Algorithms for RNA Secondary Structure Analysis: Prediction of Pseudoknots and the Consensus Shapes Approach

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

vorgelegt von
Jens Reeder

Bielefeld im Dezember 2007

Gedruckt auf alterungsbeständigem Papier °° ISO 9706

Acknowledgments

First of all, I thank my supervisor, Robert Giegerich, who brought me into the RNA field. I enjoyed many fruitful discussions with him.

I thank Ivo Hofacker for appraising this thesis and for numerous hints concerning RNA bioinformatics, he gave me throughout the last couple of years.

I thank Peter Steffen who developed the ADP compiler and extended it to my needs. Without him, my programs would be much slower.

I thank all of my colleagues who helped me to solve the everyday trouble in the life of a bioinformatician and for proofreading this thesis.

I thank Corinna Theis who developed *KnotInFrame* in her bachelor thesis. I also thank Alexander Hosfeld for evaluating my programs on the BRAliBase in his diploma thesis.

I thank Janina Reeder for carefully proofreading this thesis, but moreover I thank her for supporting me over all the years and for starting a family with me.

I am thankful for my daughter Emma who always makes me smile when work troubles me.

Finally, I thank my parents for their support.

Contents

Chapter 1

Introduction

Our understanding of the role of RNA has undergone a major change in the last decade. Once believed to be only a mere carrier of information and structural component of the ribosomal machinery in the advent of the genomic age, it is now clear that RNAs play a much more active role. RNAs can act as regulators and can have catalytic activity – roles previously only attributed to proteins. There is still much speculation in the scientific community as to what extent RNAs are responsible for the complexity in higher organisms which can hardly be explained with only proteins as regulators [Mattick, 2003].

In order to investigate the roles of RNA, it is therefore necessary to search for new classes of RNA. For those and already known classes, analyses of their presence in different species of the tree of life will provide further insight about the evolution of biomolecules and especially RNAs. Since RNA function often follows its structure, the need for computer programs for RNA structure prediction is an immanent part of this procedure. The secondary structure of RNA – the level of base pairing – strongly determines the tertiary structure. As the latter is computationally intractable and experimentally expensive to obtain, secondary structure analysis has become an accepted substitute. In this thesis, I present two new algorithms (and a few variations thereof) for the prediction of RNA secondary structures.

The first algorithm addresses the problem of predicting a secondary structure from a single sequence including RNA pseudoknots. Pseudoknots have been shown to be functionally relevant in many RNA mediated processes. However, pseudoknots are excluded from considerations by state-of-the-art RNA folding programs for reasons of computational complexity. While folding a sequence of length n into unknotted structures requires $O(n^3)$ time and $O(n^2)$ space, finding the best structure including arbitrary pseudoknots has been proven to be NP-complete [Akutsu, 2000; Lyngsø and Pedersen, 2001]. Nevertheless, I demonstrate in this work that certain types of pseudoknots can be included in

the folding process with only a moderate increase of computational cost.

In analogy to protein coding RNA, where a conserved encoded protein hints at a similar metabolic function, structural conservation in RNA may give clues to RNA function and to finding of RNA genes. However, structure conservation is more complex to deal with computationally than sequence conservation. The method considered to be at least conceptually the ideal approach in this situation is the Sankoff algorithm [Sankoff, 1985]. It simultaneously aligns two sequences and predicts a common secondary structure. Unfortunately, it is computationally rather expensive – $O(n^6)$ time and $O(n^4)$ space for two sequences, and for more than two sequences it becomes exponential in the number of sequences! Therefore, several heuristic implementations emerged in the last decade trying to make the Sankoff approach practical by introducing pragmatic restrictions on the search space.

In this thesis, I propose to redefine the consensus structure prediction problem in a way that does not imply a multiple sequence alignment step. For a family of RNA sequences, my method explicitly and independently enumerates the near-optimal abstract shape space and predicts an abstract shape as the consensus for all sequences. For each sequence, it delivers the thermodynamically best structure which has this shape. The technique of abstract shapes analysis is employed here for a synoptic view of the suboptimal folding space. As the shape space is much smaller than the structure space, and identification of common shapes can be done in linear time (in the number of shapes considered), the method is essentially linear in the number of sequences. Evaluations show that the new method compares favorably with available alternatives.

Organization of this thesis

I begin with a short review of the current biological knowledge of RNA, and in particular non-coding RNA.

In Chapter 3, I include some basic definitions of the RNA folding problem and describe some common ways of representing secondary structures. Then, I present three RNA folding algorithms and recast their recurrences into the ADP approach.

In Chapter 4, I introduce the pseudoknot as a structural element of RNA secondary structure. I devise a classification scheme for pseudoknots and show their biological importance. After that, I summarize the previous algorithmic approaches for RNA pseudoknot folding.

Chapter 5 constitutes the first main part of this thesis. Here, I motivate and define a new class of pseudoknots, the class of canonical simple recursive pseudoknots. I continue with its implementation in the tool *pknotsRG* followed by a detailed evaluation. I demon-

strate how Sparse Dynamic Programming can be used to accelerate pseudoknot folding. The chapter closes with a discussion of some algorithmic extensions as well as the use of a modified version of *pknotsRG* designed for the detection of ribosomal frameshift signals.

The second main algorithmic achievement of this thesis is reported in Chapter 6. Following a brief discussion of the general background, I describe the consensus shape approach, a new way for comparative secondary structure prediction, and show its performance in comparison to other algorithms. In analogy to the original Sankoff algorithm I also present a variant of the consensus shape method which computes aligned consensus structures. In the end, I combine the pseudoknot prediction algorithm and the consensus structure method into a powerful comparative pseudoknot predictions tool.

I conclude with a discussion and some implications of my work.

Previous publications

Major parts of the work reported here have already been published in the scientific literature:

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

2. **Jens Reeder** and Robert Giegerich. Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. *Bioinformatics*, 21(17): 3516–3523, 2005.

3. **Jens Reeder**, Peter Steffen, and Robert Giegerich. pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows. *Nucleic Acids Research*, 35(suppl_2): W320–324, 2007.

Additional work, related to this work in a more indirect way has appeared in:

4. Peter Steffen, Björn Voß, Marc Rehmsmeier, **Jens Reeder**, and Robert Giegerich. RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics*, 22(4): 500–503, 2006.

5. **Jens Reeder**, Matthias Höchsmann, Marc Rehmsmeier, Björn Voß, and Robert Giegerich. Beyond Mfold: Recent advances in RNA bioinformatics. *Journal of Biotechnology*, 124(1): 41–55, 2006.

6. Janina Reeder, **Jens Reeder**, and Robert Giegerich. Locomotif: From graphical motif description to RNA motif search. *Bioinformatics*, 23(13): i392–400, 2007.

Chapter 2

Biological Background

2.1 RNA structure

RNA (**r**ibo**n**ucleic **a**cid) is a linear macro-molecule composed of four different monomers, namely the nucleotides Adenine (A), Cytosine (C), Guanine (G), and Uracil (U). Its sugar-phosphate backbone covalently connects successive nucleotides via a phosphodiester bond. Unlike DNA (**d**eoxyribo**n**ucleic **a**cid), RNA ususally occurs single stranded with some exceptions, e.g. in double stranded RNA viruses. Due to base pair complementarity mediated by hydrogen bonds, *intra*-molecular base pairings can be formed, and thus, the RNA molecule folds back onto itself. The most common base pairings are the Watson-Crick pairs $A \bullet U$ and $G \bullet C$, but $G \bullet U$ pairs also occur frequently. When several base pairs stack directly on top of each other, they form a helical region similar to double stranded DNA. This stacking of base pairs has a stabilizing effect on the molecule's structure, caused by the overlap of the π-orbitals of the nucleotides' ring systems. However, helix formation always leads to the inclusion of an otherwise free moving, unpaired loop region which destabilizes the molecules' structure. Depending on the enclosing number of base pairs, one distinguishes between hairpin loops (closed by one base pair), bulge and internal loops (closed by two base pairs), and multiloops - closed by more than two base pairs (Figure 2.1). Under equilibrium conditions, an RNA molecule folds into the structure that minimizes the sum of stabilizing and destabilizing effects. This folding is often called the *minimum free energy* (MFE) structure.

RNA molecules fold hierarchically, i.e. first, the base pairings are formed, and later, the helices arrange themselves to a precise 3-dimensional structure [Tinoco and Bustamante, 1999]. The 3-dimensional fold is often stabilized by non standard base pairs, triple base pairs, and backbone-loop interactions. Nevertheless, the 2-D structure – defined as the set of canonical base pairs – builds a scaffold for the 3-D structure. That is why biologists

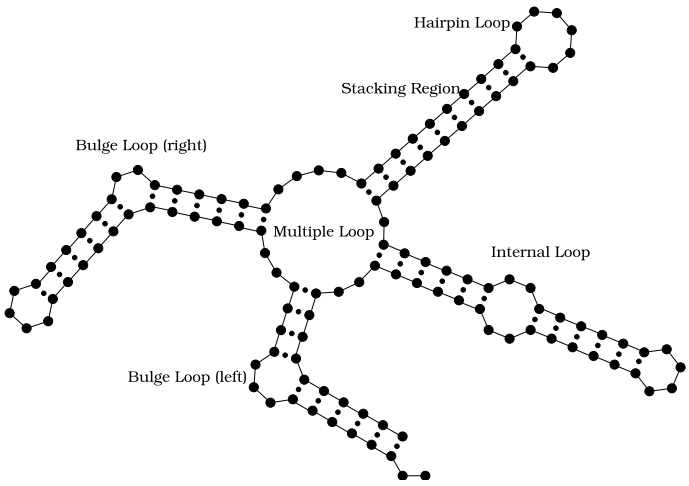


Figure 2.1: RNA folds back onto itself and hereby forms helices (stacking regions) and loop elements.

can make assumptions on the function or the relationship of two RNA molecules simply by comparing their secondary structure.

2.2 Roles of RNA

RNA has many roles, some of which were known for over 50 years, but the meaning and importance of some new classes of RNA have become visible just recently.

Following the long time unchallenged dogma of molecular biology, "DNA makes RNA makes protein". This basically means that the genetic information stored in the genomic DNA is transcribed into *messenger RNA* (mRNA) which serves as an information carrier to the ribosome, where the protein is finally synthesized from the RNA template. In this scheme, the mRNA acts as a linear molecule without any structural information. However, there are some known exceptions to this rule, where (mostly) small structural motifs within the mRNA influence the translation into the protein. One can distinguish between motifs occurring within the protein coding reading frame, e.g. ribosomal frameshift inducing motifs ([Dinman, 2006], see also Chapter 5.6), and motifs occurring in the 5' and 3' untranslated region (UTR), such as the well-known iron response element [Kikinis *et al.*, 1995].

In addition to the role as mRNA, there is a momentarily fast growing fraction of RNAs which are transcribed from genomic DNA but never get translated into proteins. Those kind of RNAs are subsumed under the name of *non-coding RNA* (ncRNA).

The two best and longest known classes of RNA are key players in the protein building

machinery, namely the *transfer-RNA* (tRNA) and the *ribosomal RNA* (rRNA). The tRNA acts as an adaptor molecule and mediates the translation of the triplet code encoded in the mRNA into a sequence of amino acids. Translation takes place at the ribosome which is a huge ribonucleo-protein-complex consisting of several different rRNAs and proteins. It is now evident that the catalytic activity of the ribosome is performed solely by RNAs. The proteins are only attributed a structure-stabilizing role.

Another class of *small nuclear RNAs* (snRNA) is involved in the splicing process of eukaryotic mRNAs. The first observed members of this class were rich in Uracil which led to the name uRNA.

A recently discovered class of non-coding RNAs are *microRNAs* (miRNAs) [Lee and Ambros, 2001; Lagos-Quintana *et al.*, 2001; Lau *et al.*, 2001] which are small, 21-24 nucleotide long RNAs processed from a ∼90 nucleotide long hairpin precursor. miRNAs act as translational repressors by binding to complementary sequences within protein-coding mRNA, oftentimes with a tissue- or stage-specific pattern.

Many rRNAs contain modified nucleotides. The locus of the modification is targeted by *small nucleolar RNAs* (snoRNAs) which by complementarity guide site-specific methylations or pseudouridylations to the rRNA.

In principle, tRNAs, miRNAs, and snoRNAs all follow the same pattern: In cooperation with some proteins they exert their function in a target-sequence specific manner. The RNA has the role of identifying the target by base pair complementarity, and the protein then performs its, perhaps catalytic, function on the target. Thus, every molecule has its perfectly designed part on the molecular stage.

Apart from this scheme, there are also RNAs which act without the help of a protein and regulate gene expression directly by building a duplex with the target mRNA. One such *regulatory RNA* is OxyS, first discovered in *E. coli*. By binding to its target mRNA of the gene *fhlA*, it renders the ribosome binding site (RBS) inaccessible and thus downregulates translation [Argaman and Altuvia, 2000]. Interestingly, OxyS also upregulates the activity of another gene (*rpoS*) by titration of Hfq protein. In absence of OxyS, Hfq binds to a sequence of *rpos* mRNA which otherwise forms an inhibitory intramolecular structure [Altuvia and Wagner, 2000].

At the moment, there is much speculation if the already known ncRNAs are remnants of an ancient RNA world or just the tip of the iceberg. It has been stated that the complexity of higher organisms cannot solely be achieved by (regulatory) proteins, but that there must be a hidden layer of regulatory ncRNAs [Mattick, 2003]. Hopefully the near future will tell us, if this idea proves right. I am convinced that the tools developed in this thesis will be a help in this task.

Chapter 3

RNA Folding

In this section, I introduce three basic Dynamic Programming algorithms for RNA secondary structure prediction. The last one serves as the basis for the extensions necessary to include pseudoknots as described in Chapter 5. The algorithms are explained in traditional matrix recurrences and also in ADP style. ADP (*Algebraic Dynamic Programming*) is a domain specific language for solving Dynamic Programming algorithms over sequential data. I introduce its basic concepts needed for understanding the algorithms presented later in this thesis. However, I do not cover all aspects of the ADP language. For details of ADP I refer to [Giegerich *et al.*, 2004a; Steffen and Giegerich, 2005; Giegerich and Steffen, 2006].

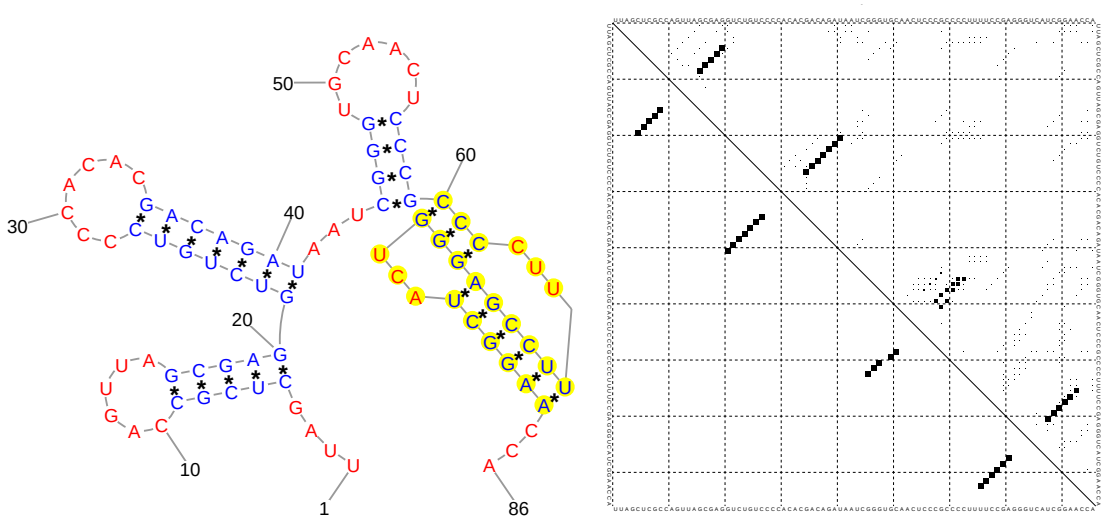
3.1 RNA secondary structures

3.1.1 Basic definitions

From a computer science point of view, RNA sequences are strings over the nucleotide alphabet A, C, G, U. In the following, s will denote a sequence, n its length. A secondary structure is a set $\mathcal{S}$ of base pairs, where allowed pairs are $G \bullet C$, $A \bullet U$ and $G \bullet U$. For every two RNA base pairs $i \bullet j$ and $k \bullet l$ from $\mathcal{S}$ with $i < k$, the following must hold:

1. $j - i > 3$
2. If $i = k$ then $j = l$
3. Either $i < j < k < l$ or $i < k < l < j$

Rule 1 imposes a minimal hairpin loop size of three nucleotides, rule 2 disallows triple base pair interactions, and finally, rule 3 ensures that base pairs either precede each other or are



(a) Squiggle plot

(b) Dot plot

(c) Arc representation

```
UUAGCUCGCCAGUUAGCGAGGUCUGUCCCCACACGACAGAUAAUCGGGUGCAACUCCCGCCCCUUUUCCGAGGGUCAUCGGAACCA
....(((((......))))).((((((.......)))))).....((((.......))))(((...[[[[[[))...]]]]]]...
```

(d) Dot-bracket notation

Figure 3.1: Equivalent representations of an RNA secondary structure with a pseudoknot at its 3' end. The dot plot (b) was produced by the Vienna RNA package and thus does not show the pseudoknot in the MFE structure on the lower left triangle.

properly nested. Exceptions to the third rule introduce pseudoknots which are generally not allowed in secondary structures. However, in Chapter 3, I relax this definition in order to include some pseudoknots into the notion of secondary structure.

3.1.2 Representations of secondary structures

The secondary structure of RNA molecules can be represented in various ways. The most common and obvious representation is the so called *squiggle* plot (Figure 3.1 a), in which helices are drawn double-stranded and loops as arcs connecting the helices. This type of

representation is widely used in the biological literature. An alternative, but equivalent representation is the *arc-* or *domes*-representation (Figure 3.1 c), where the backbone is drawn as a horizontal line and the base pairings as arcs. For properly nested structures, one can draw all arcs in the area above the backbone without any crossing lines. For pseudoknots, arcs beneath the backbone are necessary to minimize crossings. In the computer science community the *dot-bracket*-notation (Figure 3.1 d) is often used, where a dot represents an unpaired nucleotide, and each pair of opening and closing brackets of the same type (rounded, curly, or squared) symbolizes a base pair.

In a *dot plot* (Figure 3.1 b), two types of information are displayed. In the lower left half, exactly one structure is defined. Here, a dot at the intersection of row i and column j denotes a base pair of nucleotides i and j. In the upper right half, an ensemble of structures is displayed. Here, each dot stands for the probability of this particular base pair to be formed in the equilibrium ensemble. Usually, the size of the dot is proportional to the probability.

3.2 Algorithms for RNA secondary structure prediction

3.2.1 Maximizing the number of base pairs

The first algorithm for RNA folding was due to pioneering work of Nussinov [Nussinov *et al.*, 1978] in the late '70s. It seeks to find the structure with the maximal number of base pairs. Following a Dynamic Programming scheme, a matrix M is filled recursively. The value of entry $M(i,j)$ is defined as the maximum number of base pairs of the subsequence $s_i ... s_j$. The calculation of M follows the observation that the base at position j is either unpaired or paired to some base k. In the latter case, the sequence is divided into two parts: the subsequence from i to $k-1$ and the subsequence from $k+1$ to $j-1$ in the interior of the base pair. Formulated as a matrix recurrence:

$$M(i,j) = \begin{cases} 0 & \text{if } j \leq i+1 \\ M(i,j-1) & \\ \min\limits_{i \leq k < j} (M(i,k-1) + M(k+1,j-1) + 1) & \text{with } basepair(k,j) \end{cases}$$

There are many ways how the matrix can be calculated, e.g. row by row or column by column, as long as the computation proceeds from shorter to longer subsequences. The structure which finally achieves the maximal number of base pairs (stored in $M(1,n)$) can be deduced by backtracking the optimal solution through the dynamic programming matrix. The matrix can be stored in a triangular fashion but still requires $O(n^2)$ memory space. The runtime is in $O(n^3)$ due to the moving k in the last recursion.

ADP tree grammar

Now, I rewrite the same algorithm in ADP, beginning with the *tree grammar* which defines the search space, i.e. the set of admissible structures over the input sequence.

```
M = tabulated (
     nil      <<< empty        |||
     unpaired <<< M ~~- base   |||
     split    <<< M ~~! (pair <<< base -~~ M ~~- base) `with` basepairing ... h)
```

In the first line, I specify that the result of `M` parses are stored in a table by using the keyword `tabulated`. The second line initializes empty sequences with the algebra function `nil`. Here, the `<<<` operator feeds the result of the `empty` parser into the algebra function `nil`. The `empty` parser succeeds only on empty (zero length) strings. The next alternative in line three, connected via the `|||` operator, skips one base on the 3' end. In ADP, operators containing a `~` character (such as the most general `~~~`) partition the input sequence and direct the resulting subsequences into the left- and right-hand side parsers. The specialized `~~-` operator divides the input string into two parts: Here, the left part of size $j - i - 1$ recursively calls the result of `M`. The right part of size 1 is consumed by a single character parser `base`. This case is evaluated by the algebra function `unpaired`. Finally, there is the most complex alternative in line four. It splits the input into the base pair $k \bullet j$, the subsequence to the left of k, and the subsequence in the interior of the base pair. The `basepairing` predicate is applied on the two nucleotides parsed by the `base` parsers via the `with` keyword. The specialized `~~!` combinator guarantees a minimal hairpin loop size of one nucleotide. In the end, a choice function `h` is applied on the list of alternatives via the `...` operator.

