_or_ss_left_ss_end = tabulated (
>      (II)               addss <<< block_dlr ~~~ region |||
>                         ml_comps4                     |||
>                         block_dlr                     ... h_i)

>  stack         = (sr  <<< base -~~ closed ~~- base) `with` basepairing
```

```
>  hairpin       = (hl  <<< base -~~ base ~~! (region `with` minloopsize 3)
>                       ~~- base ~~- base)
>                  `with` stackpairing

>  leftB         = (sp  <<< base -~~ base ~~! (bl <<< region  ~~~ initstem)
>                       ~~- base ~~- base)
>                  `with` stackpairing

>  rightB        = (sp  <<< base -~~ base ~~! (br <<< initstem ~~~ region)
>                       ~~- base ~~- base)
>                  `with` stackpairing

>  iloop         = (sp  <<< base -~~ base ~~!
>                           (il <<< (region `with` (maxsize 30))
>                                   ~~~ closed ~~~
>                                   (region `with` (maxsize 30)))
>                       ~~- base ~~- base)
>                  `with` stackpairing
```

This grammar is especially needed for the calculation of the partition function. The reason for this is, that, using the `canonicals` grammar, structures with several stems get overrepresented, due to the possibility of multiple combinations of dangling bases. For four stems this means, that the same structure (in "Vienna" notation) can be derived in $4^4 = 64$ ways, whereas a single stem can only be derived in $4^1 = 4$ ways. This results in a higher value for the partition function which lowers the probability of each individual structure. As the probability of a shape is the sum of the probabilities of its structures, this would lead to incorrect higher probabilities for shapes with more stems. In terms of a landscape this means that each valley in the landscape gets broadened by a factor of $(nr.\ of\ stems)^4$.

B.2 Evaluation Algebras

The grammars from above are complemented by algebras which evaluate the candidates derived by the grammar. The choice function `h_l` has a special function as it filters out elements having non-negative energy or a Boltzmann weighted energy smaller 1.0. It is used in the above grammars in the productions for `edangle` (`canonicals` grammar) and `edanglel`, `edangler`, `edanglelr`, `nodangle` (`canonicals_unique_dangles` grammar). The idea behind is that a substructure in the external or a multiple loop should not have positive energy, as in this case it would not form. In the following I will give algebras for structure enumeration, structure counting, minimum free energy calculation, pretty printing in "Vienna" notation, pretty printing in shapes notation, shape analysis, partition function calculation and the product algebra to combine algebras. They are all given with the additional functions for the `canonicals_unique_dangles` grammar but also apply to the `canonicals` grammar.

B.2.1 Structure Enumeration

```
> enum :: Array Int Ebase -> a ->
>         Canonical_Algebra Int (Int,Int) Closed Closed
> enum seq _ = (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>               ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,
>               ssadd,cons,ul,combine,acomb,h,h_i,h_l,h_s)
>               where
>  sadd b = Sadd (s b)
>  cadd = Cadd
>  ambd c b a = Ambd c (s b) a
>  nil _ = Nil
>  edl b = Edl (s b)
>  edr c b = Edr c (s b)
>  edlr lb c rb = Edlr (s lb) c (s rb)
>  drem = Drem
>  is = Is
>  sr lb c rb = Sr (s lb) c (s rb)
>  hl  llb lb l rb rrb = Hl  (s llb) (s lb) l (s rb) (s rrb)
>  sp  llb lb c rb rrb = Sp (s llb) (s lb) c (s rb) (s rrb)
>  bl  c bulge        = Bl c bulge
>  br  bulge c        = Br bulge c
>  il  reg1 c reg2    = Il reg1 c reg2
>  ml llb lb multi rb rrb = Ml (s llb) (s lb) multi (s rb) (s rrb)
>  mldr    llb lb    multi dr rb rrb
>                = Mldr   (s llb)(s lb)       multi (s dr)(s rb)(s rrb)
>  mladr   llb lb    multi dr rb rrb
>                = Mladr  (s llb)(s lb)       multi (s dr)(s rb)(s rrb)
>  mldlr   llb lb dl multi dr rb rrb
>                = Mldlr  (s llb)(s lb)(s dl) multi (s dr)(s rb)(s rrb)
```

```
>  mladlr  llb lb dl multi dr rb rrb
>                = Mladlr (s llb)(s lb)(s dl) multi (s dr)(s rb)(s rrb)
>  mldladr llb lb dl multi dr rb rrb
>                = Mladlr (s llb)(s lb)(s dl) multi (s dr)(s rb)(s rrb)
>  mladldr llb lb dl multi dr rb rrb
>                = Mladlr (s llb)(s lb)(s dl) multi (s dr)(s rb)(s rrb)
>  mldl    llb lb dl multi    rb rrb
>                = Mldl   (s llb)(s lb)(s dl) multi       (s rb)(s rrb)
>  mladl   llb lb dl multi    rb rrb
>                = Mladl  (s llb)(s lb)(s dl) multi       (s rb)(s rrb)
>  addss = Addss
>  ssadd = Ssadd
>  cons = Cons
>  ul = Ul
>  combine = Combine
>  acomb c1 b c2 = Acomb c1 (s b) c2
>  h = id
>  h_i = id
>  h_l = id
>  h_s = id

>  s i = seq!i
```

B.2.2 Pretty Printing - "Vienna" Notation

The `prettyprint`-algebra takes as arguments (`left, right`) the symbols representing dangling bases on the 5'- and 3'-side, respectively. In the "Vienna" notation there is no explicit representation of dangling bases, so that the "Vienna" notation is achieved with `prettyprint '.' '.'`.