ADP signature and algebra

An ADP program is completed by the definition of at least one algebra. The algebra functions are restricted by a *signature* which serves as an interface to the evaluating algebras. The signature defines the number of algebra functions and for each function, the number and types of parameters and the result type. Our example requires the declaration of four algebra functions plus the mandatory choice function `h`:

```
type Algebra alphabet answer = (
   ()       -> answer,                          -- nil
   answer   -> alphabet -> answer,              -- unpaired
   alphabet -> answer -> alphabet -> answer,    -- pair
   answer   -> answer -> answer,                -- split
   [answer] -> [answer]                         -- h
   )
```

Here, the type variable `alphabet` stands for the input type and the result domain is specified by `answer`.

For a base pair maximization algebra, I instantiate the signature with the following algebra functions:

```
bpmax :: Algebra Char Int
bpmax = (nil, unpaired, pair, split, h) where

  nil        ()  = 0
  unpaired   a b = a
  split      a b = a + b
  pair     l a r = a + 1

  h []           = []
  h xs           = [maximum xs]
```

The complete, executable code of this algorithm along with more algebra functions (counting, pretty printing and the generic algebra combinator `***`) is given in the appendix.

It turned out that Nussinov-style base pair maximization has a limited prediction accuracy. Mainly, there are two reasons for this. First, not all base pairs are formed equally likely. Instead, the amount of a base pair's stabilizing contribution depends on the neighboring bases. The second reason stems from the destabilizing effect resulting from unpaired loop regions. For most loops, a logarithmic energy penalty is incurred which cannot be modeled in the Nussinov approach. These observations led to an improved algorithm based on free energy minimization.

3.2.2 RNA folding by free energy minimization

Free energy based RNA folding was first described by Zuker and Stiegler [1981]. The underlying energy model [Mathews *et al.*, 1999] has been refined several times over the last twenty years but remained the same in its principles: RNA secondary structures are stabilized by the stacking effect of two adjacent base pairs or by single bases stacking on a base pair at either end of a helix. These single stacking bases are often called dangling bases. The formation of helices in turn prevents the free movement of the included loop regions which destabilizes the folding. Hairpin, internal, and bulge loops are all attributed precisely measured energy values for smaller loop lengths and logarithmically increasing energies for larger loop lengths. Multiloops are scored with an affine energy model, with a large constant (+3.4 kcal/mol) for multiloop initiation and a smaller constant (+0.4 kcal/mol) for each helix protruding from the multiloop.

In the following, I neglect the effect of single base stacking for clarity and brevity. Including the dangling bases in the recursions quadruples all rules starting or ending a helix, with in each case one rule for no dangling base, for one dangling base at the 5' side, for one dangling base at the 3' side, and finally for dangling bases at both the 5' and 3' side. Also, I simplify the evaluation of multiloops. Instead of the above mentioned affine model, all multiloops are scored with a constant initiation parameter. This energy model has also been used in the algorithm of Zuker and Stiegler [1981] which I describe now.

$$W(i,j) = \begin{cases} V(i,j) \\ W(i+1,j) \\ W(i,j-1) \\ \min\limits_{i<k<j}(W(i,k)+W(k+1,j)) \end{cases}$$

$$V(i,j) = \begin{cases} hl(i,j) \\ st(i,j)+V(i+1,j-1) \\ \min\limits_{i<i'<j'<j}(il(i,i',j',j)+V(i',j')) & \text{with } i'-i+j-j'>2 \\ \min\limits_{i+1<k<j-1}(W(i+1,k)+W(k+1,j-1)) \end{cases}$$

Similar to the M matrix in the Nussinov algorithm, the matrix entry $W(i,j)$ stores the energy of the best folding of the subsequence $s_i \dots s_j$. The additional matrix V stores the minimum free energy of subsequence $s_i \dots s_j$, given that s_i and s_j form a base pair. Now, the benefit of matrix V is that one can compute the energy of, say, a structure starting with an internal loop opened by base pair $i \bullet j$ and closed by $k \bullet l$ directly by adding $V(k,l)$ and the internal loop contribution $il(i,k,l,j)$. Note that by definition $W(i,j) \leq V(i,j)$.

We can directly translate these recurrences into ADP introducing a few additional algebra functions (`openL, openR, split, ml`) and the terminal parser `loop` which reads an arbitrary, non empty sequence.

```
W = tabulated (
                      V          |||
    openL <<< base -~~ W          |||
    openR <<<          W ~~- base |||
    split <<< W    ~~~ W          ... h)

V = tabulated (
   (hl <<< base -~~ loop ~~- base                          |||
    st <<< base -~~ V    ~~- base                          |||
    il <<< base -~~ loop ~~~ V ~~~ loop ~~- base           |||
    ml <<< base -~~ W    ~~~ W ~~- base) `with` basepairing ... h)
```

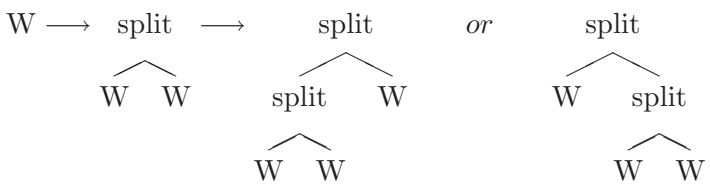


Figure 3.2: Ambiguous derivation for three adjacent `Ws`.

The algorithm as written above has a time complexity of $O(n^4)$ due to the handling of internal loops: For all opening base pairs $i \bullet j$, it iterates over all closing base pairs $i' \bullet j'$. Usually, RNA folding algorithms limit the size of internal loops to a small constant (e.g. 30), thus leading to an overall $O(n^3)$ time complexity. This heuristic is strongly supported by the energy model and only fails for artificial sequences. Nevertheless, there is an algorithm by Lyngsø *et al.* [1999] which uses a certain characteristic of the energy model in order to evaluate all possible internal loops within $O(n^3)$ time. In practice, most RNA folding tools resort to the simple heuristic of limiting internal loop size.

One can easily show that some structures can be derived from the grammar in more than one way. E.g. consider the case of three adjacent helices. The derivation always has to start with one W becoming two Ws using the *split* rule. Then, we can either further split the left W or the right W and end with three adjacent Ws in either way (see also Figure 3.2). As long as we are searching for only one minimal free energy structure, this ambiguity does no harm. But when it comes to folding space statistics, like Boltzmann weighted counting (partition function [McCaskill, 1990]), it must not be ignored.

3.2.3 Non-ambiguous energy based folding

I continue with an algorithm which overcomes all of the previous limitations and implements all features of the current energy model [Mathews *et al.*, 1999]. Basically, it is an ADP reimplementation of the grammar underlying *RNAsubopt* [Wuchty *et al.*, 1999]. Also, it will serve as the framework for the extensions allowing for pseudoknots in Chapter 5.

In order to remove ambiguity, structures have to be derived in a unique way. This is done by building the structure stepwise from the 5' end to the 3' end. With each splitting operation, exactly one helical substructure emerges at the leftmost (sub-)sequence end via the nonterminal `dangle`. This also holds true for the interior of multiloops.

The correct handling of dangling bases requires four slightly different rules for each of the nonterminals `dangle` and `multiloop`. All other dangling bases are handled implicitly by the algebra functions `hl,bl,br,il`. I already mentioned that the currently accepted

energy model for multiloops is more complex than the implementation of the previous section. It rather imposes an affine cost $ml = 3.4 + k * 0.4$, depending on the number k of helices protruding from the multi-loop. (In an earlier model, unpaired bases in the multi-loop were also penalized). This requires an additional matrix `ml_comps`, in which each use of the nonterminal `dangle` receives a 0.4 kcal/mol penalty.

As a further extension, I exclude isolated base pairs by always requiring two base pairs in a row in the rules for `closed` (the analogon to the V matrix). The grammar's axiom `struct` mirrors the W matrix.

```
struct = listed (
          sadd <<< base   -~~ struct |||
          cadd <<< dangle ~~~ struct |||
          nil  <<< empty             ... h)

dangle =  edl  <<< base -~~ closed ~~. loc  |||
          edr  <<<  loc .~~ closed ~~- base |||
          edlr <<< base -~~ closed ~~- base |||
          is   <<<  loc .~~ closed ~~. loc  ... h

closed = tabulated (
         (stack ||| hairpin ||| leftB ||| rightB ||| iloop ||| multiloop)
                                                        `with` stackpair ... h)

stack   = sr <<< base -~~ closed ~~- base
hairpin = hl <<< base -~~ base --~ region `with` (minloopsize 3)~~- base ~~- base
leftB   = bl <<< base -~~ base --~ region ~~~ closed           ~~- base ~~- base
rightB  = br <<< base -~~ base --~            closed ~~~ region ~~- base ~~- base
iloop   = il <<< base -~~ base --~ region ~~~ closed ~~~ region ~~- base ~~- base

multiloop =
   mldl  <<< base -~~ base --~ base --- ml_comps1          ~~- base ~~- base |||
   mldr  <<< base -~~ base --~          ml_comps1 ~~- base ~~- base ~~- base |||
   mldlr <<< base -~~ base --~ base --- ml_comps1 ~~- base ~~- base ~~- base |||
   ml    <<< base -~~ base --~          ml_comps1          ~~- base ~~- base ... h
         where
         ml_comps1 = tabulated (
                     sadd  <<< base   -~~ ml_comps1 |||
                     cadd  <<< dangle ~~~ ml_comps  ... h)

         ml_comps  = tabulated (
                     sadd  <<< base   -~~ ml_comps |||
                     cadd  <<< dangle ~~~ ml_comps |||
                     addss <<< dangle ~~~ uregion  ... h)
```

For efficiency reasons, some specialized operators are needed: `--~` splits the two leftmost characters from the input sequence, `---` the three leftmost characters. `~~.` and `.~~` do not split the sequence but rather return, in combination with the `loc` parser, their actual positions in the sequence. This is needed for the correct handling of single base stacking implemented in algebra functions `edl, edr, edlr`. Note that the nonterminal

`struct` is always called with a fixed right boundary, namely n. Hence, I do not need a quadratic table to store its results, but only a linear one. The tabulation of nonterminal `mlcomps1` is optional and leaves some room for time-space tradeoffs within the asymptotic efficiency class.

Chapter 4

Pseudoknot Folding

In this chapter, I introduce pseudoknots as structural elements of RNAs. Currently, there is no accepted nomenclature for different kinds of pseudoknots. Therefore, I shortly define and explain the most prominent classes of pseudoknots. After that, I summarize selected algorithms for pseudoknot prediction and classify them with respect to their generality.

4.1 Classification of pseudoknots

RNA pseudoknots arise when bases within a loop region of a regular secondary structure, enclosed by at least one base pair, interact with bases *outside* of the loop. Those bases violate the nesting assumption of secondary structures. Formally, I define pseudoknots as the set of base pairs which overlap with at least one other base pair:

$$\mathcal{S}_{pk} = \{(i \bullet j) \mid \exists\ (k \bullet l) : i < k < j < l\ \vee\ k < i < l < j\}$$

In some publications, all kinds of pseudoknots are generally regarded as part of the tertiary structure. I have a different viewpoint accepting all Watson-Crick and $G \bullet U$ pairs as part of the secondary structure, whether they are properly nested or not. The tertiary structure then includes non-standard base pairs, triple base pairs, and all other types of bondings.

4.1.1 H-type pseudoknots

The most simple pseudoknot forms when a hairpin loop interacts with bases outside of the loop, generating two helices and two loops (Figure 4.1). This directly implies that the two helices stack on top of each other, building a quasi-continuous helix. A single unpaired base may occur at the interface of the helices, since it does not disrupt the helix stacking.

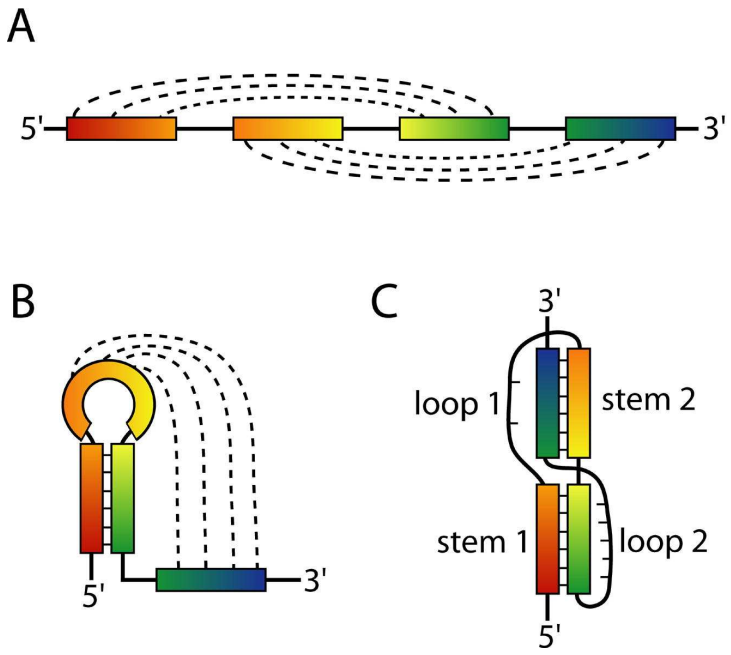


Figure 4.1: An H-type pseudoknot is formed by interaction of a hairpin loop with bases outside the loop. A: Linear representation with base pairs drawn as arcs over and under the sequence. B: First, the hairpin forms, then the pseudoknot. C: Coaxial arrangement without intervening unpaired bases leads to the H-type pseudoknot. Image reproduced from [Staple and Butcher, 2005]

H-type pseudoknots are the best studied non-nested structures with many known biological examples. There is even an approximation for loop free energy values available, based on polymer thermodynamics and fitting to known pseudoknot data [Gultyaev *et al.*, 1999]. In some definitions, the H-type pseudoknot helices are allowed to have (small) bulges and internal loops. I call such knots *relaxed H-type pseudoknots*.

4.1.2 Simple pseudoknots

A *simple pseudoknot* also contains two helices but has no restrictions on the size of the inner loop. The helices may be interrupted by bulges and internal loops, but no interactions from the loop regions to other loops are allowed. Due to its three loops, a simple pseudoknot has much more freedom to explore alternative stacking conformations. A *simple recursive pseudoknot* has further secondary structure elements (including pseudoknots) within its loops, as long as they require only pairings with bases from that particular loop. In other words, no *inter*-loop pairings are allowed. For further reference, I call this class *sr-PK*.

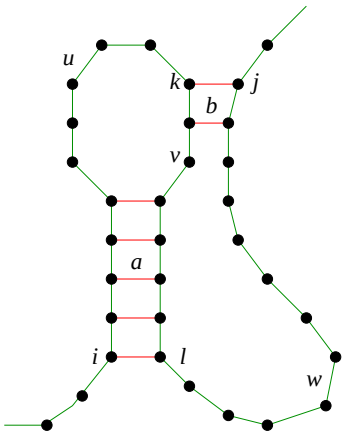


Figure 4.2: A simple pseudoknot, formed by helices $a - a'$ and $b - b'$, with intervening sequences u, v, w. If the internal parts u, v, w contain further secondary structures, it is called a simple recursive pseudoknot (sr-PK).

4.1.3 Planar pseudoknots

Planar pseudoknots are defined as structures which can be drawn on a plane without any crossing of arcs. This allows for structures such as the frequently seen *kissing hairpin* pseudoknot, but also for arbitrarily long chains of pseudoknot helices of the type $ABA'CB'DC'D'E \ldots YX'Y'$ where $A - A'$, $B' - B'$, ... each denote one or more continuous base pairs.

4.1.4 Non-planar pseudoknots

The most complex pseudoknot class is entitled *non-planar pseudoknots*. It subsumes all interactions not covered by one of the above classes. To my knowledge, there is only one example of a non-planar pseudoknot, namely the pseudoknot surrounding the alpha operon ribosome binding site in bacteria ([Schlax *et al.*, 2001], see Figure 4.3). Its helix crossing scheme is $ABCDA'C'B'D'$ and cannot be drawn on a plane without any crossing lines. The absence of more examples for this class should not lead to the conclusion that this class may not be biologically plausible. Since there are no efficient tools available which can systematically search for all non-planar pseudoknots, their detection is to some extent a matter of luck (or wet-lab experiment).

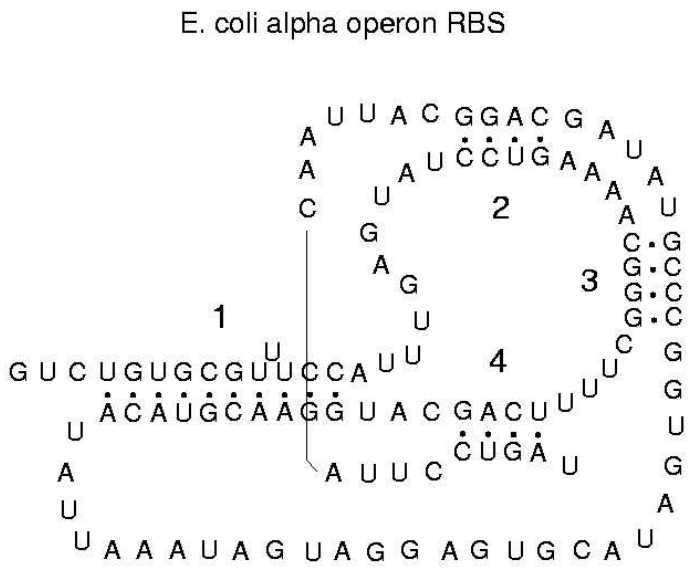


Figure 4.3: The non-planar pseudoknot surrounding the alpha operon ribosome binding site in bacteria.

4.2 Biological relevance

Pseudoknots have been shown to be functionally relevant in many RNA mediated catalytic processes. Examples are the RNA of the telomerase ribonucleoprotein complex [Chen *et al.*, 2000] and the self-splicing group I introns [Cech, 1988] where the pseudoknot establishes the catalytic core. Pseudoknots were located in the mRNA of prion proteins of humans and are confirmed for many other species [Barette *et al.*, 2001].

A complex double pseudoknot is the hepatitis delta virus (HDV) ribozyme [Ferré-D'Amaré *et al.*, 1998]. HDV's circular genome is replicated in a rolling-circle mechanism leading to a long multi-genome RNA strand. The pseudoknot catalyzes the self-cleavage into genome length units required for virus packaging. The HDV ribozyme is the fastest known naturally occurring catalytic RNA with a reaction rate of one per second. The helix conformation of HDV pseudoknot is $ABCDC'A'D'B'$ classifying it as a planar pseudoknot.

Pseudoknots are also part of many non-catalytic processes. They are known to stimulate efficient programmed -1 ribosomal frameshifting (-1 PRF), a mechanism used by a wide range of RNA viruses to encode two proteins within one genomic region. Recently, an unusual three-stemmed pseudoknot has been identified that promotes programmed -1 ribosomal frameshifting in the SARS coronavirus [Plant *et al.*, 2005]. This pseudoknot is thought to pause the ribosome during translation, which then shifts back by one nucleotide on the heptameric "slippery site". The pseudoknot seems to be conserved in all

coronaviruses, and thus could be a target for anti-viral therapeutics. In a study by [Jacobs *et al.*, 2007], over a thousand of potential -1 PRF signals were detected in the yeast genome. The majority of signals, however, seem to direct the ribosome to premature termination codons. This suggests a mechanism of post-transcriptional gene regulation through the nonsense-mediated mRNA decay pathway. Further genome wide studies have to elucidate this phenomenon. In Chapter 5, I present a search program for -1 PRF signals which is based on the work of this thesis.

4.3 Previous algorithmic work

Well established algorithms for the prediction of RNA secondary structures from a single sequence (*Mfold* [Zuker and Sankoff, 1984], *RNAfold* [Hofacker *et al.*, 1994]) are commonly based on the thermodynamic model of Mathews *et al.* [1999], returning the structure of minimal free energy. In spite of their importance, pseudoknots are excluded from consideration by these programs for reasons of computational complexity: While folding a sequence of length n into unknotted structures requires $O(n^3)$ time and $O(n^2)$ space, finding the best structure including arbitrary pseudoknots has been proven to be NP-complete [Akutsu, 2000; Lyngsø and Pedersen, 2001]. In fact, the proof given in [Lyngsø and Pedersen, 2001] uses a scoring scheme based on adjacent base pairs only. This is even simpler than the MFE model, because it neglects entropic energies from loops. These complexity results leave basically three routes to achieve practical algorithms.

Pseudoknot heuristics

The first route sacrifices the guarantee for optimality. Heuristic methods such as genetic algorithms [Shapiro and Wu, 1997] try to find good, if not optimal, pseudoknotted structures by randomly mutating a set of structures. A mutation is more successful, if it leads to more stable structures. Generally, genetic algorithms can find the optimal structure, but oftentimes, they will converge to a locally optimal structure. Then, the tool *Kinefold* [Isambert and Siggia, 2000; Xayaphoummine *et al.*, 2003] computes a stochastic simulation of RNA folding based on individual helix opening and closing operations. Finally, there are algorithms that try to build pseudoknot structures by clever iterative helix picking. In [Ren *et al.*, 2005], the probability of a helix to be part of the final structure depends solely on the energy of the helix. Of course, this neglects the destabilizing effect of loops which may lead to unpredictable results.

Simplification of the energy model

The second route is to consider pseudoknots in full generality, but to resort to an even more simplistic energy model. An $O(n^4)$ time and $O(n^3)$ space algorithm for base pair maximization has been given in [Akutsu, 2000], and $O(n^3)$ time algorithms based on maximum weight matching in [Tabaska *et al.*, 1998] and [Ruan *et al.*, 2004].

Recently, an $O(n^4)$ time and $O(n^3)$ space algorithm based on the technique of [Akutsu, 2000] using a thermodynamic model has been reported in [Deogun *et al.*, 2004]. It models destabilizing effects of loops bounded by nested structures but neglects the loops connecting the pseudoknotted parts. While it can handle simple pseudoknots consisting of more than two helices, it is restricted to non-recursive pseudoknots.

Search space reduction

Finally, the third route is the one also followed here: One retains the established thermodynamic model but restricts to a more tractable subclass of pseudoknots. For some quite general classes of pseudoknots, polynomial time algorithms have been designed: Rivas and Eddy achieve $O(n^6)$ time and $O(n^4)$ space [Rivas and Eddy, 1999]. This algorithm is available, and in spite of the high computational cost, it is actually used in practice. I shall call it *pknotsRE* for later reference. The key idea of their work is the use of four-dimensional so called *gap-matrices*. The matrix entry (i, j, k, l) with $i < k < l < j$ stores the best folding of subsequence $s_i \dots s_j$, *excluding* the inner part $s_k \dots s_l$. Combination of two gap matrices then produces a pseudoknot. By subsequent combination of multiple gap matrices, complex pseudoknot topologies can arise. The drawback of this powerful, but computationally expensive algorithm is the following paradox: Pseudoknots with complex helix interactions naturally require longer primary sequences than simpler ones. Yet, the high runtime complexity of $O(n^6)$ as well as the space consumption of $O(n^4)$ restricts the use of this algorithm to a maximal sequence length of around 150 nucleotides. Most of the pseudoknots predicted belong to a much simpler structural class and do not exhibit chains of crosswise interactions.

In [Dirks and Pierce, 2003], a simplification of [Rivas and Eddy, 1999] is proposed that also implements the partition function allowing for analyses such as probabilistic sampling. The main restriction is that pseudoknots are constructed of at most two gap matrices which in turn prohibits structures like kissing hairpins. The time complexity is reduced to $O(n^5)$ by an idea similar to the fast evaluation of internal loops [Lyngsø *et al.*, 1999]. However, the space requirements remain at $O(n^4)$. The algorithm is implemented in the software *NUPACK*.

Further improvements have been shown to be possible for yet more restricted classes,

e.g. the one by Lyngsø and Pedersen [2000]. Their model allows for structures composed of five components $s_1 s_2 s_1' s_2' s_1''$, where both $s_1 s_1' s_1''$ and $s_2 s_2'$ are nested structures. The algorithmic complexity is $O(n^5)$ time and $O(n^3)$ space, but to my knowledge, no implementation is available.

In addition to the maximal base pair algorithm in [Akutsu, 2000], the author also proposes an extension accounting for entropic loop effects. These modifications do not alter the class of predicted pseudoknots but raise the asymptotic runtime to $O(n^5)$.

4.3.1 A hierarchy of pseudoknot classes

Unfortunately, most of the above algorithms do not coincide with one of the pseudoknot classes. E.g., *pknotsRE* can predict the non-planar helix conformation of the alpha operon ribosome binding site but fails for other non-planar pseudoknots such as $ABCA'DB'EC'D'E'$. This makes it hard to relate the algorithms in terms of generality and efficiency. For some algorithms however, Condon and coworkers [Condon *et al.*, 2004] devised and proved a hierarchy of their pseudoknot complexity. In detail, they proved that

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

Following the terminology of [Condon *et al.*, 2004], PKF is the class of pseudoknot-free structures. L&P denotes a slightly simplified class of [Lyngsø and Pedersen, 2000]: Instead of five segments, the sequence may be decomposed into four segments $s_1 s_2 s_1' s_2'$, where both $s_1 s_1'$ and $s_2 s_2'$ are pseudoknot-free. I call the original class outlined in [Lyngsø and Pedersen, 2000] $L\&P^+$. D&P is from [Dirks and Pierce, 2003], A&U is the work by Akutsu [Akutsu, 2000], and R&E is the class of *pknotsRE*. As expected, the largest class has the steepest computational requirements, both in time and in space. It should be noted that the algorithm of D&P needs asymptotically more space than the A&U algorithm, although the first class is properly contained in the latter. This may stem from the fact that Dirks and Pierce also compute the partition function for pseudoknots of class D&P. Calculating the partition function requires the implicitly underlying grammar to be non-ambiguous. Yet, the A&U grammar is ambiguous and needs to be modified for partition function folding. It is not clear whether this can be done without requiring modifications which alter the asymptotic efficiency. A summary of these observations is given in Table 4.1.

The class of pseudoknots handled by the tool *pknotsRG*, developed in this thesis, is a proper subset of D&P and thus A&U and R&E, but not of L&P. It shares pseudoknots with L&P, but each class contains some type of pseudoknots exclusively. I elaborate on this fact in Section 5.1.5 and give a thorough presentation of *pknotsRG* in the next chapter.

	Asymptotic Efficiency		contains	Reference
	Time	Space		
PKF	$O(n^3)$	$O(n^2)$	all nested structures	
L&P	$O(n^5)$	$O(n^3)$	$ABA'B'$	[Lyngsø and Pedersen, 2000]
D&P	$O(n^5)$	$O(n^4)$	$ABCB'C'A'$	[Dirks and Pierce, 2003]
A&U	$O(n^5)$	$O(n^3)$	$ABCB'DA'D'C'$	[Akutsu, 2000]
R&E	$O(n^6)$	$O(n^4)$	$ABA'CB'C'$	[Rivas and Eddy, 1999]
PK	NP		$ABCA'DB'EC'D'E'$	

Table 4.1: Classification of pseudoknot predicting algorithms following [Condon *et al.*, 2004]. The classes are listed with increasing complexity, with the least complex class PKF listed at the top. The example structure given for a particular class is not contained in all classes listed above this particular class. So, the pseudoknot topology $ABCB'C'A'$ is not contained in PKF and L&P, but in the other four classes. PKF is the class of pseudoknot-free, nested structures predicted by *RNAfold* and *Mfold*. PK contains all possible pseudoknots and is proven to be NP-complete, assuming a reasonable energy model.

Chapter 5

pknotsRG

In this chapter, I describe an algorithm for folding RNA secondary structures including pseudoknots under the MFE model which requires $O(n^4)$ time and $O(n^2)$ space. The algorithm considers the class of simple recursive pseudoknots, further restricted by three rules of canonization. Each simple recursive pseudoknot has a canonical representative which is recognized by the algorithm. I begin with some general considerations about a DP algorithm's asymptotic complexity. This directly leads to the definition of the class of pseudoknots captured by my algorithm, and I analyze its coverage of known pseudoknots. After a detailed description of the algorithm and its implementation in the software tool *pknotsRG*, I evaluate its predictive performance on biological data and finish with a discussion about some algorithmic variants. Parts of this chapter have already been published in [Reeder and Giegerich, 2004].

5.1 Canonical simple recursive pseudoknots

5.1.1 Anticipating the complexity of a DP algorithm

I start with a semi-formal discussion of how to estimate the efficiency of a DP algorithm for folding (or any kind of motif search) *before* it is described in detail. I consider elements of RNA structure as sequence motifs of different types: hairpins, bulges, multiloops, etc. I use the ADP notation introduced in Chapter 3. By an equation

```
m = f <<< a ~~~ b ~~~ c |||
    g <<< c ~~~ a
```

I specify that the sequence motif m can be composed in two alternative ways: The first case, labeled by f, requires adjacent occurrences of motifs a, b, and c. The second case, labeled by g, requires adjacent occurrences of motifs c and a. When motif m is to be

scored, f and g act as the scoring functions that combine the local score contribution of each case with the scores of sub-motifs a, b, and c.

What is the computational effort of locating motif m in an input sequence x of length n, say at sequence positions i through j? First, I assume that all motifs can have an arbitrary size between 0 and n. The algorithm must consider all boundary positions (i, j) for motif m which requires at least $O(n^2)$ steps. In case g, it must consider all boundary positions k where motif c meets a, such that the runtime for case g is in $O(n^3)$. In case f, there are two such moving boundaries k and l between the three sub-motifs, so I obtain $O(n^4)$ overall for motif m.

This can be improved if there is an upper bound on the size of some motif involved. If motif a is a single base, for example, the exponent of n decreases by one in both cases. Furthermore, if motif b is (say) a loop of maximal size 40, then one factor of n is reduced to a constant factor, and the overall asymptotic runtime is now $O(n^2)$. Sometimes a motif description can be restructured to improve efficiency by reducing the number of moving boundaries. Whether or not this is possible does not depend on the motif structure, but on the scoring scheme! This is a somewhat surprising observation from [Giegerich and Meyer, 2002], where such optimizations are studied, and where also the line of reasoning exercised here is given a mathematical basis.

In the sequel, I shall exploit another source of efficiency improvement. If the lengths of two sub-motifs are coupled somehow, say a and c have the same length, then the boundaries k and l in case f are related by $k - i = j - l$. When iterating over k, I can use $l := j - k + i$ (rather than $k \leq l \leq j$) and save another factor of n.

5.1.2 Canonization

When the search space of a combinatorial problem seems to be too complex to be evaluated efficiently, heuristics are employed. Canonization restricts the search space in a well-defined way, arguing that all relevant solutions in the full search space have a representative that is canonical, and hence, nothing relevant is overlooked. One such technique is the purging of structures that have isolated base pairs. Here, the plausibility argument refers to the underlying energy model, where base pairings without stacking have little or no stabilizing effect. This canonization does not affect efficiency, but it achieves a significant reduction of the search space (figures in [Giegerich, 2000]), which renders the enumeration of near-optimal solutions [Wuchty *et al.*, 1999] much more meaningful.

I shall introduce three canonization rules that reduce the class sr-PK to the class of *canonized simple recursive pseudoknots*, csr-PK. Using the notation introduced above, the

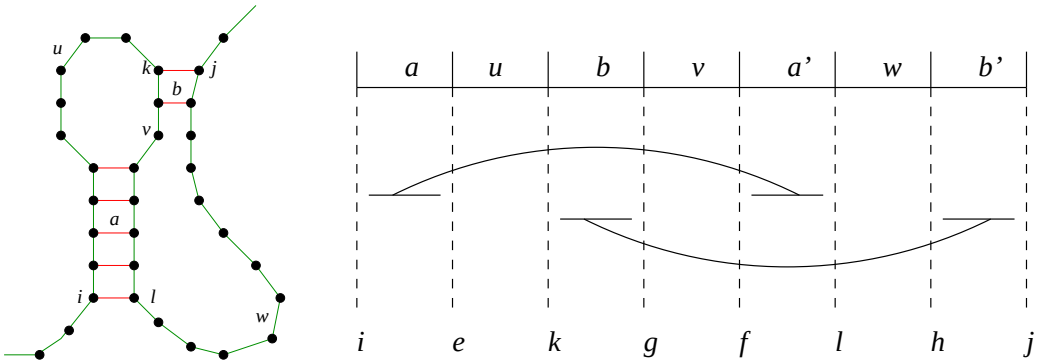


Figure 5.1: Eight moving boundaries delineating a simple (recursive) pseudoknot.

motif definition of a simple recursive pseudoknot is given by

```
knot = knt <<< a ~~~ u ~~~ b ~~~ v ~~~ a' ~~~ w ~~~ b'
```

with boundaries at sequence positions $i, e, k, g, f, l,\ h, j$ as shown in Figure 5.1.

Segment a forms a helix with a', and b with b'. Segments u, v, and w can have arbitrary structures including pseudoknots. Naively implemented, one can expect a DP algorithm of time complexity $O(n^8)$ according to the efficiency estimation technique introduced above. Now, I apply canonization. Note that it only applies to helices forming pseudoknots; other helices are unaffected. I first present the technical aspects, while the discussion of these restrictions is deferred to the next section.

Canonization Rule 1:

(a) Both strands in a helix must have the same length, i.e. $|a| = |a'|$ and $|b| = |b'|$.

(b) Both helices must not have bulges or internal loops.

Note that (b) is a stronger restriction and trivially implies (a). Under the regime of Rule 1 we may conclude:

$$\begin{aligned} f &= l - (e - i) \\ h &= j - (g - k) \end{aligned}$$

As a result, six out of eight boundaries are left that vary independently, and runtime is down to O(n^6).

Canonization Rule 2:

> The helices a, a' and b, b' facing each other must have maximal extent, or in other words, compartment v must be as short as possible under the rules of base pairing.

Observe that the maximal length of a and a' is fixed once i and l are chosen. The maximal helix length $stacklen(i, l)$ can be precomputed and stored in an $O(n^2)$ table. The same observation holds with respect to the other helix, and we fix

$$\begin{aligned} e &= i + stacklen(i, l) \\ g &= k + stacklen(k, j) \end{aligned}$$

Thus, only four independently moving boundaries – i, k, l, j – are left, and we can hope to obtain an algorithm with runtime $O(n^4)$. Scores of pseudoknots found between i and j are stored in table $knot(i, j)$, and hence the space requirements are $O(n^2)$, which is the same asymptotic space efficiency as in the folding of unknotted structures.

A subtlety arises when both helices, chosen maximally, compete for the same bases, or in other words, the length of either u,v, or w would become negative. This case is addressed by

Canonization Rule 3:

> If two maximal helices overlap, I will distinguish the following scenarios depending on where the overlap occurs within the loop segment:
>
> u: Helix a is shortened.
>
> v: The boundary is fixed at an arbitrary point between a and b.
>
> w: Helix b is shortened.

Let m and m' be the helix lengths so determined. We finally obtain

$$\begin{aligned} e &= i + m \\ g &= k + m' \end{aligned}$$

The language of pseudoknots in class csr-PK can be defined by a simple context free grammar over an infinite terminal alphabet. Let a^k denote a terminal symbol of k times the letter a. The grammar uses a single nonterminal symbol S and its productions are

$$S \;\rightarrow\; . \;\mid\; .\,S \;\mid\; S\,. \;\mid\; S\;S \;\mid\; (\,S\,) \;\mid\; [^k\,S\,\{^l\,S\,]^k\,S\,\}^l$$

for arbitrary $k, l \geq 1$. For example, the simple pseudoknot of Figure 5.1 (left side) is represented by the string `..[[[[[.....{{.]]]]].........}}` .

This grammar is useful for judging how different an experimentally determined structure is from class csr-PK. It is not useful for programming, since it is ambiguous and does not distinguish the fine grained level of detail required by the energy model.

5.1.3 Canonical representatives

A careful discussion is required to show that each simple recursive pseudoknot, if not canonical by itself, has (a) a canonical representative of (b) similar free energy.

Rule 1 (b) affects the length of helices that are considered in forming the pseudoknot. Let there be a pseudoknot between i' and j'. It is not canonical if one of the two helices contains bulges or internal loops. However, there must be at least one pair of shorter helices without bulges at i, j with $i' \leq i$ and $j \leq j'$ which serves as a canonical representative, albeit with somewhat higher free energy.

Rule 2 is justified by the fact that the energy model strongly favors helix extension. Clearly, for each family of pseudoknots delineated by i, k, l, j, there is a canonical one with maximal helices whose free energy is at least as low – except for the following case: The maximal helices compete with the internal structure of u, v, and w. It may be possible to contrive a structure where shortening (say) helix $a - a'$ by one base pair allows to create two pairs with new partner bases in u and v, resulting in a structure which has slightly lower energy. Still, the free energy of the canonical pseudoknot must be very similar.

Finally, Rule 3 requires a decision where to draw the border between two overlapping helices competing for the same bases. If the overlap occurs in loop u, the choice of shortening helix a is well justified. The pseudoknot with a shorter b helix will be evaluated anyway by the recursion for $knot(i, j')$ with $j' < j$. The equivalent holds for an overlap in loop w. At last, there is the case where the overlap occurs in the middle loop v. An arbitrary decision here can only slightly affect free energy, as the same number of base pairs are stacked either on the $a - a'$ or the $b - b'$ helix.

Let $E(s)$ denote the free energy computed for structure s. Summing up, I have shown that for each simple recursive pseudoknot K, there is a canonical one C in the search space. While I cannot prove that $E(C) \leq E(K)$, I have argued that this is likely, and if not, the energies will at least be close. Still, there might be another, energetically optimal, canonical structure S (knotted or not) such that $E(K) < E(S) < E(C)$. In this case, if only the "best" structure S is reported, neither K nor its canonical representative C is observed. (A remedy to this is the computation of near-optimal structures.)

Simple recursive pseudoknots

csr-PK	1-nt bulge	Rule 2 violated	isolated base pair	GA base pair	total
166	15	7	18	3	209

Complex pseudoknots

large bulge	internal loop	triple helix	kissing hairpins	four helices	total
2	24	12	3	1	42

Table 5.1: Class membership of 251 pseudoknots from PseudoBase. The structures were determined by comparative sequence analysis and/or by experimental techniques. The largest amount of pseudoknots is simple recursive or even canonical simple recursive.

5.1.4 Evaluation of the class csr-PK

To evaluate how well the class csr-PK covers known pseudoknots, I considered 251 pseudoknot structures from PseudoBase[1], a database containing pseudoknot sequence and structure information deduced from the literature [van Batenburg *et al.*, 2001]. Some database entries were removed, since they contain large gap regions in their loop sequence and structures. The observations are shown in Table 5.1.

I found 209 simple recursive pseudoknots and 42 of more general shapes. I further detected that 166 out of the 209 pseudoknots lie in csr-PK, i.e. they are their own canonical representatives. Therefore, class csr-PK covers approximately 66% of PseudoBase.

Further, there are 43 simple pseudoknots which fail to qualify for class csr-PK for various reasons. 15 pseudoknots have a single nucleotide bulge in their helix. I could relax canonization Rule 1 to accommodate for such situations without altering the algorithm's efficiency. In fact, I implemented a prototype algorithm which allows for a 1-nucleotide-bulge. Another 18 pseudoknots contain isolated base pairs at the end of a helix, three have non-canonical base pairs and in seven pseudoknots, one of the helices does not have maximal extent.

Considering the remaining 20% complex pseudoknots, note that often pseudoknots in more general classes also have a similar representative in csr-PK. For example, the pseudoknot of hepatitis delta virus (Figure 5.5) consists of four interacting helices of shape $a-b-c-d-c'-a'-d'-b'$, where helix $d-d'$ is very short - only two base pairs. Deleting it, helix $c-c'$ is no longer crossing with other helices, and the pseudoknot falls within class csr-PK.

[1] `http://wwwbio.LeidenUniv.nl/~Batenburg/PKB.html`

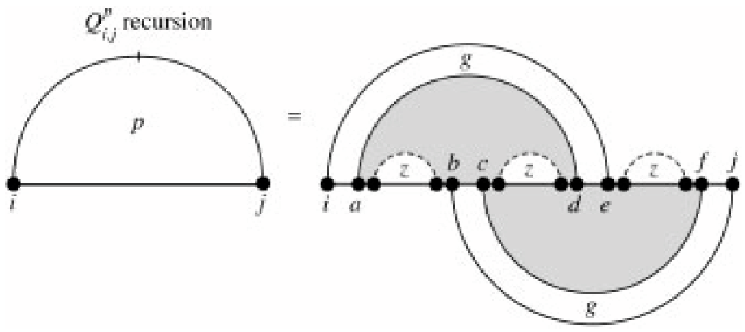


Figure 5.2: The main recursion introducing pseudoknots in *NUPACK* [Dirks and Pierce, 2003]. Two overlapping *g-regions* allow for non-nested base pairs. Recursive pseudoknots are realized by *z-regions*, the nonterminal for an arbitrary folding.

5.1.5 A hierarchy of pseudoknots - continued

How does class csr-PK blend into the hierarchy of [Condon *et al.*, 2004]?