```
> prettyprint :: Char -> Char -> a -> b ->
>                Canonical_Algebra i (Int,Int) String String
> prettyprint left right _ _ =
>         (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>          ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,
>          addss,ssadd,cons,ul,combine,acomb,h,h_i,h_l,h_s) where
>  sadd _  s  = '.':s
>  cadd s1 s2 = s1++s2
>  ambd s1 b s2 = s1++('.':s2)
>  nil  _     = []
>  edl  _  s   = left:s
>  edr  s  _   = s++right:[]
>  edlr _  s _ = left:s++right:[]
>  drem        = id
>  is          = id
>  sr   _  s _ = '(':s++")"
```

```
>  hl   _  _ (h1,h2) _ _   = '(':'(':dots (h2-h1)++"))"
>  sp   _  _ s _ _ = '(':'(':s++"))"
>  bl   (l1,l2) s  = dots (l2-l1)++s
>  br   s (r1,r2)  = s++dots (r2-r1)
>  il  (l1,l2) s (r1,r2) = dots (l2-l1)++s++dots (r2-r1)
>  ml    _ _ s _ _     = '(':'(':s++"))"
>  mldr  _ _ s _ _ _   = '(':'(':s++left:"))"
>  mladr  _ _ s _ _ _   = '(':'(':s++left:"))"
>  mldlr _ _ _ s _ _ _ = '(':'(':right:s++left:"))"
>  mladlr _ _ _ s _ _ _ = '(':'(':right:s++left:"))"
>  mldladr _ _ _ s _ _ _ = '(':'(':right:s++left:"))"
>  mladldr _ _ _ s _ _ _ = '(':'(':right:s++left:"))"
>  mldl  _ _ _ s _ _   = '(':'(':right:s++"))"
>  mladl  _ _ _ s _ _   = '(':'(':right:s++"))"
>  addss s (r1,r2) = s++dots (r2-r1)
>  ssadd (l1,l2) s = dots (l2-l1)++s
>  cons  s1  s2    = s1++s2
>  ul  s     = s
>  combine s1 s2 = s1 ++ s2
>  acomb s1 b s2 = s1 ++ '.' : s2
>  h   = id
>  h_i = id
>  h_l = id
>  h_s = id
>  dots i = replicate i '.'

``Vienna'' notation algebra:

> vienna = prettyprint '.' '.'
```

B.2.3 Minimum Free Energy Calculation

To be able to assign the correct dangling energies, some functions, such as `ambd` (ambiguous dangle), need to know the indices of the closed substructure at the 5'-end of the subword, rather than the indices of the subword itself. For this reason, all functions that append singlestrands and the function `cadd` communicate the former in their results. Furthermore, the corresponding functions for multiloops need to know the indices of the 5'-end and the 3'-end closed substructure to decide whether a base dangles to the closing stem of the multiloop or the stem at the 5'- and 3'-end, respectively.

```
> mfe :: Array Int Ebase -> Float ->
>        Canonical_Algebra Int (Int,Int) (Float,Int,Int)
>                              (Float,(Int,Int),(Int,Int))
```

```
> mfe array takes = (sadd,cadd,ambd array,nil,edl array,edr array,edlr array,
>                    drem,is array,sr array,hl array,sp array,bl array,
>                    br array,il array,ml array,mldr array,mladr array,
>                    mldlr array,mladlr array,mldladr array,mladldr array,
>                    mldl array,mladl array,addss,ssadd,cons,ul,combine,
>                    acomb array,h,h_i,h_l,h_s) where
>  sadd lb (e,_,rb) = (e,lb,rb)
>  cadd (e1,lb1,rb1) (e2,lb2,rb2)     = (e1 + e2,lb1,rb1)
>  ambd inp (e1,lb1,rb1) db (e2,lb2,rb2)
>   =(e1+e2+(min (dr_energy inp (lb1,rb1))(dl_energy inp (lb2,rb2))),lb1,rb1)
>  nil _ = (0,n,n)
>  edl  inp dl (e,lb,rb)
>   = (e + dl_energy inp (lb,rb),lb,rb)
>  edr  inp    (e,lb,rb) dr
>   = (e + dr_energy inp (lb,rb),lb,rb)
>  edlr inp dl (e,lb,rb) dr
>   = (e + dl_energy inp (lb,rb) + dr_energy inp (lb,rb),lb,rb)
>  drem = id
>  is inp    (e,lb,rb)    = (e + termaupenalty (inp!lb) (inp!rb),lb,rb)
>  sr inp lb (e,_,_) rb = (e + sr_energy inp (lb,rb),lb,rb)
>  hl inp llb lb loop rb rrb
>   = (hl_energy inp (lb,rb) + sr_energy inp (llb,rrb),llb,rrb)
>  sp inp llb lb (e,_,_) rb rrb = (e + sr_energy inp (llb,rrb), llb,rrb)
>  bl inp (l,r) (e,lend,rend)
>   = (e + bl_energy inp l    (l,r) (rend+1),l,rend)
>  br inp (e,lend,rend) (l,r)
>   = (e + br_energy inp (lend-1) (l,r) (r+1),lend,r)
>  il inp (l1,l2) (e,l,r) (r1,r2)
>   = (e + il_energy inp (l1,l2) (r1,r2), l1, r2)
>  ml inp llb lb (e,_,_) rb rrb
>   = (380+e+sr_energy inp (llb,rrb)+termaupenalty (inp!lb) (inp!rb),llb,rrb)
>  mldr inp llb lb (e,_,_) dr rb rrb
>   = (380 + e + dri_energy inp (lb,rb)  + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>  mladr inp llb lb (e,_,(k,l)) dr rb rrb
>   = (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>     where dangle_e = min (dri_energy inp (lb,rb)) (dr_energy inp (k,l))
>  mldlr inp llb lb dl (e,_,_) dr rb rrb
>   = (380 + e + dli_energy inp (lb,rb)  + dri_energy inp (lb,rb)
>      + sr_energy inp (llb,rrb) + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>  mladlr inp llb lb dl (e,(i,j),(k,l)) dr rb rrb
>   = (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>      where dangle_e = (min (dli_energy inp (lb,rb)) (dl_energy inp (i,j)))
>                       + (min (dri_energy inp (lb,rb)) (dr_energy inp (k,l)))
>  mldladr inp llb lb dl (e,(i,j),(k,l)) dr rb rrb
```