The class of canonical simple recursive pseudoknots is a proper subclass of D&P (see Chapter 4.3 for an explanation and nomenclature of each class). This can easily be proven by the fact that each csr-PK can be composed by two gap matrices and three intervening loop segments, which is exactly what the recursions of D&P explicitly state (see Figure 5.2).

However, csr-PK is not a subclass of L&P. Since class L&P does not contain any recursive pseudoknots, it specifically does not contain any csr-PK. The other way round, L&P$^+$ does contain kissing hairpins, and the simpler class L&P also contains simple pseudoknots with bulges or internal loops. Both types of pseudoknots are not contained within csr-PK. But note that both L&P and csr-PK contain canonical simple pseudoknots.

5.2 Implementation

A necessary prerequisite for the implementation of the new pseudoknot folding algorithm is a scoring function, i.e. the underlying thermodynamic model. Since there is no well-established energy model for pseudoknots, I had to develop my own model.

5.2.1 A pseudoknot energy model

The lack of a well founded energy model is probably one of the largest difficulties in RNA pseudoknot prediction. Only for the least complex class, the H-type pseudoknots, a reasonable model does exist. In [Gultyaev *et al.*, 1999], Gultyaev and coworkers present

a free energy model based on the theory of polymer thermodynamics fitted to known H-type pseudoknots. Their key observations are that the loop energies depend on the loop *and* stem sizes. Furthermore, the asymmetry in RNA helices results in different energy contributions for each loop. Roughly speaking, there is a logarithmic decrease in stability with longer loops. The minimal destabilizing configuration of loop and stem sizes is penalized by 3.5 kcal/mol per loop, thus 7 kcal/mol total. With slightly longer loops, the added penalty quickly reaches 9 kcal/mol. For long loops, say 10 nucleotides, the loop energy ranges from 6 to 7.4 kcal/mol. It is not straightforward to extend their model to simple (and even recursive) pseudoknots, which is why I chose to define a new energy model.

Overall, the energy of a pseudoknot consists of stabilizing and destabilizing terms. Whenever possible, I use the values from the current thermodynamic energy model for properly nested RNA secondary structures [Mathews *et al.*, 1999]. As stabilizing terms, I count the nearest neighbor stacking energies of the pseudoknot helices and contributions of dangling bases at both ends of each helix. If the length of the middle part v is smaller or equal to 1, the pseudoknot helices stack coaxially on each other, and I further add the appropriate stacking energy. In [Rivas and Eddy, 1999], (which uses an outdated energy model for nested structures), a pseudoknot initiation parameter of 7 kcal/mol is proposed. This consideration is inspired by the treatment of multiloops which also incur an initiation penalty. However, I found that setting this value to 9 kcal/mol results in better performance using the current energy model. Moreover, the relatively high initiation parameter acts as a safeguard against too many false positive pseudoknots. My observation is supported by the similar choice made by Dirks and Pierce [2003]. Finally, I penalize each unpaired nucleotide inside a pseudoknot loop with 0.1 kcal/mol. This seems to be a good approximation of the values given in [Gultyaev *et al.*, 1999]. Compared to their model, my approach probably underestimates the stability of H-type pseudoknots with very short loops but agrees well for the remaining majority.

5.2.2 ADP implementation

As a starting point, I use the algorithm of Section 3.2.3 which is based on the standard MFE model with dangling bases. It is non-ambiguous and requires $O(n^3)$ time and $O(n^2)$ space.

The implementation strictly follows the outline given in the previous section, except that a considerable amount of detail, related to the energy model, has to be taken care of. While ADP bans the use of subscripts, my canonization ideas require to explicitly manipulate subscripts. I include the concrete pseudoknot code but explain only the essential

points. A subscript pair (i, j) denotes input sequence positions $inp_{i+1} \ldots inp_j$, [...] denotes lists, and `<-` denotes enumerating a list of alternative values. Comments start with `--` and continue until the end of line.

I begin with the computation of `stacklen` and then introduce a new nonterminal symbol `knot` representing the best folding of a *csr-PK*. Finally, I incorporate the nonterminal in the main folding routine.

Precomputing the maximal helix lengths

The computation of all maximal helix lengths is a simple dynamic programming problem on its own. I solve it, of course, with ADP:

The signature defines only two functions (`sum` and `end`) plus the choice function `h`:

```
type StacklenAlgebra base comp = (
        base -> comp   -> base ->  comp,  -- sum
        base -> Region -> base ->  comp,  -- end
        [comp]                 -> [comp]  -- h
        )
```

Instead of simply maximizing the number of base pairs in a helix, I chose to minimize the free energy in accordance with the overall objective of finding the minimum free energy structure. It is well known that base pair maximization and free energy minimization on the level of overall structure prediction are not equivalent, mainly due to the destabilizing effect of unpaired loop regions which are not taken into account in the base pair model. It is noteworthy, however, that even for the simpler case of finding an optimal helix, they are not equivalent. The difference originates from two values in the current energy model, namely the energy for a $U \bullet G$ pair stacking on top of a $G \bullet U$ pair and $G \bullet U$ stacking on $U \bullet G$. The first case destabilizes the structure by +1.3 kcal/mol, the second case by +0.3 kcal/mol. As a consequence, minimum free energy helices will not begin or end with such stack pairs. However, they may form within a helix, if the surrounding base pairs accommodate for the small destabilizing effect.

The algebra `stacklen_alg` computes a tuple of energy and length, where minimization acts on the first component – the energy.

```
stacklen_alg :: RNAInput -> StacklenAlgebra Int (Int,Int)
stacklen_alg inp = (sum,end,h) where
     sum lb (c,k) rb   = (c + sr_energy inp (lb, rb), k+1)
     end lb _     rb   = (0,1)

     h [] = []
     h xs = [minimum xs]
```

When two equally stable helices are encountered, the smaller one is chosen by the choice function `h`, leaving more bases to the interior of the later pseudoknot. The `end` of a helix has one base pair and thus is of length one, but it has no stacks and thus an energy of zero. With each additional base pair, the corresponding energy is added via `sr_energy`, and the length is incremented by one.

The grammar restricts the search space to helices with at least one base pair and the usual three nucleotide minimal hairpin size. Helices with length zero result in empty lists.

```
stacklen = tabulated(
           (end <<< base -~~ region `with` (minloopsize 3) ~~- base |||
            sum <<< base -~~ stacklen    ~~- base ) `with` basepair ... h)
```

The asymptotic efficiency of this tiny DP problem is $O(n^2)$ in both time and space. The values finally stored in table `stacklen` will now be used for the composition of pseudoknots.

Computing the pseudoknot

I introduce a new nonterminal symbol `knot` which holds the best folding of a subsequence into a pseudoknot.

```
knot (i,j) = [pk' energy a u b v a' w b' | k<-[i+3 .. j-8], l<-[k+4 .. j-4],
```

This line chooses k and l from the interval $[i, j]$ and puts together the results from substructures a, u, b, v, a', w, b' and some local `energy` contribution under the scoring function `pk'`. The boundaries for k and l arise from some simple prerequisites: Each helix must have a minimum length of two bases. Due to stereo-chemical reasons (a more detailed look at this problem will be given in the next section) one base in the front part (u) and two bases in the back part (w) are left explicitly unpaired: These bases should bridge the stacks. This consideration is adopted from *pknotsRE*.

The next definitions implement canonization Rules 1, 2, and 3. They determine the helix lengths, finally computed into the variables h and h'. Some care has to be taken that the pseudoknot helices are not overlapping. If such an overlap occurs, the helices are shortened as necessary. Eventually, if either h or h' is smaller than two, a pseudoknot is not possible at this particular location.

```
-- look up precomputed helix energy and length
(alphanrg, alphalen) <- stacklen (i,k),
(betanrg,  betalen ) <- stacklen (l,j),

-- don't let a-a' and b-b' collide within u
let h = min alphalen (k-i-1),
h >= 2,
```

```
                        -- don't let a-a' and b-b' collide within w
                        let tmph' = min betalen (j-l-2),

                        -- don't let a-a' and b-b' collide within v
                        let h' = min tmph' (l-k-h),
                        h' >= 2,
```

The next lines define the pseudoknot components a through b', plus the local helix energy contribution. For sake of simplicity, I (mis-) use the `region` parser for the helix parts. Their role as helices will be explicitly handled by the algebra function `pk'`. To allow for dangling bases, additional parameters need to be provided with `front, middle`, and `back`.

```
                        a <- region              (i    , i+h    ),
                        u <- front  j            (i+h+1, k      ),
                        b <- region              (k    , k+h'   ),
                        v <- middle (j-h') (i+h) (k+h' , l-h    ),
                        a'<- region              (l-h  , l      ),
                        w <- back   i            (l    , j-h'-2 ),
                        b'<- region              (j-h' , j      ),

                        -- recalculate the energy of shortened helices
                        (acorrectionterm, _) <- stacklen (i+h -1,l-h +1),
                        (bcorrectionterm, _) <- stacklen (k+h'-1,j-h'+1),

                        let energy = alphanrg - acorrectionterm
                                    + betanrg - bcorrectionterm
                    ]
```

If the helices must be chosen shorter than maximal to avoid an overlap, a correction term has to be subtracted. This explains the negative terms in the energy computation.

Left to be defined are the interior structures `front`, `middle`, and `back`.

```
front j =           front'          |||
          frd j <<< front' ~~- base ... h

front'  = ul    <<< singlestrand    |||
                    pk_comps        ... h
```

These cases take care of a potentially dangling base from the b-helix, and if the remaining region is not single stranded, a list of arbitrary substructures is generated from `pk_comps`. `frd` and `ul` are the corresponding functions from the energy model.

For the correct computation of `middle`'s energy contribution, two additional parameters are necessary (see also production `v` above). The first two rules of `middle` deal with the case of coaxially stacked pseudoknot helices, the next four rules implement dangling bases.

```
middle k l = emptymid k l <<< empty                  |||
             midbase  k l <<< base                   |||
             middlro  k l <<< base -~~ base          |||
             middl    k   <<< base -~~ mid           |||
             middr      l <<<          mid ~~- base  |||
             middlr   k l <<< base -~~ mid ~~- base  |||
                                       mid           ... h

mid        = ul           <<< singlestrand           |||
                              pk_comps                ... h
```

The rules for `back` follow the same logic as `front`, except that the dangling base is now adjacent to the a-helix.

```
back i =                    back' |||
         bkd i <<< base -~~ back' ... h

back'  = ul <<< singlestrand      |||
                pk_comps          ... h
```

Returning to the first clause (`knot`), it chooses k, l inside $[i, j]$, computes h and h' using the precomputed maximal helix information, and passes these boundaries to the pseudoknot compartments. Methodically, this is a use of inherited attributes with the underlying tree grammar and appears to be a novel technique in dynamic programming, at least in its grammar oriented tradition [Searls, 1997; Lefebvre, 1996; Rivas and Eddy, 2000; Evers and Giegerich, 2001].

Incorporation into the template grammar

With nested secondary structures, we do not need to be concerned about 3-dimensional foldings whatsoever. Every properly nested secondary structure is stereo-chemically sound.

Unfortunately, the same does not hold for pseudoknotted structures. There are two aspects that need to be taken into account:

First, the pseudoknot stems have to be bridged by loops inside the pseudoknot. This can be guaranteed with relatively low effort by explicitly leaving a few nucleotides in the loops unpaired. Of course, each choice where to leave the bases unpaired may not be perfect for all situations and thus, sacrifices optimality. The decision taken here is to leave one nucleotide explicitly unpaired in compartment u. This nucleotide should bridge the b helix through the major groove, as this is known to be true from H-type pseudoknot crystal structures. The a helix has to be bridged through the minor groove, thus requiring at least two unpaired nucleotides also left explicitly unpaired in the w segment.

Second, the whole pseudoknot has to be bridged if included within a structure spanning the whole pseudoknot. Imagine a pseudoknot with coaxially stacked helices, thus forming

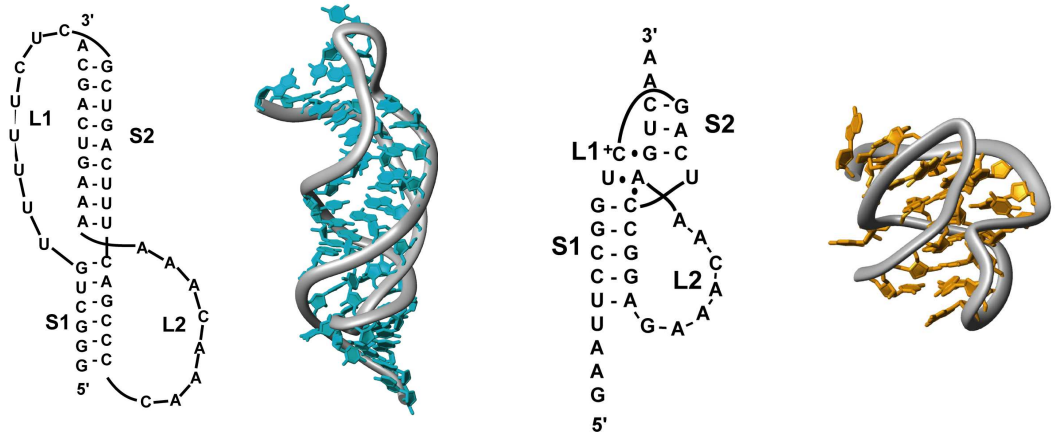


Figure 5.3: Pseudoknot at the core of human telomerase (hTR, left side), a ribonucleoprotein complex, and frameshifting pseudoknot of pea enation mosaic virus (PEMV, right side). Both are classical H-type pseudoknots but have a very different 3-dimensional fold. While the helix ends of hTR lie on opposite sides, a sharp bend in the helix junction brings the end of the PEMV pseudoknot into proximity.

a quasi-continuous helix, as e.g. the pseudoknot of the RNA component of the human telomerase ribonucleoprotein complex (Figure 5.3, left side). In 3D, the pseudoknot's 5' and 3' ends are located on opposite sides of the molecule. Hence, structures such as `((((([[[.{{{{]]]..}}}}}))))` are sterically infeasible. Clearly, a short single stranded stretch of nucleotides has to bridge the pseudoknot, but the length of this stretch strongly depends on the actual 3-dimensional structure. On the other hand, the pseudoknot of pea enation mosaic virus (Figure 5.3, right side) has a sharp bend at its helix junction which brings the pseudoknots ends almost in proximity. Here, fewer bases are needed for bridging the ends of the pseudoknot.

This is where we leave the field of secondary structure prediction. To solve those issues, I would need a 3-dimensional pseudoknot fold. Having the 3D-coordinates of the ends of the pseudoknot, I could calculate the bases necessary for bridging the pseudoknot. Yet, 3D-folding of a 2D-structure itself is a hard problem and clearly not within the scope of this thesis.

The above considerations led to the following changes in the template grammar of Section 3.2.3:

Pseudoknots are non-closed components but can otherwise occur wherever the grammar generates a `dangle`. Therefore, a new nonterminal `dangle'` is introduced which can

then either become a `closed` component via `dangle`'s productions or become a pseudoknot.

```
dangle'    = dangle        ||| dangleknot      ... h
dangle     = edl  <<< base -~~ closed ~~. loc  |||
             edr  <<<  loc .~~ closed ~~- base |||
             edlr <<< base -~~ closed ~~- base |||
             is   <<<  loc .~~ closed ~~. loc  ... h

dangleknot = kndr <<<          knot   ~~- base |||
             kndl <<< base -~~ knot            |||
             kndlr<<< base -~~ knot   ~~- base |||
             pk   <<<          knot            ... h
```

The four productions of `dangleknot` implement dangling of bases outside of the pseudoknot.

Since I chose to model pseudoknots as non-closed components but still want to be able to model structures like `((((..[[[..{{{{]]].}}}}.....))))`, I introduce two special cases within `ml_comps1` which allow for such structures, as long as there are at least three unpaired nucleotides at either the pseudoknot's 5' or 3' end. These unpaired bases should act as safeguards against at least some sterically infeasible structures.

```
ml_comps1 = tabulated (
            sadd  <<< base     -~~ ml_comps1 |||
            cadd  <<< mldangle ~~~ ml_comps  |||
            sadd  <<< region `with` (minloopsize 3) ~~~ dangleknot |||
            addss <<< dangleknot ~~~ region `with` (minloopsize 3) ... h)
```

The last nonterminal I need to define is `pk_comps` which introduces recursive structures within the pseudoknot loops. `pk_comps` is basically a copy of `ml_comps` with the exception that unpaired nucleotides generated from these rules incur a small penalty (0.1 kcal/mol).

```
pk_comps  = tabulated (
            cadd <<< singlestrand -~~ pk_comps |||     -- one base at a time
            cadd <<< mldangle     ~~~ pk_comps |||
            cadd <<< mldangle     ~~~ (ul <<< emptystrand) ... h)
```

The complete grammar is displayed graphically in Figure 5.4 and is also given in APD notation in the appendix.

5.2.3 Better than optimal ...

There are many reasons why "the" MFE structure may only be part of what we want to know about a molecule's foldings. Often, the native structure of an RNA is not predicted as the MFE structure. This may stem from uncertainties in the energy parameters, or the molecule may not reach its MFE structure, either due to interactions with another

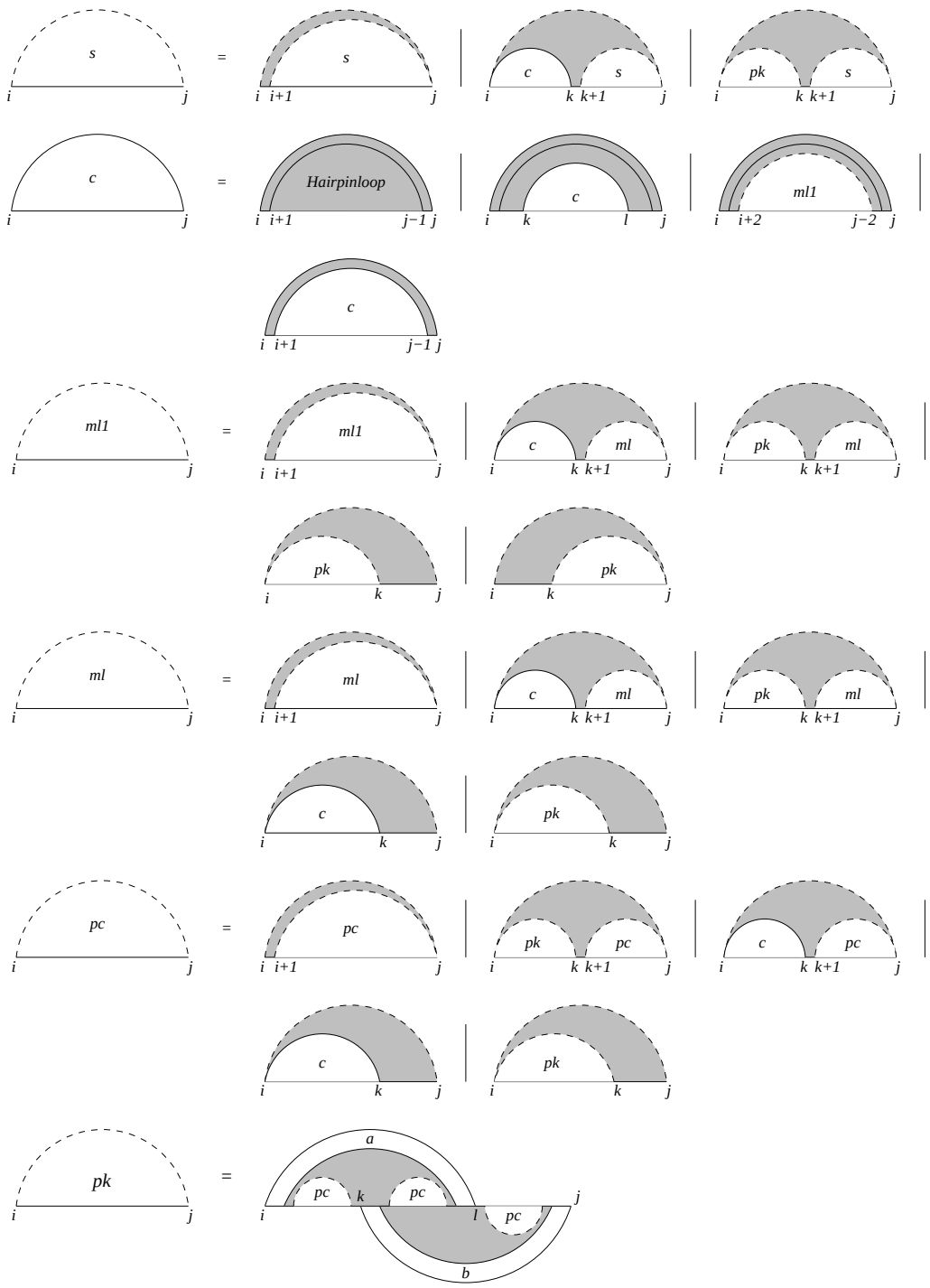


Figure 5.4: Graphical representation of the grammar underlying *pknotsRG*. The solid horizontal line represents the RNA sequence; solid arcs stand for base pairs. White regions are structure parts yet to be filled in subsequent recursion steps. Note that this graphical view does not account for dangling bases.

molecule or by ending in a kinetic trap. To deal with the problem of the optimal (knotted) structure being non-canonical and its canonical representative being dominated by an unrelated structure, I provide two means: First of all, the algorithm is non-ambiguous which is the prerequisite for a non-redundant enumeration of near-optimal structures [Giegerich, 2000]. We can ask the program to report all structures within a given energy increment above the MFE. This is done in a way similar to *RNAsubopt* [Wuchty *et al.*, 1999] from the Vienna RNA package for pseudoknot-free structures. Second, I shall provide three variants of the program:

pknotsRG-mfe	computes the **mfe** structure (or the k best), pseudoknotted or not.
pknotsRG-enf	picks out the energetically best structure from the folding space that contains at least one pseudoknot (**enforced pseudoknot mode**).
pknotsRG-loc	computes the energetically best pseudoknot that can be formed **locally**, i. e. somewhere in the sequence. "Best" is defined here as minimal free energy per base to avoid a built-in bias towards large pseudoknots.