```
>   = (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>      where dangle_e = dli_energy inp (lb,rb)
>                     + min (dri_energy inp (lb,rb)) (dr_energy inp (k,l))
>  mladldr inp llb lb dl (e,(i,j),(k,l)) dr rb rrb
>   = (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb),llb,rrb)
>      where dangle_e = (min (dli_energy inp (lb,rb)) (dl_energy inp (i,j)))
>                     + dri_energy inp (lb,rb)
>  mldl inp llb lb dl (e,_,_) rb rrb
>   = (380 + e + dli_energy inp (lb,rb)  + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb), llb,rrb)
>  mladl inp llb lb dl (e,(i,j),_) rb rrb
>   = (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb), llb,rrb)
>      where dangle_e = min (dli_energy inp (lb,rb)) (dl_energy inp (i,j))
>  addss (e,(lb1,rb1),(lb2,rb2)) (i,j)
>   = (e + ss_energy (i,j),(lb1,rb1),(lb2,rb2))
>  ssadd (i,j) (e,lb,rb) = (40 + e + ss_energy (i,j),(lb,rb),(lb,rb))
>  cons (e1,lb1,rb1) (e2,lb2,rb2) = (e1 + e2,lb1,rb1)
>  ul (e,lb,rb) = (40 + e,(lb,rb),(lb,rb))
>  combine (e1,(lb1,rb1),_) (e2,_,(lb2,rb2)) = (e1 + e2,(lb1,rb1),(lb2,rb2))
>  acomb inp (e1,(lba,rba),(lb1,rb1)) b (e2,(lb2,rb2),(lbb,rbb))
>   = (e1 + e2
>   + (min (dr_energy inp (lb1,rb1)) (dl_energy inp (lb2,rb2))),
>     (lba,rba),(lbb,rbb))

>  h [] = []
>  h xs = [minimum xs]
>  h_i = h
>  h_l [] = []
>  h_l xs = if (minE_xs < 0.0) then [(minE_xs,i,j)] else []
>    where (minE_xs,i,j) = minimum xs
>  h_s = h

>  (_,n) = bounds array
```

B.2.4 Shapes Notation

The general `shape` algebra takes 6 arguments: `edangle_op`, `edangle_cl`, `loop_ss`, `loop_op`, `loop_cl` and `ss`. `edangle_op` and `edangle_cl` give the opening and closing character in the case that only the nesting of hairpins and multiloops is considered. Accordingly, `loop_op` and `loop_cl` give the characters when all loop types should get represented. `loop_ss` and `ss` represent the singlestranded regions of loops and of the external and multi loop, respectively. In the case of abstracting from certain elements, e.g. all singlestrands, the corresponding arguments are assigned the empty string.

```
> shape :: String -> String -> String -> String -> String -> String ->
>        Array Int Ebase -> Float ->
>        Canonical_Algebra Int (Int,Int) String String

> shape edangle_op edangle_cl loop_ss loop_op loop_cl ss array takes =
>       (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>        ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,
>        ssadd,cons,ul,combine,acomb,h,h_i,h_l,h_s) where
>  sadd _ s     = if (ss == "" && s == "") then "_" else app ss s
>  cadd s1 s2   = if (ss == "" && s2 =="_") then s1 else app s1 s2
>  ambd s1 b s2 = app (app s1 ss) s2
>  nil  _       = ""
>  edl  _  s    = ss++edangle_op++s++edangle_cl
>  edr  s _     = edangle_op++s++edangle_cl++ss
>  edlr _ s _   = ss++edangle_op++s++edangle_cl++ss
>  drem    s    = edangle_op++s++edangle_cl
>  is           = id
>  sr _ s _     = s
>  hl _ _ _ _ _ = loop_op++loop_ss++loop_cl
>  sp _ _ s _ _ = s
>  bl _ s       = loop_op++loop_ss++s++loop_cl
>  br s _       = loop_op++s++loop_ss++loop_cl
>  il _ s _     = loop_op++loop_ss++s++loop_ss++loop_cl
>  ml _ _ s _ _ = loop_op++s++loop_cl
>  mldr _ _ s _ _ _  = loop_op++ (app s ss) ++loop_cl
>  mladr _ _ s _ _ _ = loop_op++ (app s ss) ++loop_cl
>  mldlr _ _ _ s _ _ _  = loop_op++ (app ss (app s ss)) ++loop_cl
>  mladlr _ _ _ s _ _ _ = loop_op++ (app ss (app s ss)) ++loop_cl
>  mldladr _ _ _ s _ _ _ = loop_op++ (app ss (app s ss)) ++loop_cl
>  mladldr _ _ _ s _ _ _ = loop_op++ (app ss (app s ss)) ++loop_cl
>  mldl _ _ _ s _ _  = loop_op++ (app ss s) ++loop_cl
>  mladl _ _ _ s _ _ = loop_op++ (app ss s) ++loop_cl
>  addss s _    = app s ss
>  ssadd _ s    = app ss s
>  cons s1 s2   = app s1 s2
>  ul s         = s
>  combine s1 s2= app s1 s2
>  acomb s1 b s2= app (app s1 ss) s2
>  h   = nub
>  h_i = h
>  h_l = h
>  h_s = h

app fuses adjacent '_'s to one '_' when concatenating strings

> app :: String -> String -> String
```

```
> app [] ys = ys
> app "_" "_" = "_"
> app (x:[]) (y:[]) = x:y:[]
> app (x:[]) (y:ys) = app (app (x:[]) (y:[])) ys
> app (x:xs) ys = x : app xs ys
```

The different abstraction levels are defined by different assignments for the arguments of the `shape` algebra. The definitions for the five shape types I implemented so far are as follows:

```
> shape1 = shape "" "" "_" "[" "]" "_"
> shape2 = shape "" "" "" "[" "]" "_"
> shape3 = shape "" "" "" "[" "]" ""
> shape4 = shape "[" "]" "" "" "" "_"
> shape5 = shape "[" "]" "" "" "" ""
```

B.2.5 Shape Analysis

As stated in the outline of the implementation of `RNAshapes` (see Section 5.4.1), the algebra makes use of a combination of the `minimum free energy`, `shape` and `Vienna notation` algebras. To achieve this the `shapes` algebra obtains the functions from those algebras, except the choice function which is implemented explicitly to facilitate the filtering of equal shapes. The shape algebra is given as an argument to enable the use of the different abstraction levels.

```
> shapes :: (Array Int Ebase -> Float ->
>            Canonical_Algebra Int (Int,Int) String String) ->
>            Array Int Ebase -> Float ->
>            Canonical_Algebra Int (Int,Int) ((Float,Int,Int),String,String)
>                                  ((Float,(Int,Int),(Int,Int)),String,String)
> shapes alg1 array takes =
>         (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>          ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,ssadd,
>          cons,ul,combine,acomb,h,h_i,h_l,h_s) where
>  (sadd1,cadd1,ambd1,nil1,edl1,edr1,edlr1,drem1,is1,sr1,hl1,sp1,bl1,br1,il1,
>   ml1,mldr1,mladr1,mldlr1,mladlr1,mldladr1,mladldr1,mldl1,mladl1,addss1,
>   ssadd1,cons1,ul1,combine1,acomb1,h1,h_i1,h_l1,h_s1) = mfe array takes
>  (sadd2,cadd2,ambd2,nil2,edl2,edr2,edlr2,drem2,is2,sr2,hl2,sp2,bl2,br2,il2,
>   ml2,mldr2,mladr2,mldlr2,mladlr2,mldladr2,mladldr2,mldl2,mladl2,addss2,
>   ssadd2,cons2,ul2,combine2,acomb2,h2,h_i2,h_l2,h_s2) = alg1 array takes
>  (sadd3,cadd3,ambd3,nil3,edl3,edr3,edlr3,drem3,is3,sr3,hl3,sp3,bl3,br3,il3,
>   ml3,mldr3,mladr3,mldlr3,mladlr3,mldladr3,mladldr3,mldl3,mladl3,addss3,
>   ssadd3,cons3,ul3,combine3,acomb3,h3,h_i3,h_l3,h_s3) = vienna array takes

>  sadd b (a1,a2,a3) = (sadd1 b a1, sadd2 b a2, sadd3 b a3)
>  cadd (c1,c2,c3) (a1,a2,a3) = (cadd1 c1 a1, cadd2 c2 a2, cadd3 c3 a3)
>  ambd (c1,c2,c3) b (a1,a2,a3) = (ambd1 c1 b a1,
```