Enforced pseudoknot mode

The enforced mode can be implemented very elegantly as an instance of so called *classified DP*. I divide the search space into two *classes*, properly nested structures and pseudoknot containing structures. Now, I solve the optimization problem conceptually for each class separately and get two optimal results, one for each class. Technically, I need a *classification algebra* that maps each (partial) candidate of the search space onto one class. Per default, all classification algebras have the identity function as the choice function.

Since there are only two classes, I can use a boolean variable as the algebra's result type, where a true value represents a structure containing a pseudoknot. Then, there are basically only three interesting algebra functions, namely `nil`, `hl`, and `pk'`:

```
hl  llb lb r rb rrb      = False

nil _                    = False

pk'  x a u b v a' w b' y = True
```

The empty folding (`nil`) and hairpins (evaluated by `hl`) are the end of the recursion in my DP scheme and certainly do not contain pseudoknots. Regardless of the inner components, function `pk'` always applies to pseudoknots.

All other algebra functions either simply pass on the result of their recursive component (e.g. `sr`) or combine the results of their sub-parses by a logical OR-operation

Program variant	Nonterminals	Productions	Tables
pknotsRG-mfe	24	63	7
pknotsRG-loc	26	67	7
pknotsRG-enf	33	84	11

Table 5.2: The grammar sizes of three variants of *pknotsRG*

(denoted by `||`):

```
sr  lb e rb = e

cons  c1 c  = c1 || c
cadd  x  e  = x  || e
```

These functions are implemented in an algebra called `knot_or_not`. Altogether, the star product of three algebras `knot_or_not *** mfe *** prettyprint` computes two optimal foldings, one for each class, and their respective energy and dot-bracket representation.

Unfortunately, the ADP compiler (ADPC) [Steffen, 2006] is, at the time of writing this thesis, not able to compile the general star product of three or more algebras to C code[2], which is why the above implementation only works in the Haskell embedding. For the fast C implementation, I either have to manually incorporate the classifying procedure into the C code, or I must modify the grammar. The second one is the choice taken here. By grammar duplication, I obtain for each nonterminal two almost identical variants. The first nonterminal stores the best folding not containing a pseudoknot according to its productions. The second one, with almost identical rules, is restricted to contain at least one pseudoknot. By carefully modifying the productions, I keep track of whether there has been a pseudoknot in a subcomponent or not. The size of the resulting grammar is given in Table 5.2.

Local pseudoknot mode

The best local pseudoknot motif is included by adding four cases and making `bestPK` the new axiom:

```
bestPK    = unscale   <<< bestleft
bestleft  = skipleft  <<< base      -~~ bestleft ||| bestright ... h_r
bestright = skipright <<< bestright ~~- base     ||| knot_rel  ... h_r
knotrel   = scale     <<< knot
```

Algebra functions `scale` and `unscale` deal with the length normalization of the pseudoknot energy. These clauses have time complexity $O(n^2)$ and preserve the non-ambiguity

[2]Recently, a PhD student in our group started to work on including this feature into the compiler.

of the algorithm. If desired, an enumeration of near-optimal "local" pseudoknots is also feasible.

The relative effort of implementing the three variants of *pknotsRG* can be judged from the sizes of the tree grammars required which are summarized in Table 5.2.

5.3 Evaluation

First, I compare the predictive accuracy achieved by my approach and other DP based algorithms. I restrict the evaluation to programs implementing the full energy model, knowing that simpler models allow for faster programs but at the cost of less accurate results. I finish with a benchmark of the asymptotic efficiency.

5.3.1 Predictive accuracy

I have already evaluated the class csr-PK against known pseudoknots, and I know that my algorithm correctly implements this class in its search space. What is really tested in the following is the adequacy of the current thermodynamic model (which my algorithm shares with *RNAfold* for nested structures and, in an older version, with *pknotsRE*), and the results in this section may improve if this model is further improved in the future.

I applied my algorithm on the set of sequences listed in Table A.1, including 251 sequences from PseudoBase [van Batenburg *et al.*, 2001]. Although there is some redundancy on the sequence level, there is a good reason why I decided to use all available sequences for testing: Even near identical sequences can have different MFE structures, or a small change may prevent successful pseudoknot prediction. In contrast to [Deogun *et al.*, 2004], I did not restrict the evaluation to the class of pseudoknots recognized by my program. It is also instructive to retain the difficult cases and see whether the predictions catch at least some aspect of a more general pseudoknot. For example, for the sequence of hepatitis delta virus (HDV), which natively folds into a planar pseudoknot, my algorithm predicts all helices except for the very short helix 5 (see Figure 5.5).

The pseudoknots from PseudoBase usually do not have large flanking regions. In fact, the database maintainers encourage submitters to cut out the pseudoknot from the original sequence, flanked by 5 to 10 nucleotides. Also, sequences containing multiple pseudoknots should be submitted separately. In order to test the algorithm on sequences with more flanking context and, possibly, multiple pseudoknots, I included several other test candidates. Namely, the complete *E.coli transfer-messenger RNA* (tmRNA), the 3' UTR of turnip yellow mosaic virus (TYMV, [Deiman *et al.*, 1997]), tobacco mosaic virus (TMV, [van Belkum *et al.*, 1985]), odontoglossum ringspot virus (ORSV, [Gultyaev

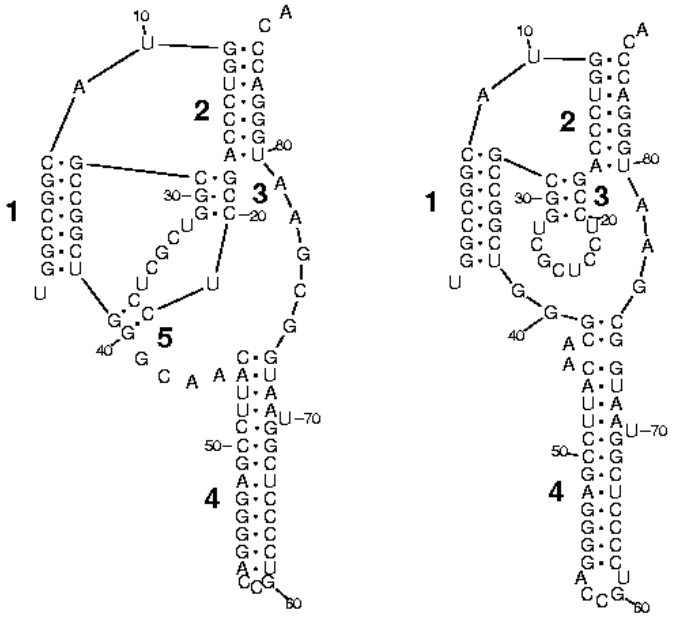


Figure 5.5: The pseudoknot of hepatitis delta virus: On the left side the structure proposed by [Ferré-D'Amaré *et al.*, 1998], on the right side the MFE-structure predicted by *pknotsRG*. The algorithm misses only the short helix 5 and adds three base pairs to helix 4. Image created with *RnaViz2* [Rijk *et al.*, 2003].

et al., 1994]), and satellite tobacco necrosis virus (STNV, [Danthinne *et al.*, 1991]). Their pseudoknots are partially included in PseudoBase without flanking regions, but in general, the prediction is much more difficult and error-prone, when larger sequences are under evaluation.

I compared the results to the output of *RNAfold*, as a representative for RNA folding tools without pseudoknot folding capability, and to *pknotsRE* and *NUPACK* where computationally feasible. Implementations for the algorithms of the A&U and L&P class are currently either not available or not existent, respectively. For each predicted structure, I counted the number of correctly and falsely predicted base pairs (TP and FP). Let BP be the number of base pairs in the reference structure from the database or literature. False negatives (FN) can be derived from: $FN = BP - TP$. Finally, I measured the sensitivity, defined as (TP/BP), and selectivity (TP/TP+FP).

In Table 5.3, I list the prediction accuracy for the sequence set. For all sequences except the tmRNA, I enhance the prediction accuracy with respect to *RNAfold*. Both, sensitivity and selectivity, are increased. Compared to *pknotsRE*, my results are slightly

	RNAfold			*pknotsRE*			*NUPACK*			*pknotsRG-mfe*		
Sequence ID	Sens	Sel	K	Sens	Sel	K	Sens	Sel	K	Sens	Sel	K
PseudoBase	55.7	63.7	-	73.9	71.0	-	77.7	72.9	-	78.9	76.7	-
HIVRT	42.9	73.3	1/2	100	100	2/2	97.4	100	2/2	100	100	2/2
tRNAs	59.4	48.5	-	78.1	64.3	-	59.8	46.1	-	64.7	51.8	-
HDV	37.5	42.9	2/4	43.8	46.7	2/4	59.4	61.3	1/4	90.6	93.5	3/4
TYMV	70.8	73.9	1/2	100	96.0	2/2	87.5	87.5	2/2	91.7	88.0	2/2
TMV-up	52.0	61.9	3/6	52.0	59.1	3/6	52.0	61.9	3/6	88.0	88.0	6/6
TMV-down	67.7	74.2	2/4	94.1	94.1	4/4	52.9	54.5	2/4	100	100	4/4
STNV	37.7	32.5	2/8	*	*	*	*	*	*	60.9	54.6	4/8
tmRNA	51.0	48.6	3/6	*	*	*	*	*	*	51.0	49.1	3/6
ORSV	41.2	46.7	7/22	*	*	*	*	*	*	75.0	73.4	19/22

Table 5.3: Evaluation of predictive accuracy.
Sens = sensitivity, Sel = selectivity, K = (number of correct predicted pseudoknot helices)/ (expected number of pseudoknot helices). A helix is accepted as correct, if it overlaps with at least one base pair of the reference structure. For HIVRT, PseudoBase, and tRNA, the average over all sequences is taken. * marks cases where *pknotsRE* and *NUPACK* were unable to terminate.

better, probably because we are using the newer and subtler energy model. Interestingly, *pknotsRG* also performs better than *NUPACK*, although both implement a very similar energy model. Perhaps, *NUPACK*'s larger search space is the source for some false predictions. However, there may be some other flaws in its implementation: I noticed that *NUPACK* predicts a nested structure with -18.4 kcal/mol for the sequence of TMV-up, and *pknotsRG* finds the correct pseudoknot with energy -25.3 kcal/mol. It is unclear to me, why *NUPACK* does not predict a similar pseudoknot given the low energy of *pknotsRG*'s prediction.

I also folded eleven randomly selected human tRNAs, previously used in other studies [Mathews and Turner, 2002] (third line in Table 5.3), and found false positive pseudoknots in three of them. Interestingly, the overall prediction accuracy is, nevertheless, higher than that of *RNAfold*. Note that *pknotsRE* outperforms all other programs on this sequence set, probably because it also implements coaxial stacking of multiloop helices. tRNAs are known to form two coaxial stacks which results in their typical 3-dimensional fold.

The evaluation is in good accordance with the evaluation in [Condon *et al.*, 2004]. There, *pknotsRG* showed a leading performance with 0.76 sensitivity and 0.77 selectivity (called specificity there) on a short sequence set. Similar values were measured for *HotKnots*, introduced in that particular study. *pknotsRE* and *NUPACK* both performed

slightly worse than *pknotsRG*. On the longer sequence set (> 200 nucleotides), the performance dropped for all evaluated programs. Sensitivities fall to around 50% and selectivities even below 40%. This shows the general problem of folding long sequences via free energy minimization. I comment on this phenomenon in Section 5.5.3.

Since I use the same energy model as *RNAfold*, predictions of *RNAfold* and *pknotsRG* for unknotted structures are identical. Of course, if there is more than one optimal structure, each of the optimal alternatives may be reported, and thus, the same folding cannot be guaranteed.

5.3.2 Computational performance

I benchmarked the performance of the programs used in the previous evaluation by 20 iterations with random sequences of length 20 to 800 bases where applicable. Time and memory consumption were estimated with the `memtime` utility on a 2.8 GHz Intel Xeon machine running Sun Solaris 10 equipped with 2 GB main memory. Benchmarks of *NUPACK* had to be done on a 4 GB Linux machine due to compiling issues. This machine is approximately 33% faster, which should be kept in mind when comparing *NUPACK* runtimes.

The results of the benchmark are displayed in Figure 5.6. The limiting factor for *pknotsRE* is clearly the demanding $O(n^6)$ runtime. A sequence of length 100 is folded within ∼5000 seconds; a sequence twice as long would require three to four days. The lower asymptotic runtime of *NUPACK* allows for much faster folding than *pknotsRE*. The limiting factor now becomes the steep $O(n^4)$ memory requirement. With almost 3 GB for folding 150 nucleotides, we are about to hit the 32-bit boundary. Interestingly, *NUPACK* requires approximately the six-fold amount of *pknotsRE*'s memory despite both being in the same asymptotic class. This can be explained by the fact that *NUPACK* uses twice as many 4-dimensional tables and is implemented in an object-oriented way, which may require a little overhead.

Clearly, *pknotsRG* is able to fold sequences that are longer than *pknotsRE*'s and *NUPACK*'s limits. Short sequences up to 200 nucleotides are folded within a second. Long sequences (800 bp) take about six minutes. If we restrict the maximal pseudoknot size to a reasonable constant, say 150 nucleotides, we can further increase the runtime. The algorithm now runs in $O(cn^3)$ with a rather large constant c. This enables us to fold sequences of length 1000 in 90 seconds. Being in the same asymptotic space complexity class as usual RNA folding routines (e.g *RNAfold*), the memory problems are basically removed. 30 MB for a folding of 800 nucleotides poses no problem to modern hardware.

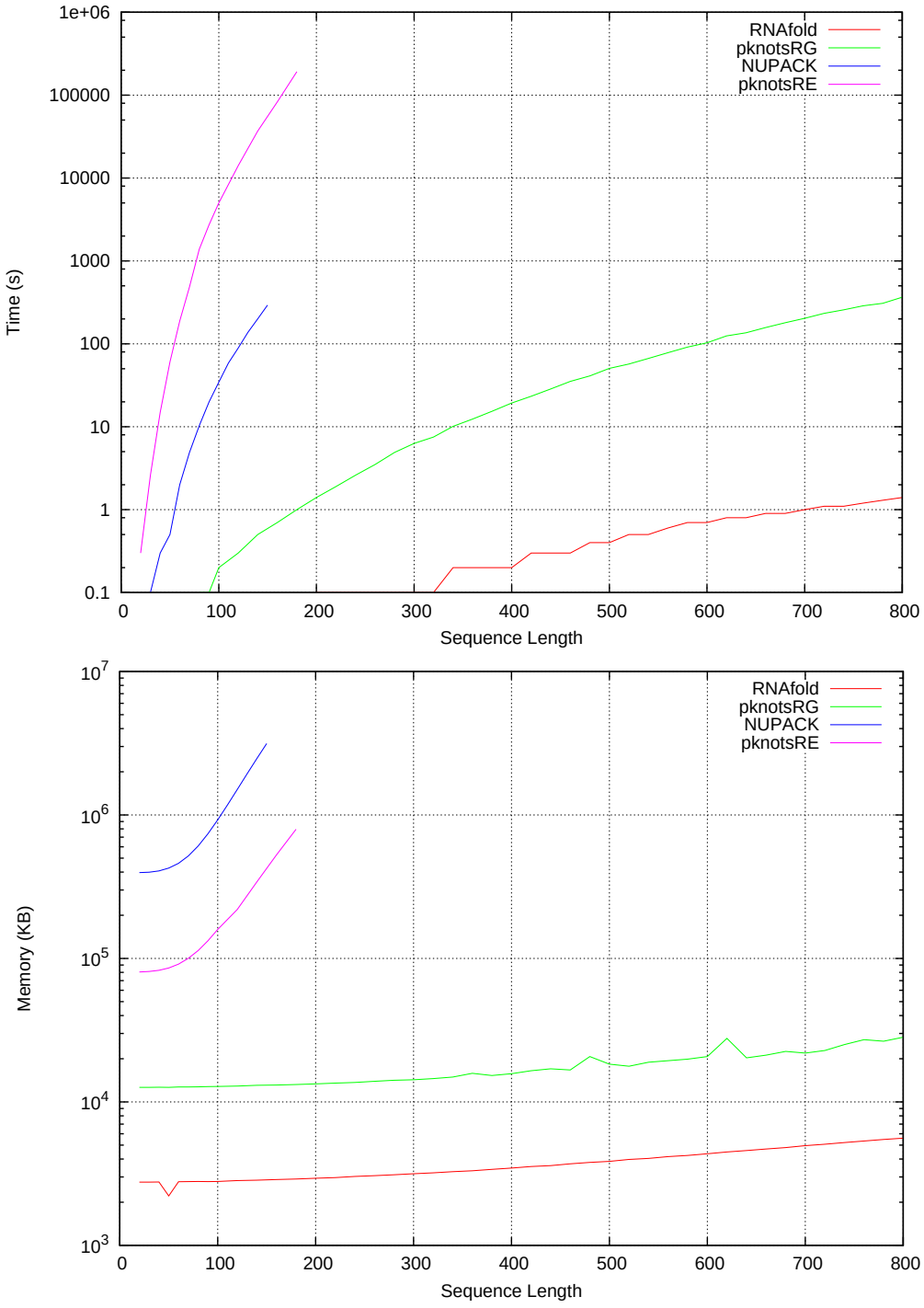


Figure 5.6: Benchmark of *pknotsRG* and other programs. *pknotsRG* is superior to *pknotsRE* and *NUPACK*, both in terms of time and memory. For comparison, also the values for *RNAfold* are displayed. Note that the time and memory axes are drawn logarithmically.

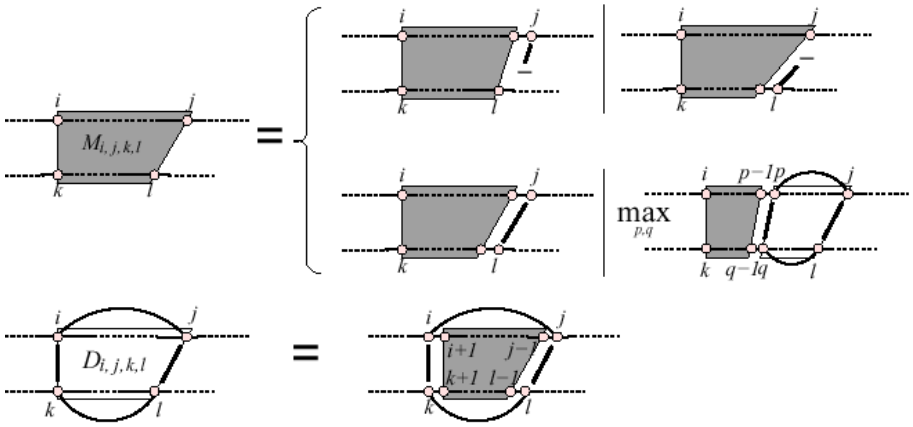


Figure 5.7: Recurrences for the Sankoff-style alignment of base pair probability matrices.

For a fair comparison, the reader should keep in mind that the extra time spent by *pknotsRE* and *NUPACK* is not strictly wasted: It is spent on assuring that the optimal folding of the input RNA sequence does not contain a more complex pseudoknot of lower free energy than the reported structure. *pknotsRG* does not consider such structures and hence, cannot make this assertion.

5.4 Sparse DP

One way to speed up Dynamic Programming algorithms is to make use of the implicit sparseness of the search space. Eppstein and coworkers [Eppstein *et al.*, 1990, 1992a,b] analyzed several sequence analysis algorithms and detected situations for which intermediate solutions can be safely ruled out without evaluating them. In the following, I describe how to exclude certain pseudoknot candidates from the folding space of *pknotsRG* and thus, save a significant amount of computation time.

The idea roughly follows the technique of the algorithm implemented in the tool *LocARNA* [Will *et al.*, 2007] which improves the Sankoff style RNA secondary structure and alignment algorithm *PMcomp* [Hofacker *et al.*, 2004]. I do not explain the algorithm in detail here, but only the parts necessary to understand the modifications to the original *pknotsRG* algorithm.

The recurrences used in *PMcomp* are shown in Figure 5.7. The matrix D stores the best alignment of subsequences $s_i^1 ... s_j^1$ and $s_k^2 ... s_l^2$, given that s_i^1 aligns to s_k^2 and s_j^1 to s_l^2 and that (s_i^1, s_j^1) and (s_k^2, s_l^2) each form a base pair. The best result for unconstrained folding and alignment is stored in matrix M. It can easily be seen that the most expensive

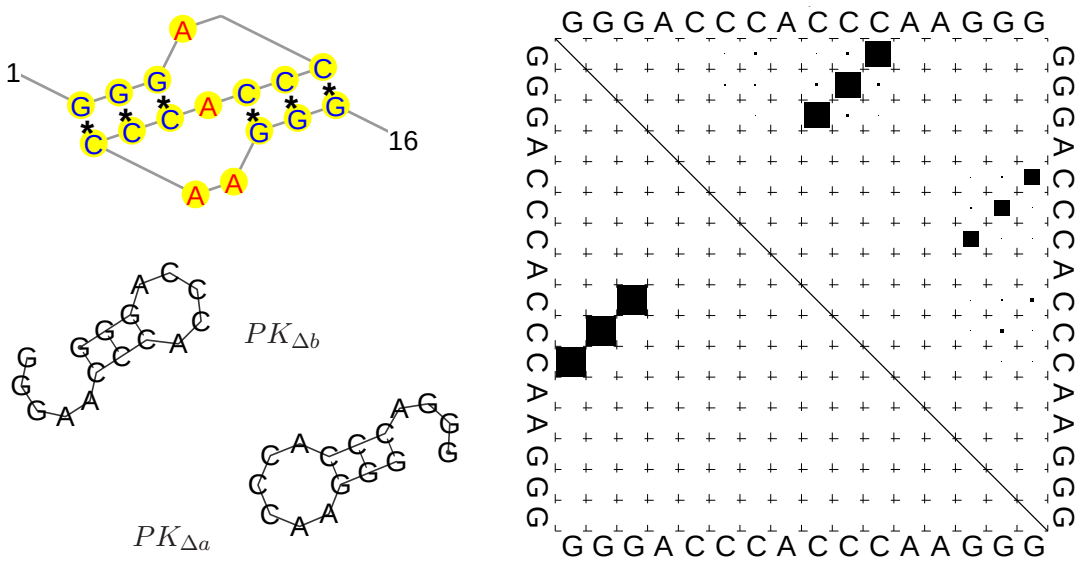


Figure 5.8: A pseudoknot and its (nested) dot plot. The size of the dots is proportional to the respective base pair probability.

part of the recurrences lies within the last alternative for the computation of M. Two base pairs, one for each sequence, are added to the consensus structure, thus requiring overall $O(n^6)$ time.

Now, the trick of *LocARNA* is that not all possible base pairs are examined for inclusion in the consensus structure but, only those having a base pair probability p larger than a threshold value p^*. In accordance with [Will *et al.*, 2007], I call them *significant* base pairs. Of course, given a fixed p^*, for each base, there are at most $1/p^* \in O(1)$ significant base pairs and thus, $n * p^* = m \in O(n)$ in a complete probability matrix. It directly follows that by restricting the most expensive loop to the significant base pairs, computation time can be reduced to $O(n^2 * m^2)$.

5.4.1 Sparse *pknotsRG*

Now, I demonstrate a similar idea for reducing the number of evaluated pseudoknots. Although the partition function algorithm computing the base pair probabilities (taken from the Vienna RNA library) evaluates only nested structures, the helices involved in a pseudoknot can often be seen in the dot plot. An artificial toy example is shown in Figure 5.8. Instead of the pseudoknot, two alternative structures can be folded with a reasonable energy by deleting one of the helices each time. For later use, I call the structures $PK_{\Delta a}$ and $PK_{\Delta b}$. One can see both helices in the dot plot, where base pairs from $PK_{\Delta b}$ have probabilities around 69% and base pairs from $PK_{\Delta a}$ around 25%. With larger sequences,

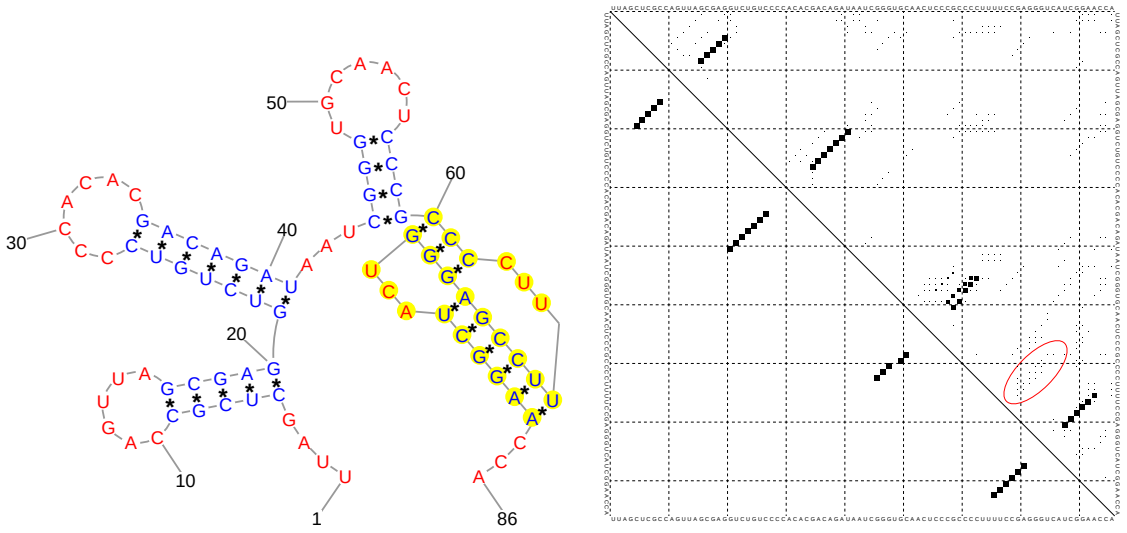


Figure 5.9: The pseudoknot of turnip yellow mosaic virus and its dot plot. The base pairs of the smaller pseudoknot helix are marked with a red ellipse.

the pseudoknot helices are also visible in the dot plot, however, having probabilities smaller by two orders of magnitude as shown in Figure 5.9. The shorter pseudoknot helix has base pair probabilities of around 0.001 (i.e. 0.1%).

In the optimal case, the pseudoknot-free MFE structure is just lacking one pseudoknot helix but otherwise shares all base pairs as seen as in the toy example ($PK_{\Delta b}$ in Figure 5.8). Then, we can expect the probabilities of the shared helix to be reasonably high. On the other hand, if the pseudoknot-free MFE structure folds an alternative structure not sharing any base pairs with the pseudoknot, probabilities tend to be lower. The same holds for the second pseudoknot helix which is always contained only in suboptimal structures (except for the rare case that the two alternative structures are co-optimals).

I now adapt the *pknotsRG* algorithm in order to use the information from the dot plots. Remember that the $O(n^4)$ time requirement stems from two variable boundaries k, l, which move in between the left and right boundaries i and j.

```
knot (i,j) = tabulated [ ... | k <- [i..j], l <- [k..j],
                                          ...
                                                        ]
```

Notice that in all evaluated pseudoknots, k is paired with j and base l with i. In order to incorporate the base pair probabilities into *pknotsRG*, I define a new array of lists `jpair` that stores, for each j, a list of all significant base pairs (k, j). The array can be computed before the main folding routine:

```
jpair:: Array Int [Int]
jpair = accumArray (flip (:)) [] (1,n) [(j,k) | j <- [1..n],
                                                k <- [1..j-5],
                                                bp_prob k j >= p* ]
```

The ks are stored in reverse order, such that the loop is terminated as soon as k passes beyond the left pseudoknot border i:

```
knot (i,j) = tabulated [ ... | k <- takeWhile (>i) (jpair!j), l <-[k..j],
                                                    ...
                                                                ]
```

An analogous array of lists `ipair` stores an ascending list of significant base pairs for each i.

```
ipair:: Array Int [Int]
ipair = accumArray (flip(:)) [] (1,n) [(i,l) | i <- [1..n],
                                               l <- reverse [i+5..n],
                                               bp_prob i l >= p* ]
```

Both, `ipair` and `jpair` store lists of size of at most $1/p^*$ and thus, require $O(n * 1/p^*)$ space. Including `ipair` into the pseudoknot loop yields:

```
knot (i,j) = tabulated [ ... | k <- takeWhile (>i)                  (jpair!j),
                               l <- takeWhile (<j) (dropWhile (<k) (ipair!i)),
                                                ...                          ]
```

Here, `dropWhile` and `takeWhile` assure that only values from the interval $[k+1 \ldots j-1]$ will be evaluated.

Following the efficiency analysis of *LocARNA*, the loop is executed in the order of $O(n^2 * (1/p^*)^2)$. Assuming a constant p^* thus reduces the overall runtime of the *pknotsRG* algorithm to $O(n^3)$, since the pseudoknot code is embedded in an $O(n^3)$ context for general folding.

The actual runtime, however, strongly depends on the choice of p^*. Recall that for the pseudoknot of TYMV, we would need a threshold of as low as 0.001. In order to estimate the effect of different cut-off values, random sequences of varying sizes, whose MFE prediction contains a pseudoknot, were folded with the new method and compared to the original algorithm (Figure 5.10). Observe that with high p^*-values (say 0.01 to 0.0001) more than 50% of the foldings differ on average. A value as low as $1 \cdot 10^{-6}$ suffices for more than 80% of the random sequences and $1 \cdot 10^{-9}$ almost always (99%).

By visual inspection, I identified three main reasons for mispredictions at low p^* values. Namely, pseudoknots with a very short helix, pseudoknots stretching over a long subsequence, and structures containing multiple pseudoknots. In all three cases, chances

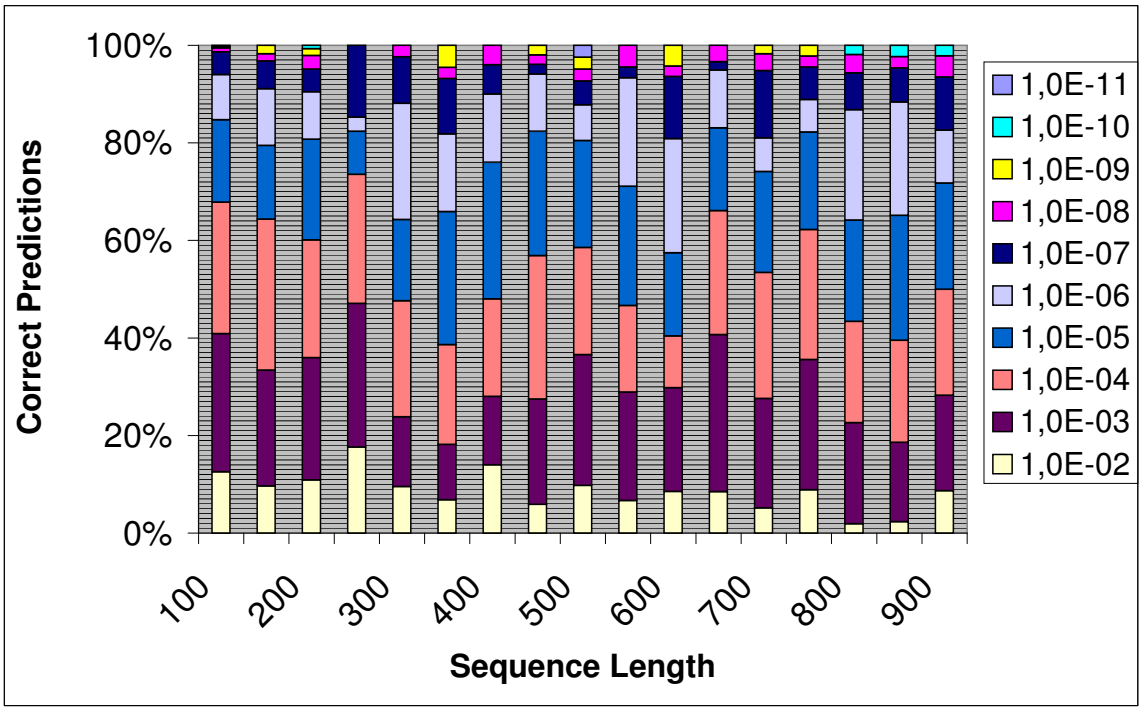


Figure 5.10: Evaluation of a suitable p^* parameter. For each sequence length, I generated 100 random sequences and recorded which p^* achieves the same folding as the original *pknotsRG*. E.g. for sequence length 100 there are 40% identical predictions with p^* set to $1 \cdot 10^{-3}$ and 85% at $p^* = 1 \cdot 10^{-5}$. In general, the effect of p^* is independent of the sequence length.

are high that the alternative pseudoknot free structures considered by the Vienna library partition function have significantly lower energies than the (pseudoknot) MFE prediction. Consequently, the base pair probabilities of the pseudoknot helices will be extremely low. When talking about asymptotic considerations, one has to bear in mind that constant factors, such as $1/p^*$ in our case, can only be omitted for input sequences larger than a certain length N_0. Below that threshold constant factors may still have a dominant effect. Unfortunately, with *pknotsRG* we usually fold sequences shorter than 1000 nucleotides. Hence, $1/p^*$ cannot be seen as a negligible constant. As the number of significant base pairs per base cannot exceed n, the algorithm is effectively in the same efficiency class as before.

However, it is known that a dot plot is generally only sparsely populated, and we can expect a far lower number of significant base pairs than $1/p^*$ per base. In fact, one can see from Figure 5.11 that the number of significant base pairs, averaged over all bases,

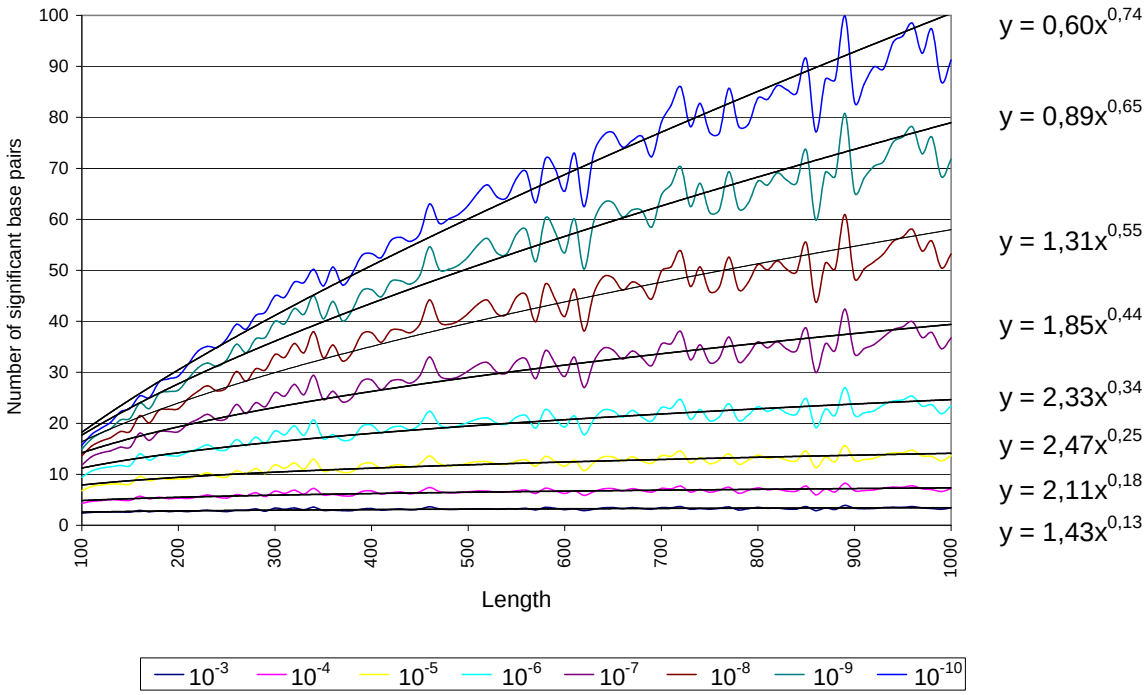


Figure 5.11: The average number of significant base pairs in one column or row of the dot plot grows sublinear. The regression parameters are noted on the right side.

does not grow linearly with n. E.g., with $p^* = 1 \cdot 10^{-6}$ it grows proportional to $n^{0.34}$, and even with $p^* = 1 \cdot 10^{-9}$, sufficient for 99% accurate prediction, growth is within $O(n^{0.65})$. This, of course, means that although an improvement in the worst case runtime is not guaranteed, sparse *pknotsRG* may still be faster in the average case. And indeed, one can detect a remarkable speed up in the benchmark in Figure 5.12. A 400 nucleotide sequence folds within half the time for a low cut-off ($p^* = 1 \cdot 10^{-9}$). With long sequences and a slightly higher cut-off ($p^* = 1 \cdot 10^{-6}$), folding is faster by a factor of 10.

Note that the estimated polynomial runtime from regression analysis is even lower than $O(n^3)$, although the pseudoknots are still computed in an overall cubic context. This is again due to the effect of constant factors: computing the pseudoknot loop is much more expensive than all the other loops together and thus dominates.

Summing up, one can say that the use of base pair probabilities considerably speeds up pseudoknot computation with only a little sacrifice in accuracy.

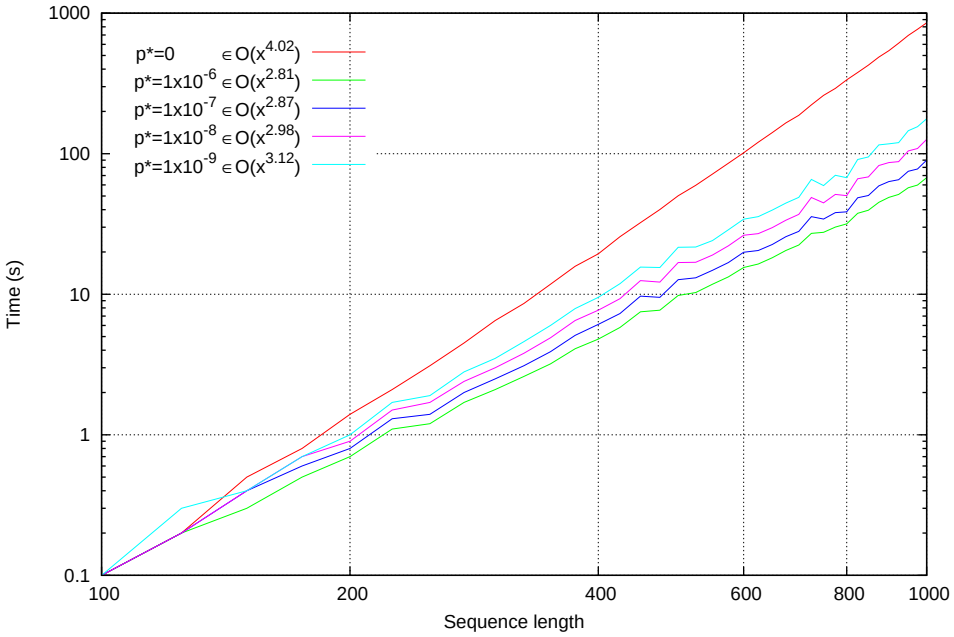


Figure 5.12: Benchmark of the sparse *pknotsRG* variant. The red line ($p^* = 0$) shows the runtime of the original *pknotsRG* on random data. The observed asymptotic efficiency is given in the legend. Note that the plot is drawn logarithmically on both axes in order to clearly show the asymptotic behavior.

5.5 Discussion

In the following, I discuss extensions of the implemented model and their expected computational cost.

5.5.1 Bulges, triple crossing and kissing hairpins

Canonization Rule 1 can be relaxed to allow for bulges inside the helices forming a pseudoknot. As long as their number (and hence, the length difference of the two arms of a helix,) is bounded by a constant, asymptotic efficiency is not affected. As proof-of-concept, I implemented a variant of *pknotsRG* which allows for small, 1-nt bulges. The changes only slightly affect the algorithm's runtime but require a few extra lines of code which take care of whether Rule 3 truncates one of the maximal helices before, within, or after the bulge. Of course, the same observation holds for small internal loops.