```
>                                  ambd2 c2 b a2,
>                                  ambd3 c3 b a3)
>  nil a = (nil1 a, nil2 a, nil3 a)
>  edl b (c1,c2,c3) = (edl1 b c1, edl2 b c2, edl3 b c3)
>  edr (c1,c2,c3) b = (edr1 c1 b, edr2 c2 b, edr3 c3 b)
>  edlr b (c1,c2,c3) b' = (edlr1 b c1 b', edlr2 b c2 b', edlr3 b c3 b')
>  drem (c1,c2,c3) = (drem1 c1, drem2 c2, drem3 c3)
>  is (c1,c2,c3)   = (is1 c1, is2 c2, is3 c3)
>  sr b (c1,c2,c3) b' = (sr1 b c1 b', sr2 b c2 b', sr3 b c3 b')
>  hl b1 b2 u b2' b1' = (hl1 b1 b2 u b2' b1',
>                        hl2 b1 b2 u b2' b1',
>                        hl3 b1 b2 u b2' b1')
>  sp b1 b2 (c1,c2,c3) b2' b1' = (sp1 b1 b2 c1 b2' b1',
>                                 sp2 b1 b2 c2 b2' b1',
>                                 sp3 b1 b2 c3 b2' b1')
>  bl u (c1,c2,c3) = (bl1 u c1, bl2 u c2, bl3 u c3)
>  br (c1,c2,c3) u = (br1 c1 u, br2 c2 u, br3 c3 u)
>  il r1 (c1,c2,c3) r2 = (il1 r1 c1 r2 ,il2 r1 c2 r2, il3 r1 c3 r2)
>  ml b1 b2 (m1,m2,m3) b2' b1' = (ml1 b1 b2 m1 b2' b1',
>                                 ml2 b1 b2 m2 b2' b1',
>                                 ml3 b1 b2 m3 b2' b1')
>  mldr b1 b2 (m1,m2,m3) d b2' b1' = (mldr1 b1 b2 m1 d b2' b1',
>                                     mldr2 b1 b2 m2 d b2' b1',
>                                     mldr3 b1 b2 m3 d b2' b1')
>  mladr b1 b2 (m1,m2,m3) d b2' b1' = (mladr1 b1 b2 m1 d b2' b1',
>                                      mladr2 b1 b2 m2 d b2' b1',
>                                      mladr3 b1 b2 m3 d b2' b1')
>  mldlr b1 b2 d (m1,m2,m3) d' b2' b1' = (mldlr1 b1 b2 d m1 d' b2' b1',
>                                         mldlr2 b1 b2 d m2 d' b2' b1',
>                                         mldlr3 b1 b2 d m3 d' b2' b1')
>  mladlr b1 b2 d (m1,m2,m3) d' b2' b1' = (mladlr1 b1 b2 d m1 d' b2' b1',
>                                          mladlr2 b1 b2 d m2 d' b2' b1',
>                                          mladlr3 b1 b2 d m3 d' b2' b1')
>  mldladr b1 b2 d (m1,m2,m3) d' b2' b1' = (mldladr1 b1 b2 d m1 d' b2' b1',
>                                           mldladr2 b1 b2 d m2 d' b2' b1',
>                                           mldladr3 b1 b2 d m3 d' b2' b1')
>  mladldr b1 b2 d (m1,m2,m3) d' b2' b1' = (mladldr1 b1 b2 d m1 d' b2' b1',
>                                           mladldr2 b1 b2 d m2 d' b2' b1',
>                                           mladldr3 b1 b2 d m3 d' b2' b1')
>  mldl b1 b2 d (m1,m2,m3) b2' b1' = (mldl1 b1 b2 d m1 b2' b1',
>                                     mldl2 b1 b2 d m2 b2' b1',
>                                     mldl3 b1 b2 d m3 b2' b1')
>  mladl b1 b2 d (m1,m2,m3) b2' b1' = (mladl1 b1 b2 d m1 b2' b1',
>                                      mladl2 b1 b2 d m2 b2' b1',
>                                      mladl3 b1 b2 d m3 b2' b1')
>  addss (c1,c2,c3) u = (addss1 c1 u, addss2 c2 u, addss3 c3 u)
>  ssadd u (c1,c2,c3) = (ssadd1 u c1, ssadd2 u c2, ssadd3 u c3)
```

```
>  cons (c1,c2,c3) (c_1,c_2,c_3) = (cons1 c1 c_1, cons2 c2 c_2, cons3 c3 c_3)
>  ul (c1,c2,c3) = (ul1 c1, ul2 c2, ul3 c3)
>  combine (c1,c2,c3) (c_1,c_2,c_3) = (combine1 c1 c_1,
>                                      combine2 c2 c_2,
>                                      combine3 c3 c_3)
>  acomb (c1,c2,c3) b (c_1,c_2,c_3) = (acomb1 c1 b c_1,
>                                      acomb2 c2 b c_2,
>                                      acomb3 c3 b c_3)

>  h [] = []
>  h xs = extract_shapes [] res_list
>    where
>      res_list = filter_erange (lowest+takes) xs
>      filter_erange _ [] = []
>      filter_erange limit (((x1,a,b),x2,x3):xs)
>            | x1 <= limit = ((x1,a,b),x2,x3):(filter_erange limit xs)
>            | otherwise   = filter_erange limit xs
>      extract_shapes shs []    = shs
>      extract_shapes shs (x:xs)= extract_shapes (add_shape x shs) xs
>      add_shape x [] = [x]
>      add_shape ((x1,a,b),x2,x3) (((s1,c,d),s2,s3):shs)
>        | x2 == s2 && x1 < s1  = ((x1,a,b),x2,x3):shs
>        | x2 == s2 && x1 >= s1 = ((s1,c,d),s2,s3):shs
>        | otherwise = ((s1,c,d),s2,s3): add_shape ((x1,a,b),x2,x3) shs
>      ((lowest,_,_),_,_) = minimum xs