Two examples of non-simple pseudoknots are shown in Figure 5.13. I can incorporate them into my algorithm by adding the definitions

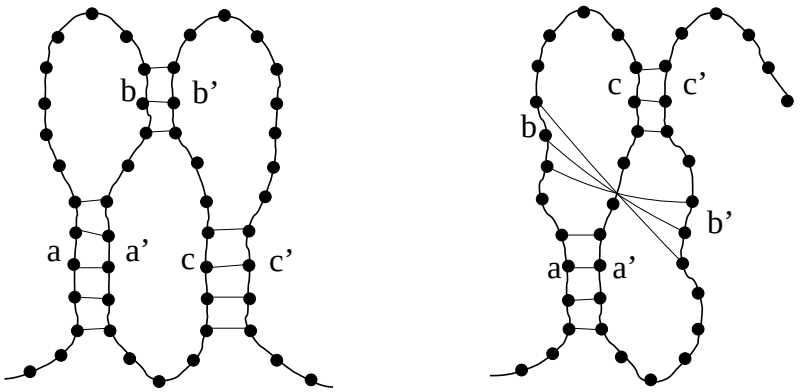


Figure 5.13: Two examples of more complex pseudoknots with three helices: Kissing hairpins (left) and triple helix interaction (right).

```
kiss   = kss <<< a~~~u~~~b~~~v~~~a'~~~w~~~c~~~x~~~b'~~~y~~~c'
triple = trp <<< a~~~u~~~b~~~v~~~c~~~w~~~a'~~~x~~~b'~~~y~~~c'
```

Canonization can be applied as before, with Rule 3 becoming more sophisticated for the triple interaction case. This would yield an algorithm of runtime $O(n^6)$ bringing runtime back to the efficiency class of the Rivas/Eddy algorithm. Note though that the space requirement remains $O(n^2)$. This is due to the fact that now three interacting helices are considered but not arbitrary chains.

5.5.2 Canonization - revisited

In Section 5.1.3, I argue that the canonization is well justified. In this section, I empirically estimate the effect of canonization Rules 2 and 3 in order to strengthen my decision. From a theoretical point of view, both rules reduce the asymptotic worst-case runtime by pruning the search space. Rule 2 lowers the runtime from $O(n^6)$ to $O(n^4)$ by reducing the number of pseudoknots which must be evaluated. For each pseudoknot boundary set (i, j, k, l), only one pseudoknot – the one with maximal helix length – is considered instead of $|a - a'| * |b - b'|$ pseudoknots. A similar consideration holds for Rule 3: Without an ad-hoc decision on where to split overlapping helices, the algorithm would have to evaluate all possible splits, bringing runtime back to $O(n^5)$.

However, the *expected* increase in runtime depends on the lengths of the maximal helices and the amount of overlap, respectively. In the average case, I expect both to be rather small and independent of the sequence length n. Thus, I can hope for faster programs in practice.

To test these hypotheses, I implemented two variants of *pknotsRG*. *pknotsRG-sr* ig-

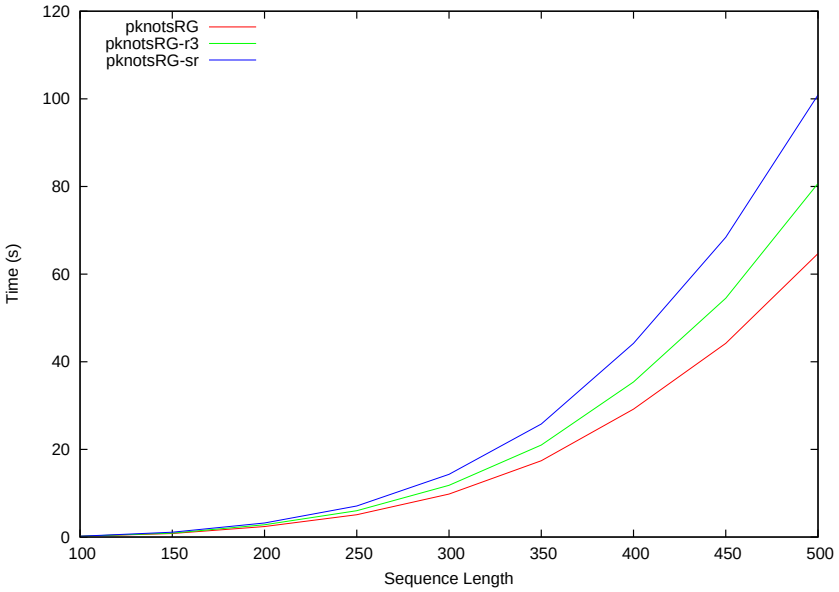


Figure 5.14: The effect of canonization Rules 2 and 3: Practical runtimes increase by 50%, if *pknotsRG* evaluates all simple recursive pseudoknots. Relaxing Rule 3 results in a 25% longer computation time.

nores Rule 2 (and implicitly Rule 3) and thus evaluates all simple unbulged recursive pseudoknots. *pknotsRG-r3* alters Rule 3 and loops over all possible configurations resolving the overlap. Note that for both variants, Rule 1 still remains in effect, which means no bulges are allowed in pseudoknot stems.

I generated 1000 random sequences, with lengths ranging from 50 to 100 nucleotides, containing a pseudoknot in the MFE folding computed by *pknotsRG-sr* (the most general folding program). For only 85 sequences, *pknotsRG* predicts another structure with a higher energy. Moreover, 24 of the 85 structures are the canonical representatives of the *pknotsRG-sr* MFE structures. Predictions of *pknotsRG* and *pknotsRG-r3* differ in 53 cases, with 12 pseudoknot structures being the canonical ones. These numbers confirm my previous rationale: the canonical pseudoknot is in most cases (over 90 %) also the optimal one. Furthermore, there are only few cases (around 6%) where my canonization prevents a particular (non-canonical) pseudoknot to be detected, because another structure has a lower energy and is reported as MFE structure.

What remains to be evaluated is the effect on the real computation time. The benchmark in Figure 5.14 shows that the observed runtime of *pknotsRG-sr* increases by a constant factor of about 1.5. The other variant, *pknotsRG-r3*, brings a 25 % increase in runtime. This suggests that Rules 2 and 3 may be relaxed at a noticeable cost but with

little improvement in search space coverage. Hence, I will offer the relaxed variants as an optional mode in the next release of the *pknotsRG* software package.

5.5.3 Folding long sequences

RNA folding *in vivo* as *in vitro* must be understood as a hierarchical process, where small structures in close vicinity form first and then combine to larger ones [Tinoco and Bustamante, 1999]. The folding path becomes relevant, and the longer a sequence, the more unlikely it is that its folding path leads to a global energy minimum. In other words, the longer the sequence, the less reliable are the results of minimum free energy folding. *pknotsRG* allows me to test this using a fairly large structure containing pseudoknots that has been proven experimentally. I considered the sequence of the group I intron from *Tetrahymena thermophila* (419 nt, GenBank accession: V01416). The MFE-structure found was quite different from the "true" structure taken from the literature [Cech, 1990]. I handcoded the experimental structure and evaluated its stability in my energy model. The result was striking: The experimental structure (-132.26 kcal/mol) was significantly far away from the possible minimum of free energy (-154.04 kcal/mol). So far in fact that it seems infeasible to detect the structure by scanning the space of near-optimal structures. This could be interpreted as the energy model being incorrect, but since it works well for short sequences, I suggest that this is an indication that the kinetics of folding already have a strong influence with this size of sequence, at least, when pseudoknots are involved.

While I have achieved a considerable speedup for predicting small pseudoknotted structures, it seems that the minimum free energy approach is not meaningful with the largest structures which it can now handle algorithmically. However, the situation changes when one is looking for particular structural motifs.

The search for such motifs using combinatorial matchers like *RNAmotif* [Macke *et al.*, 2001] is hampered by the problem that a motif description is either too specific and misses relevant instances, or else, it is too vague and produces a large number of different matches to the same sequence. An idea is to develop *thermodynamic matchers* (TDM) which are RNA folding programs based on the established MFE model but specialized to the particular structural motif at hand. Such a matcher returns the optimal way to fold a sequence into the motif structure together with the free energy of this folding. Comparing this energy to the MFE of an unrestricted folding can give a hint with respect to the significance of such a match. A TDM can easily be obtained via the *Locomotif* system [Reeder *et al.*, 2007], which in its newest version also incorporates the pseudoknot algorithm described in this thesis. In the next section, I demonstrate the development and application of a TDM for the detection of ribosomal frameshift signals.

5.6 Detection of programmed ribosomal frameshift signals

Parts of the work presented in this section have been implemented by Corinna Theis during her Bachelor thesis under my supervision. A paper describing this work is currently in preparation. In this section, I use the term "we" to refer to joint work with C. Theis.

5.6.1 Biological motivation

Programmed ribosomal frameshift (PRF) alters the reading frame by shifting the translating ribosome exactly one nucleotide to either the +1 or -1 direction. A stop codon in the original reading frame is then bypassed in the shifted reading frame. This way, two different protein products can be obtained from one mRNA. The frequency of frameshifting events at a particular site is used by viruses for a defined ratio between the two proteins. Changes in the ratio can lead to less efficient virus propagation and thus, can be a target for antiviral therapeutics. Recently, a role of -1 PRF in post-transcriptional regulation has been proposed in *Saccharomyces cerevisiae* [Jacobs *et al.*, 2007]. The authors propose a model where -1 PRF leads to premature termination targeting the mRNA for rapid degradation via the nonsense-mediated mRNA decay pathway (NMD).

-1 programmed ribosomal frameshift is most effective when two *cis*-acting signals are present: a heptameric *slippery site* and a stable secondary structure. The slippery site is the location of the actual frameshift event having the consensus sequence X XXY YYZ (triplets are shown for the pre-shifted reading frame). The structural element follows within a few bases and can be a simple stem-loop structure or a pseudoknot. It is believed that the pseudoknot promotes a higher frameshift efficiency, since it is more effective in pausing the ribosome. This can be explained by the torsional frameshift model [Plant and Dinman, 2005], in which a pseudoknot imposes a higher resistance to the translating ribosome. While a simple stem-loop can freely unwind, unwinding the first pseudoknot helix is restricted by the second helix. Energetically, this imposes a well-defined barrier, where ribosomes are directed to pause translation. It is also known that ribosome pausing is a necessary, but not sufficient prerequisite for frameshifting [Kontos *et al.*, 2001].

5.6.2 Previous work

Several computational studies, with the general goal of detecting new PRF events, were undertaken in recent years [Hammell *et al.*, 1999; Moon *et al.*, 2004; Bekaert *et al.*, 2003; Jacobs *et al.*, 2007]. The first study by Hammell *et al.* [1999] found over 200 putative PRF events in the yeast genome. Their approach relies on a strict pseudoknot structure consensus and requires two overlapping ORFs of at least 50 codons. In [Bekaert *et al.*,

2003], a machine learning approach is used to discriminate between strong and weak PRF signals. Another approach uses a combinatorial folding routine to identify possible frameshifting pseudoknots next to a slippery sequence [Moon *et al.*, 2004].

The most recent and probably most elaborate study has been used to identify over a thousand strong and statistically significant -1 PRF signals in the genome of *S. cerevisiae* [Jacobs *et al.*, 2007]. It is based on a two-step procedure, where the first step identifies slippery sequences followed by a possible frameshift pseudoknot. In the second step, they are analyzed statistically. In more detail, *RNAmotif* is used in the first step and finds all potential pseudoknots that could be formed according to the descriptor. The descriptor specifies the allowed loop and helix lengths which are extracted from known -1 PRF pseudoknots. All potential pseudoknots are then subjects of statistical analysis in step two. The potential pseudoknots are refolded with *pknotsRE*. The MFE of the folding is compared to folding energies of randomized sequences via z-score analysis. Finally, sequences with a low z-score (<-1.65) are regarded as statistically significant, since they appear to be more stable than expected by random.

Despite its overall good architecture, there is one shortcoming in the procedure. Sequences are forced to fold into a pseudoknot in the first step but are folded freely in the second step. Thus, the result of the procedure is indeed twofold: (i) The final candidates have the theoretical potential to fold a pseudoknot, and (ii) they have a MFE folding which is more stable than expected by random. However, it is not guaranteed that the stable structure which causes the good z-score actually is a pseudoknot. In fact, I found that from 1706 strong candidates only 163 contain a pseudoknot. The pseudoknot, which was folded by *RNAmotif* in step one, may have an energy similar to the MFE folding, but more likely, it will be less stable and hence, less probable to be formed in equilibrium.

In the light of these findings, I propose another approach for -1 PRF signal search: A specialized RNA folding program, called *pknotsRG-fs*, which explicitly folds a given sequence into the most stable structure conforming to the general frameshifting pseudoknot restrictions. The restricted folding energy can then be compared with the unrestricted MFE folding. This already gives a strong indication of whether the frameshift signal is likely to form or not.

In the remainder of this section, I will first introduce the specialized folding program and then show its incorporation into a tool for genome wide annotation of -1 PRF signals.

5.6.3 *pknotsRG-fs*: A specialized folding program

The RECODE database [Baranov *et al.*, 2003] contains information about 28 -1 PRF signals with a pseudoknot as cis-active signal. We analyzed the sequences and structures

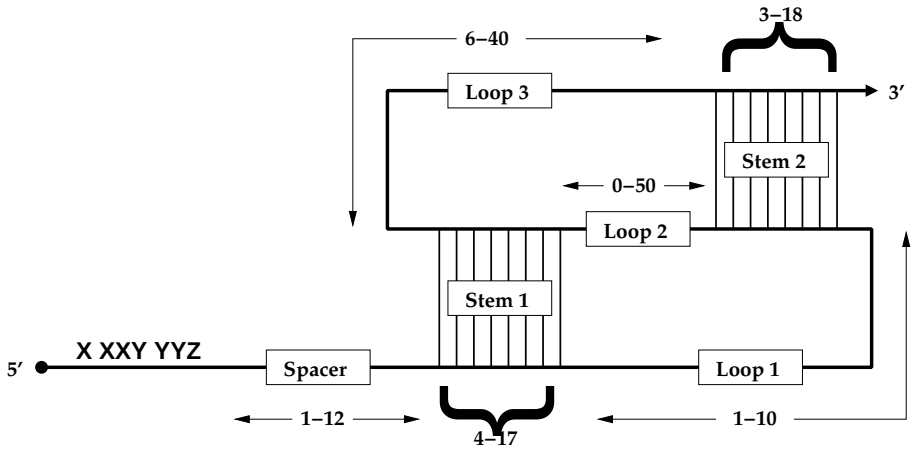


Figure 5.15: A consensus -1 PRF signal extracted from the RECODE database.

and extracted the consensus information displayed in Figure 5.15. The slippery sequence has consensus X XXY YYZ, where XXX stands for any three identical nucleotides, YYY for either three As or three T/Us, and Z for any nucleotide. The spacer region must contain at least one nucleotide and not more than twelve nucleotides. The stem and loop regions are chosen to include all but two examples which have unusually large loop sizes. It should be noted that loops 2 and 3 are large enough to further fold internally and thus, can have a stabilizing effect on the overall structure. E.g., the pseudoknot of the SARS coronavirus frameshift signal contains an additional third stem in loop 3 [Plant *et al.*, 2005].

Following these restrictions, I implemented a specialized folding routine devised to fold a sequence into the structure of lowest free energy while maintaining the given restrictions. Finding the structure with an optimal energy can, in general, be done with any Dynamic Programming (DP) approach. However, explicitly formulating the exact recurrences, which evaluate the folding space according to the above restrictions, is a tedious and error-prone task in a low-level programming language. A remedy to this is to use the more high-level ADP approach.

By some small changes to the original *pknotsRG* grammar, I obtain a new grammar precisely describing frameshift inducing pseudoknots. It incorporates all explicit length constraints but leaves enough freedom for the loop regions to fold into any further stabilizing structures. The underlying thermodynamic model [Mathews *et al.*, 1999] then comes for free – it has already been implemented and can be re-used. For efficiency reasons, I eventually compiled the grammar into a low-level programming language (C) using the ADP compiler [Steffen, 2006]. It then takes less than a second to fold a sequence of length

100. In the end, I obtain a thermodynamic matcher, *pknotsRG-fs*, which computes for a given input sequence the minimal free energy -1 PRF pseudoknot. Of course today, I would use the *Locomotif* system [Reeder *et al.*, 2007] for this task, but at the time of implementing the tool, *Locomotif* had no building block for pseudoknots. Based on *pknotsRG-fs*, we developed a search procedure termed *KnotInFrame*.

5.6.4 *KnotInFrame* search procedure

We devised a simple, yet effective pipeline for the detection of -1 PRF signals. It proceeds through the following steps:

1. Detect all slippery sequences (X XXY YYZ) in the correct reading frame (X=[A,C,G,T/U], Y=[A,T/U], Z=[A,C,G,T/U]). Also, there must not be an in-frame stop codon in the upstream sequence until the next in-frame start codon.

2. For each slippery site:

   2.1 Fold every $[40, 60, \ldots, 120]$ nucleotide long sequence downstream of the slippery site, once unconstrained with *RNAfold* and once constrained with *pknotsRG-fs*.

   2.2 For every subsequence, compute the *relative distance*:

   $$\Delta_{rel} = \frac{MFE_{RNAfold} - MFE_{TDM}}{length}$$

   2.3 Select maximal Δ_{rel} for this slippery site.

3. Sort and report all Δ_{rel} values for all slippery sites.

The choice of possible sequence lengths in step 2.1 is motivated by the actual size of frameshift signals found in RECODE. Except for two unusually long signals, all fit well in one of the five possible windows. These window sizes performed best in a simple window size and increment survey. In addition to the steps above, we already discard unlikely foldings in step 2.1, where either the MFE_{TDM} is rather high ($>$ -7.4 kcal/mol), or where the difference between MFE_{TDM} and $MFE_{RNAfold}$ is very unfavorable ($>$ 8.7 kcal/mol) for the pseudoknot. Usually, those candidates would be ranked very low anyways, but since we have no confidence in them, we remove them as early as possible. Note that some time can also be saved in step 2.1: Instead of folding each subsequence of length 40, 60, $\cdots$, 120 on its own, we only fold the longest sequence once and backtrace the optimal solutions for the shorter sequence from the same DP matrix. The optimal solution for the 40 nucleotide long subsequence can be backtraced starting from matrix entry $(0, 40)$ and correspondingly for the other subsequences.

Rank	1	2	3	4	5	6	7	8	not found
Count	18	4	1	-	1	-	1	1	2

Table 5.4: Result of *KnotInFrame* on the RECODE database. Most of the annotated -1 PRF signals are ranked on position one or two.

In the end, we obtain, for each input sequence, a sorted list of frameshift signal candidates. In order to see how this new method actually performs on real data, we turn again to the RECODE database.

5.6.5 Evaluation

We extracted all 28 sequence entries of RECODE annotated as -1 PRF signal in early 2007. In total, these sequences have a length of 230 kb and contain $\sim$ 800 slippery sites in the correct reading frame and are, thus, analyzed by our tool. Of those, $\sim$ 600 are discarded in step 2.1, due to a low folding energy as explained above. For most of the entries, *KnotInFrame* is able to predict the position of the true -1 PRF signal with rank one (18) or two (4). Some signals were ranked rather low, and two signals were not found at all, since they are either too long or have a slippery site not consistent with our consensus. Here, only four nucleotides are annotated in RECODE. A detailed overview of the results is given in Table 5.4.

At this point, the question arises if the true signals will also be ranked at a high position, when more slippery sites in a possibly longer input sequence compete for rank one. A more detailed look (see Figure 5.16) at the relative distances reveals that in fact, true -1 PRF signals have a tendency to have a higher Δ_{rel} than the $\sim$ 200 slippery sites not leading to PRF (called false positives in the following). However, the discriminative signal is not strong enough for a clear separation. One could use a relative distance of 0.1 as minimal threshold which leaves us with more true positives than false negatives. However, this threshold overlooks most true -1 PRF signals. A threshold of 0.05 captures over 50% of true positives, but then the number of false positives is approximately twice as large as the number of true positives. Finding the appropriate balance between high sensitivity and selectivity for this problem should therefore be governed by the intended use of the program.