>  h_l [] = []
>  h_l xs = extract_shapes [] res_list
>    where
>      res_list = filter_erange (lowest+takes) xs
>      filter_erange _ [] = []
>      filter_erange limit (((x1,a,b),x2,x3):xs)
>            | x1 < 0.0 && x1 <= limit
>               = ((x1,a,b),x2,x3):(filter_erange limit xs)
>            | otherwise   = filter_erange limit xs
>      extract_shapes shs []    = shs
>      extract_shapes shs (x:xs)= extract_shapes (add_shape x shs) xs
>      add_shape x [] = [x]
>      add_shape ((x1,a,b),x2,x3) (((s1,c,d),s2,s3):shs)
>        | x2 == s2 && x1 < s1  = ((x1,a,b),x2,x3):shs
>        | x2 == s2 && x1 >= s1 = ((s1,c,d),s2,s3):shs
>        | otherwise = ((s1,c,d),s2,s3): add_shape ((x1,a,b),x2,x3) shs
>      ((lowest,_,_),_,_) = minimum xs

>  h_i = h
>  h_s = h
```

```
>  (_,n) = bounds array

> shapes1 = shapes shape1
> shapes2 = shapes shape2
> shapes3 = shapes shape3
> shapes4 = shapes shape4
> shapes5 = shapes shape5
```

B.2.6 Structure Counting

```
> count :: a -> b -> Canonical_Algebra i (Int,Int) Integer Integer
> count _ _ = (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>              ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,
>              ssadd,cons,ul,combine,acomb,h,h_i,h_l,h_s) where
>  sadd _ b = b
>  cadd a b = a*b
>  ambd a _ b = a*b
>  nil _ = 1
>  edl _ b = b
>  edr a _ = a
>  edlr _ b _ = b
>  drem a = a
>  is a = a
>  sr _ b _ = b
>  hl _ _ _ _ _ = 1
>  sp _ _ c _ _ = c
>  bl _ c = c
>  br c _ = c
>  il _ c _ = c
>  ml _ _ c _ _ = c
>  mldr _ _ c _ _ _ = c
>  mladr _ _ c _ _ _ = c
>  mldlr _ _ _ c _ _ _ = c
>  mladlr _ _ _ c _ _ _ = c
>  mldladr _ _ _ c _ _ _ = c
>  mladldr _ _ _ c _ _ _ = c
>  mldl _ _ _ c _ _ = c
>  mladl _ _ _ c _ _ = c
>  addss a _ = a
>  ssadd _ b = b
>  cons a b = a * b
>  ul a  = a
>  combine a b = a*b
>  acomb a _ b = a*b
>  h [] = []
>  h xs = [sum xs]
```

```
>  h_i= h
>  h_l= h
>  h_s= h
```

B.2.7 Partition Function Calculation

```
> r_gas = 0.00198717 -- [kcal/mol] <-- 1.98717 [cal/mol]
> temperature  = 310.15  -- [K]
> mk_pf x = exp ((-x/100) / (r_gas * temperature))

> p_func :: Array Int Ebase -> Float ->      -- closed      answer
>           Canonical_Algebra Int (Int,Int) (Float,Int,Int)
>                                 (Float,(Int,Int),(Int,Int))
> p_func array takes = (sadd,cadd,ambd array,nil,edl array,edr array,
>                       edlr array,drem,is array,sr array,hl array,sp array,
>                       bl array,br array,il array,ml array,mldr array,
>                       mladr array,mldlr array,mladlr array,mldladr array,
>                       mladldr array,mldl array,mladl array,addss,ssadd,
>                       cons,ul,combine,acomb array,h,h_i,h_l,h_s) where
>  sadd b (q,lb,rb) = (q,lb,rb)
>  cadd (q1,lb1,rb1) (q2,lb2,rb2)   = (q1 * q2, lb1,rb1)
>  ambd inp (q1,lb1,rb1) b (q2,lb2,rb2)
>   =(q1 * q2
>    * mk_pf(min (dr_energy inp (lb1,rb1))(dl_energy inp (lb2,rb2))),lb1,rb1)
>  nil _ = (1.0,n,n)
>  edl  inp dl (q,lb,rb)    = (q * mk_pf (dl_energy inp (lb,rb)),lb,rb)
>  edr  inp    (q,lb,rb) dr = (q * mk_pf (dr_energy inp (lb,rb)),lb,rb)
>  edlr inp dl (q,lb,rb) dr
>   = (q * mk_pf (dl_energy inp (lb,rb) + dr_energy inp (lb,rb)),lb,rb)
>  drem = id
>  is inp    (q,lb,rb)  = (q * mk_pf (termaupenalty (inp!lb) (inp!rb)),lb,rb)
>  sr inp lb (q,_,_) rb = (q * mk_pf (sr_energy inp (lb,rb)),lb,rb)
>  hl inp llb lb loop rb rrb
>   = (mk_pf (hl_energy inp (lb,rb) + sr_energy inp (llb,rrb)),llb,rrb)
>  sp inp llb lb (q,_,_) rb rrb
>   = (q * mk_pf (sr_energy inp (llb,rrb)), llb,rrb)
>  bl inp (l,r) (q,lend,rend)
>   = (q * mk_pf (bl_energy inp l    (l,r) (rend+1)),l,rend)
>  br inp (q,lend,rend) (l,r)
>   = (q * mk_pf (br_energy inp (lend-1) (l,r) (r+1)),lend,r)
>  il inp (l1,l2) (q,l,r) (r1,r2)
>   = (q * mk_pf (il_energy inp (l1,l2) (r1,r2)), l1, r2)
>  ml inp llb lb (q,_,_) rb rrb
>   = (q * mk_pf (380 + sr_energy inp (llb,rrb)
<        + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>  mldr inp llb lb (q,_,_) dr rb rrb
```

```
>   = (q * mk_pf (380 + dri_energy inp (lb,rb)  + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>  mladr inp llb lb (q,_,(k,l)) dr rb rrb
>   = (q * mk_pf(380 + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>      where dangle_e = min (dri_energy inp (lb,rb)) (dr_energy inp (k,l))
>  mldlr inp llb lb dl (q,_,_) dr rb rrb
>   = (q * mk_pf (380 + dli_energy inp (lb,rb)  + dri_energy inp (lb,rb)
>      + sr_energy inp (llb,rrb) + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>  mladlr inp llb lb dl (q,(i,j),(k,l)) dr rb rrb
>   = (q * mk_pf (380 + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>      where dangle_e = (min (dli_energy inp (lb,rb)) (dl_energy inp (i,j)))
>                     + (min (dri_energy inp (lb,rb)) (dr_energy inp (k,l)))
>  mldladr inp llb lb dl (q,(i,j),(k,l)) dr rb rrb
>   = (q * mk_pf (380 + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>      where dangle_e = dli_energy inp (lb,rb)
>                     + min (dri_energy inp (lb,rb)) (dr_energy inp (k,l))
>  mladldr inp llb lb dl (q,(i,j),(k,l)) dr rb rrb
>   = (q * mk_pf (380 + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)),llb,rrb)
>      where dangle_e = (min (dli_energy inp (lb,rb)) (dl_energy inp (i,j)))
>                       + dri_energy inp (lb,rb)
>  mldl inp llb lb dl (q,_,_) rb rrb
>   = (q * mk_pf (380 + dli_energy inp (lb,rb) + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)), llb,rrb)
>  mladl inp llb lb dl (q,(i,j),_) rb rrb
>   = (q * (380 + e + dangle_e + sr_energy inp (llb,rrb)
>      + termaupenalty (inp!lb) (inp!rb)), llb,rrb)
>      where dangle_e = min (dli_energy inp (lb,rb)) (dl_energy inp (i,j))
>  addss (q,(lb1,rb1),(lb2,rb2)) (i,j)
<     = (q * mk_pf(ss_energy (i,j)),(lb1,rb1),(lb2,rb2))
>  ssadd (i,j) (q,lb,rb) = (q * mk_pf(40 + ss_energy (i,j)),(lb,rb),(lb,rb))
>  cons (q1,lb1,rb1) (q2,lb2,rb2) = (q1 * q2,lb1,rb1)
>  ul (q,lb,rb) = (q * mk_pf(40),(lb,rb),(lb,rb))
>  combine (q1,(lb1,rb1),_) (q2,_,(lb2,rb2)) = (q1 * q2,(lb1,rb1),(lb2,rb2))
>  acomb inp (q1,(lba,rba),(lb1,rb1)) b (q2,(lb2,rb2),(lbb,rbb))
>   = (q1 * q2
>      * mk_pf (min (dr_energy inp (lb1,rb1)) (dl_energy inp (lb2,rb2))),
>      (lba,rba),(lbb,rbb))
>  h []   = []
>  h xs   = [foldl1 (sum_triples) xs]
>       where sum_triples (x1,_,_) (x2,l,r) = (x1+x2,l,r)
>  h_l [] = []
>  h_l xs = [(y,a,b) | (y,a,b) <- [foldl1 (sum_triples) xs], y >= 1.0]
>       where sum_triples (x1,_,_) (x2,l,r) = (x1+x2,l,r)
```

```
>  h_s    = h
>  h_i    = h
>  (_,n) = bounds array
```

B.2.8 Product Algebra

```
> infix ***
> (alg1 *** alg2) basearray takes =
>         (sadd,cadd,ambd,nil,edl,edr,edlr,drem,is,sr,hl,sp,bl,br,il,
>          ml,mldr,mladr,mldlr,mladlr,mldladr,mladldr,mldl,mladl,addss,ssadd,
>          cons,ul,combine,acomb,h,h_i,h_l,h_s) where
>  (sadd1,cadd1,ambd1,nil1,edl1,edr1,edlr1,drem1,is1,sr1,hl1,sp1,bl1,br1,il1,
>   ml1,mldr1,mladr1,mldlr1,mladlr1,mldladr1,mladldr1,mldl1,mladl1,addss1,
>   ssadd1,cons1,ul1,combine1,acomb1,h1,h_i1,h_l1,h_s1)= alg1 basearray takes
>  (sadd2,cadd2,ambd2,nil2,edl2,edr2,edlr2,drem2,is2,sr2,hl2,sp2,bl2,br2,il2,
>   ml2,mldr2,mladr2,mldlr2,mladlr2,mldladr2,mladldr2,mldl2,mladl2,addss2,
>   ssadd2,cons2,ul2,combine2,acomb2,h2,h_i2,h_l2,h_s2)= alg2 basearray takes

>  sadd b (a1,a2) = (sadd1 b a1, sadd2 b a2)
>  cadd (c1,c2) (a1,a2) = (cadd1 c1 a1, cadd2 c2 a2)
>  ambd (c1,c2) b (a1,a2) = (ambd1 c1 b a1, ambd2 c2 b a2)
>  nil a = (nil1 a, nil2 a)
>  edl b (c1,c2) = (edl1 b c1, edl2 b c2)
>  edr (c1,c2) b = (edr1 c1 b, edr2 c2 b)
>  edlr b (c1,c2) b' = (edlr1 b c1 b', edlr2 b c2 b')
>  drem (c1,c2) = (drem1 c1, drem2 c2)
>  is (c1,c2)   = (is1 c1, is2 c2)
>  sr b (c1,c2) b' = (sr1 b c1 b', sr2 b c2 b')
>  hl b1 b2 u b2' b1' = (hl1 b1 b2 u b2' b1', hl2 b1 b2 u b2' b1')
>  sp b1 b2 (c1,c2) b2' b1' = (sp1 b1 b2 c1 b2' b1',sp2 b1 b2 c2 b2' b1')
>  bl u (c1,c2) = (bl1 u c1, bl2 u c2)
>  br (c1,c2) u = (br1 c1 u, br2 c2 u)
>  il r1 (c1,c2) r2 = (il1 r1 c1 r2 ,il2 r1 c2 r2)
>  ml b1 b2 (m1,m2) b2' b1' = (ml1 b1 b2 m1 b2' b1', ml2 b1 b2 m2 b2' b1')
>  mldr b1 b2 (m1,m2) d b2' b1' = (mldr1 b1 b2 m1 d b2' b1',
>                                  mldr2 b1 b2 m2 d b2' b1')
>  mladr b1 b2 (m1,m2) d b2' b1' = (mladr1 b1 b2 m1 d b2' b1',
>                                   mladr2 b1 b2 m2 d b2' b1')
>  mldlr b1 b2 d (m1,m2) d_ b2' b1' = (mldlr1 b1 b2 d m1 d_ b2' b1',
>                                      mldlr2 b1 b2 d m2 d_ b2' b1')
>  mladlr b1 b2 d (m1,m2) d_ b2' b1' = (mladlr1 b1 b2 d m1 d_ b2' b1',
>                                       mladlr2 b1 b2 d m2 d_ b2' b1')
>  mldladr b1 b2 d (m1,m2) d_ b2' b1' = (mldladr1 b1 b2 d m1 d_ b2' b1',
>                                        mldladr2 b1 b2 d m2 d_ b2' b1')
>  mladldr b1 b2 d (m1,m2) d_ b2' b1' = (mladldr1 b1 b2 d m1 d_ b2' b1',
>                                        mladldr2 b1 b2 d m2 d_ b2' b1')
```

```
>  mldl b1 b2 d (m1,m2) b2' b1' = (mldl1 b1 b2 d m1 b2' b1',
>                                  mldl2 b1 b2 d m2 b2' b1')
>  mladl b1 b2 d (m1,m2) b2' b1' = (mladl1 b1 b2 d m1 b2' b1',
>                                   mladl2 b1 b2 d m2 b2' b1')
>  addss (c1,c2) u = (addss1 c1 u, addss2 c2 u)
>  ssadd u (c1,c2) = (ssadd1 u c1, ssadd2 u c2)
>  cons (c1,c2) (c_1,c_2) = (cons1 c1 c_1, cons2 c2 c_2)
>  ul (c1,c2) = (ul1 c1, ul2 c2)
>  combine (c1,c2) (c_1,c_2) = (combine1 c1 c_1, combine2 c2 c_2)
>  acomb (c1,c2) b (c_1,c_2) = (acomb1 c1 b c_1, acomb2 c2 b c_2)

>  h xs   = [(x1,x2)| x1 <- nub $   h1 [ y1 | (y1,y2) <- xs],
>                     x2 <-         h2 [ y2 | (y1,y2) <- xs, y1 == x1]]
>  h_i xs = [(x1,x2)| x1 <- nub $ h_i1 [ y1 | (y1,y2) <- xs],
>                     x2 <-       h_i2 [ y2 | (y1,y2) <- xs, y1 == x1]]
>  h_l xs = [(x1,x2)| x1 <- nub $ h_l1 [ y1 | (y1,y2) <- xs],
>                     x2 <-       h_l2 [ y2 | (y1,y2) <- xs, y1 == x1]]
>  h_s xs = [(x1,x2)| x1 <- nub $ h_s1 [ y1 | (y1,y2) <- xs],
>                     x2 <-       h_s2 [ y2 | (y1,y2) <- xs, y1 == x1]]
```

APPENDIX C

Usage of RNAshapes

The program RNAshapes has the following syntax:

```
RNAshapes -t [shape_type] -e [energy_range] -s [sequence]
shape_type:   1 - 5 (least - most abstract, default 5),
              p1 - p5 (probabilities of shapes)
energy_range: The energy range that is to be analysed in kcal/mol (default 1).
              This parameter has no effect when calculating probabilities.
sequence:     The raw sequence (if not given reads from stdin)

Examples:
> RNAshapes -t 5 -e 5 -s gttaatgtagcttaataacaagatggataattgtatcccataaaca
        gttaatgtagcttaataacaagatggataattgtatcccataaaca
-4.5    ((((.((.....)).)))).....(((((....))))).......  [][]
-3.62   (((.(((.................(((((....)))))))).))).  []
0.0     .........................................  _

> RNAshapes -t p5 -s gttaatgtagcttaataacaagatggataattgtatcccataaaca
        gttaatgtagcttaataacaagatggataattgtatcccataaaca
-4.5    ((((.((.....)).)))).....(((((....))))).......  0.7720283       [][]
-3.62   (((.(((.................(((((....)))))))).))).  0.22614306      []
0.0     .........................................  1.5578709e-4    _
```