In order to test the hypothesis that structural RNAs have a higher MFE than random sequences, we also performed a z-score analysis for all candidates reported by *KnotInFrame*. For each candidate, we shuffled 100 randomized sequences with the same dinucleotide content and folded them with *pknotsRG-fs*. In [Höchsmann *et al.*, 2006], it has

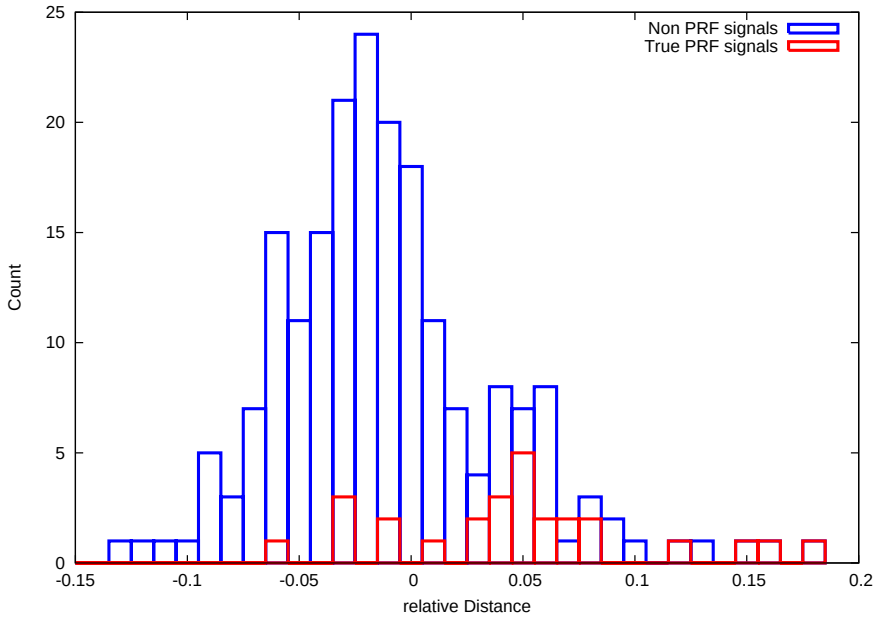


Figure 5.16: Distribution of relative distances for annotated -1 PRF signals (red) and non-shifting sequences (blue). Although, there is a bias for real PRF signals to higher relative distances, the overlap with non-PRF signals is too large for a clear separation.

been shown that using a TDM for z-score analysis provides a stronger signal than using a general folding algorithm. However, the outcome for -1 PRF signals is less fruitful than the results of [Höchsmann *et al.*, 2006] as shown in Figure 5.17. Most of the false positives have a z-score between 0 and -3. The fact that the z-scores are not centered at 0 is most certainly a result of discarding unfavorable folds in step 2.1. This already introduces a bias for more stable structures than expected by random. With a z-score of -3 and lower, the number of true positives equals approximately the number of true negatives. A small number of true positives is also deeply buried in the cloud of false positives which makes it hard to detect them either by z-score analysis or via the relative distance. Altogether, it turns out that with z-score analysis, we have the same difficulties in separating the true PRF signals from the wrong ones. Therefore, we refrain from using the z-score in *KnotInFrame*, since it would increase the runtime by a factor of 100 (for folding the randomized sequence) without a significant increase in performance.

It seems that with only one sequence at hand, we can do no better as aforementioned. However, the situation changes, if we have a set of homologous sequences. Then, the co-occurrence of a slippery site and a conserved, stable frameshifting pseudoknot could improve the search for -1 PRF signals in a future version of *KnotInFrame*.

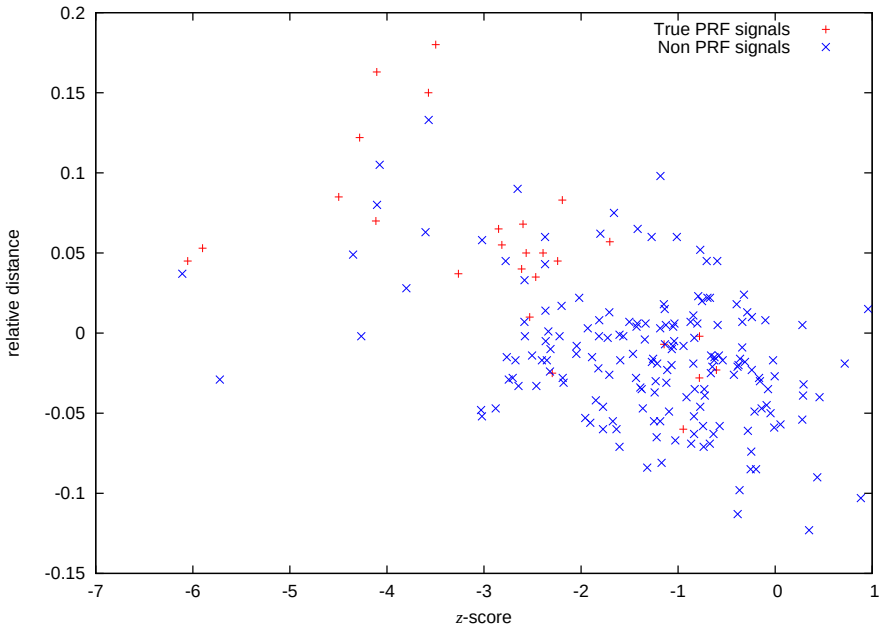


Figure 5.17: z-scores plotted against relative distances. Although only weakly correlated with Δ_{rel}, the z-score analysis does not provide significant further informations for separating -1 PRF signals from the background distribution.

5.7 Potential improvements of *pknotsRG*

The energy model for pseudoknots is by no means optimal. It is carefully optimized to predict as few spurious pseudoknots as possible while at the same time, correctly predicting most known pseudoknots. Nevertheless, I believe that the values are much more imprecise than the experimentally measured values for nested components. Unfortunately, it is unlikely that more precise values will be experimentally determined in the near future for the following reason: Pseudoknot stability is strongly dependent on the concentration of divalent ions. However, the ion concentration at which the energy parameters for the nearest-neighbor model were measured is unfavorable for pseudoknot formation. Therefore, it is questionable whether there will be a consistent model for nested structures **and** pseudoknots. One possible way to improve the implemented pseudoknot energy model would be to include the theoretical values reported in [Gultyaev *et al.*, 1999]. I could easily adapt *pknotsRG* to use a more complex energy model, such as a logarithmic and/or stem length dependent loop energy. However, then the problem arises of how to score pseudoknots more complex than those considered in [Gultyaev *et al.*, 1999]. In particular, a smooth transition in terms of energy from simple H-type to more complex pseudoknots

has to be assured.

Some extensions of the algorithm mentioned in this chapter are already implemented but need to be wrapped up before releasing them in the *pknotsRG* package. Those are the relaxation of canonization Rules 2 and 3 as described in Section 5.5.2 and the sparse version of *pknotsRG*. A variant that allows for small internal loops and bulges in the pseudoknot helices is also planned for the next release.

5.8 Software availability

The *pknotsRG* program family is available for download on the Bielefeld Bioinformatics Server[3], both as C source code and precompiled binary executable for most common platforms (Solaris, Linux, Windows and Mac OS X). It can also be used online with some restrictions, such as maximal sequence length and suboptimal energy barrier. However, the downloadable version has none of the Web Server restrictions. It folds arbitrary long sequences (given enough time and memory) and reports as many suboptimal solutions as the user requests.

Another useful extension for large scale analysis is a sliding window technique with adjustable window size and window position increment. For each position of the window, the analysis is performed according to the user settings (e.g. enforced mode with 20% suboptimals). When the window is shifted, only the non-overlapping part of the new window has to be computed, while the old part can be be re-used. Depending on the window size and increment, this saves a considerable amount of time compared to a wrapper program that folds each window independently. The window functionality as well as the enumeration of suboptimal structures were automatically compiled by the ADP compiler – another benefit of the ADP approach.

The Windows version contains a basic graphical user interface; the other versions provide a powerful interactive command line tool. All versions are available in the download section of the project home page at `http://bibiserv.techfak.uni-bielefeld.de/pknotsrg` .

KnotInFrame will be available online at `http://bibiserv.techfak.uni-bielefeld.de/knotinframe` .

[3] http://bibiserv.techfak.uni-bielefeld.de

Chapter 6

Consensus Shapes

In this chapter, I report on a new comparative secondary structure prediction algorithm termed *RNAcast*. It is based on the idea that related RNA structures, although not being identical, have to share the overall *shape* in order to perform a similar function. Following a short overview of related approaches, I define the mathematically precise meaning of a common *shape* and demonstrate its applicability in the tool *RNAcast*. This part has already been published [Reeder and Giegerich, 2005]. Then, I continue with an extension of the method presented leading to the tool *RNAforecast*. Finally, I merge last chapter's *pknotsRG* and *RNAcast* to yield a powerful, comparative pseudoknot folding algorithm.

6.1 Comparative structure prediction and the Sankoff algorithm

Single sequence minimum free energy folding methods, such as *Mfold* [Zuker and Stiegler, 1981; Zuker, 2003], *RNAfold* [Hofacker *et al.*, 1994], and *pknotsRG* [Reeder and Giegerich, 2004] are widely used today, although it is known that their results are not completely reliable. Their accuracy has been measured to be 73% for a short sequence set [Mathews *et al.*, 2004]. For longer sequences, the accuracy drops down to as low as 41% [Doshi *et al.*, 2004]. Better results are generally achieved by comparative analysis of a family of homologous sequences, where sequence and structure conservation is exploited using a resolved tertiary structure whenever available, sequence alignment, statistical methods, and human expertise [Gutell *et al.*, 1992].

A first comparative approach based on thermodynamics was formulated by Sankoff as early as 1985 [Sankoff, 1985]. It performs sequence alignment and minimal free energy folding simultaneously. The time complexity is $\mathcal{O}(n^6)$ with space $\mathcal{O}(n^4)$ for two

sequences of length n, and for more sequences, it becomes exponential in the number of sequences. Given these high computational costs, it seemed unlikely that this algorithm would ever be put into practice. For many years, it rested in oblivion. Recently, however, interest in comparative methods for RNA structure prediction has been nurtured by findings on the functional versatility of RNA and several related approaches have been suggested. Some emphasize the sequence conservation aspect, folding a predetermined sequence alignment under thermodynamic rules (*RNAalifold*, [Hofacker *et al.*, 2002]). The other extreme emphasizes thermodynamics and suggests to use multiple structure alignments of independently folded sequences [Höchsmann *et al.*, 2004; Siebert and Backofen, 2005].

Then, some approaches directly implement Sankoff's idea of simultaneous alignment and folding but introduce various pragmatic restrictions, e.g. *Dynalign* [Mathews and Turner, 2002] and *Foldalign* [Gorodkin *et al.*, 1997]. In the early days, those algorithms restricted the search space in a straightforward way in order to achieve practical programs: *Foldalign* aligned only subsequences with a predefined maximal length difference; *Dynalign* restricted the maximal distance between two aligned residues. For a review of these and other tools, the reader is referred to the study of [Gardner and Giegerich, 2004]. However, in their current versions a more dynamical way of restricting the search space is employed. *Dynalign* [Harmanci *et al.*, 2007] uses (pure) sequence alignment probabilities to constrain the (structural) alignment to positions with a significant probability. On the other hand, *LocARNA* [Will *et al.*, 2007] (see also Chapter 5.4) uses single sequence base pair probabilities from the partition function to direct the structural alignment to use only significant base pairs in the consensus structure. *Foldalign* 2.1 [Havgaard *et al.*, 2005, 2007] prunes the search space by discarding subalignments which do not exceed a length-dependent minimal score. Each of these heuristics offers a drastic improvement in runtime at a small possible loss in accuracy.

A different approach to Sankoff has been implemented in the tool *CMfinder* [Yao *et al.*, 2006]. In an expectation-maximization scheme, an alignment and a Covariance Model (CM) are iteratively refined, so that in the end, the CM holds the consensus information, and the alignment is optimal with regards to the CM.

6.1.1 Comparative RNA gene prediction

The strengthened interest in this field is mostly due to several recently described RNA gene finders based on comparative RNA structure prediction, the most prominent one being *RNAz* [Washietl *et al.*, 2005b]. They all use a similar strategy but each with a different consensus structure prediction algorithm at their core. The main idea is that

if a structure is more conserved in various species than expected by random, it is most likely under evolutionary pressure and hence, an RNA gene. *RNAz* employs *RNAalifold* to compute consensus structures and support vector machines for the classification as functional or non-functional RNA. In a prototype screen for conserved RNA structural elements, *RNAz* found thousands of putative functional non-coding RNAs in the human genome [Washietl *et al.*, 2005a]. A similar screen based on the *EvoFold* program was performed in [Pedersen *et al.*, 2006]. For its structure predictions, *EvoFold* uses *Pfold* [Knudsen and Hein, 1999], an algorithm based on phylogenetic stochastic context free grammars (phylo-SCFG). Interestingly, both screens yield approximately 40000 candidate genes, but their predictions overlap only by a small amount.

The drawback of these type of RNA gene finders is that they need a sufficiently accurate sequence alignment as prerequisite. Yet, with RNA genes, it is often the case that although the structure is well conserved, the sequences are only conserved less than 50%. In such cases, pure sequence based alignment algorithms, such as *ClustalW*, will often fail to produce a meaningful alignment. Hence, the subsequent consensus structure prediction will fail, too. There is thus an implicit paradox: The alignment step tries to minimize sequence variation, while the structure prediction tries to maximize the covariation - a better alignment (in terms of alignment score) might result in a worse covariation score.

A solution to this is to use Sankoff-style algorithms and perform both steps simultaneously. Of course, this makes its application much more costly. A gene finder based on *Dynalign* is introduced in [Uzilov *et al.*, 2006], and a screen performed with a *Foldalign* based gene predictor has been reported in [Torarinsson *et al.*, 2006]. This particular screen took about seven months on 70 computers and analyzed $\sim$1% of the human and mouse genome. Clearly, this is far away from regular use for complete genome annotation pipelines.

Behind all these approaches, there is the original Sankoff approach as the ideal method — the one that every program tries to approximate in different ways. "Making Sankoff practical" has been a recurring theme at the meetings of the computational RNA community. However, this road may require so many pragmatic restrictions that the ideal looses much of its attraction.

A way out of this dilemma may be to change the definition of a consensus structure. In Sankoff's approach, the consensus is a folded sequence alignment that optimizes a combined sequence similarity and energy score. What if we drop the implicit multiple sequence alignment step (as this problem is known to be NP-complete)? Let us agree that a consensus structure for sequences $s_1, \ldots, s_k$ is a set of structures $x_1, \ldots, x_k$, one

for each s_i, that all have – in some mathematically precise sense – a common shape. Should a sequence alignment of $s_1, \ldots, s_k$, compatible with the consensus, also be desired, it may be computed afterward from $x_1, \ldots, x_k$, rather than from $s_1, \ldots, s_k$, by multiple *structure* alignment [Höchsmann *et al.*, 2004]. The latter phase will certainly need to resort to heuristics, but for the first phase, there may be a chance to achieve a complete and non-heuristic solution in acceptable time.

6.2 An alternative to the Sankoff method

6.2.1 A hypothetical method

To explain the new approach, let us first consider a hypothetical, exhaustive method. Let s_1 and s_2 be two RNA sequences, both of length n. Let us enumerate their foldings in order of increasing free energy, yielding $x_1, x_2, \ldots, x_{N_1}$ for s_1 and $y_1, y_2, \ldots, y_{N_2}$ for s_2. The numbers N_1 and N_2 will be very large, even for small n, but let us ignore this for the moment.

If s_1 and s_2 have a common structure, there must be $x_i = y_j$ for some i and j. In fact, there may be many such pairs. We rank them by $(i + j)$, and the pair (x_i, y_j) with minimal rank is our predicted consensus. Just as well, we may produce the k top-ranking consensus pairs.

Using known algorithmic techniques, one can implement the enumeration in $\mathcal{O}(n^3 + n(N_1+N_2))$ time and $\mathcal{O}(n^2)$ space and the identification of common structures in $\mathcal{O}(n(N_1+N_2))$ time and space, where structures are represented as strings and keyword or suffix trees are employed for fast identity matching. Clearly, if we add a third sequence s_3, with structures $z_1, z_2, \ldots, z_{N_3}$, the $(N_1 + N_2)$ above is replaced by $(N_1 + N_2 + N_3)$, and hence, this method is additive in the number of sequences! Too bad it is not practical for the following two reasons:

- The numbers $N_1, N_2, \ldots$ are very large and N_i grows exponentially with n. Even if we restrict enumeration to an energy range of say 10% above the minimal free energy, N_i may be large as 100 000 or 1 000 000. This alone might not be a threat on today's computers, but here is our second problem:

- Sequences s_1 and s_2 need not have the same length, and hence, their structures cannot be identical. We must allow for some flexibility in the relative position of helices. Therefore, we need to resort to some pairwise similarity computation, catapulting computation time of the identification phase to $\mathcal{O}(n^2 \cdot N_1 \cdot N_2)$ or higher. The additive behavior is lost.

To make the hypothetical method practical, we need to restrict enumeration to a small, but representative sample of the folding space and achieve identification of consensus pairs in linear time in spite of their not being identical.

6.2.2 Outline of the consensus shapes prediction method

I build on the recent approach of *abstract RNA shapes analysis* [Giegerich *et al.*, 2004b] to solve both of the above problems. Deferring formal definitions, a shape is a family of structures sharing a common pattern of helix nesting and adjacency. The near-optimal folding space contains only a (relatively) small number of shapes. Using abstract shapes analysis, I enumerate representative structures – one per shape, and only those! – for both s_1 and s_2. The highest ranking structure pair x_i and y_j, where both have the same shape, then forms the consensus pair. While the structures x_i and y_j are only similar, their shapes can be easily computed, and identity matching on shapes can be implemented in time $\mathcal{O}(n \cdot (N_1 + N_2))$ as sketched above – for significantly reduced N_1 and N_2.

These ideas will be rigorously described below, and I shall report on their implementation and evaluation.

6.3 RNA shape analysis and consensus shapes

6.3.1 Abstract shapes

Here is a summary of the basic definitions of abstract shapes analysis.

- An RNA sequence s has **folding space** $\mathcal{F}(s)$, the set of all admissible structures under the given base pairing rules. For each structure $x \in \mathcal{F}(s)$, one can compute its free energy $E(x)$.

- The **minimal free energy structure** $mfe(s)$ for a sequence s is the structure $x \in \mathcal{F}(s)$ where $E(x)$ is minimal.

- For efficient computation of shapes via Dynamic Programming, they must be represented as trees. 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.

- The **abstract shape space** of sequence s is $\mathcal{P}(s) = \{\pi(x) \mid x \in \mathcal{F}(s)\}$. The class of p-shaped structures in $\mathcal{F}(s)$ is $\mathcal{F}(s|\ p) = \{x|\ x \in \mathcal{F}(s), \pi(x) = p\}$.

- The **shape representative structure** $\hat{p} \in \mathcal{F}(s)$ for shape p is the structure whose free energy is minimal among all members of that shape class. It is called *shrep* for short.

Abstract shape analysis, as implemented by the program *RNAshapes*, is computed for an RNA sequence s and an energy range R. It delivers a list $[(p_1, \hat{p}_1), \ldots (p_k, \hat{p}_k)]$ with the following properties:

- the p_i are different shapes, and the $\hat{p}_i$ are their respective shreps,
- the list is ordered by increasing energy: $mfe(s) = \hat{p}_1$ and $E(\hat{p}_i) \leq E(\hat{p}_{i+1})$,
- the list is restricted to the energy range indicated by $R : E(\hat{p}_i) \leq E(mfe(s)) + R$,
- the list completely covers this energy range in the abstract shape space: There is no shape p_{k+1} such that $E(\hat{p}_{k+1}) \leq E(\hat{p}_1) + R$

The strength of shape analysis lies in four aspects (for details see [Giegerich *et al.*, 2004b]):

- It produces a non-heuristic, mathematically well-defined synoptic view of the near-optimal folding space, directing the focus on a small number of shreps.
- Shape analysis uses the full energy model [Mathews *et al.*, 1999] and runs in the same asymptotic space and time complexity as suboptimal RNA folding.
- Shapes are meaningful across sequences, hence lend themselves to a comparative approach. This aspect is exploited here for the first time.
- The approach is generic with respect to the shape abstraction (π) that is actually used. Shapes can be more or less abstract depending on the level of detail considered relevant.

I illustrate the latter point by defining two shape abstractions, used in this study, in more detail. In general, shape abstractions retain nesting and adjacency of helices but disregard their size and concrete position in the primary sequence. They may choose to retain or to discard bulges and internal loops which leads to different levels of abstraction. For a mapping of an example structure see Figure 6.1. “Level 5” is the strongest abstraction and does not account for bulges and internal loops at all. It only records hairpins and multiloop bifurcations. “Level 3” retains helix interruptions, but does not specify whether they result from 5’-bulges, 3’-bulges or internal loops.

As I am not concerned with the algorithmics of shape analysis here, I can ignore tree-like representations of structures and shapes and define shape abstractions as mappings

$$
\begin{array}{llll}
\pi_5(.) & = \varepsilon & \varrho_5(.) & = \varepsilon \\
\pi_5(.s) & = \pi_5(s) & \varrho_5(.s) & = \varrho_5(s) \\
\pi_5(s.) & = \pi_5(s) & \varrho_5(s.) & = \varrho_5(s) \\
\pi_5((s)) & = [\varrho_5(s)] & \varrho_5((s)) & = \varrho_5(s) \\
\pi_5((s)s') & = [\varrho_5(s)]\pi_5(s') & \varrho_5((s)s') & = \pi_5((s)s') \\
 & & & \\
\pi_3(.) & = \varepsilon & \varrho_3((s)) & = \varrho_3(s) \\
\pi_3(.s) & = \pi_3(s) & \varrho_3(s) & = \pi_3(s) \quad \textit{in all other cases} \\
\pi_3(s.) & = \pi_3(s) & & \\
\pi_3((s)) & = [\varrho_3(s)] & & \\
\pi_3((s)s') & = [\varrho_3(s)]\pi_3(s') & &
\end{array}
$$

Table 6.1: Definition of level-5 (π_5) and level-3 (π_3) shape abstractions. s and s' denote a non-empty, well-balanced dot-bracket string, and ε denotes the empty string. Brackets in the input/output string are written in bold face. Note that the difference lies with ϱ_5 versus ϱ_3, where the former reads across bulges and internal loops, while the latter decides to record a new helix part with every interruption.

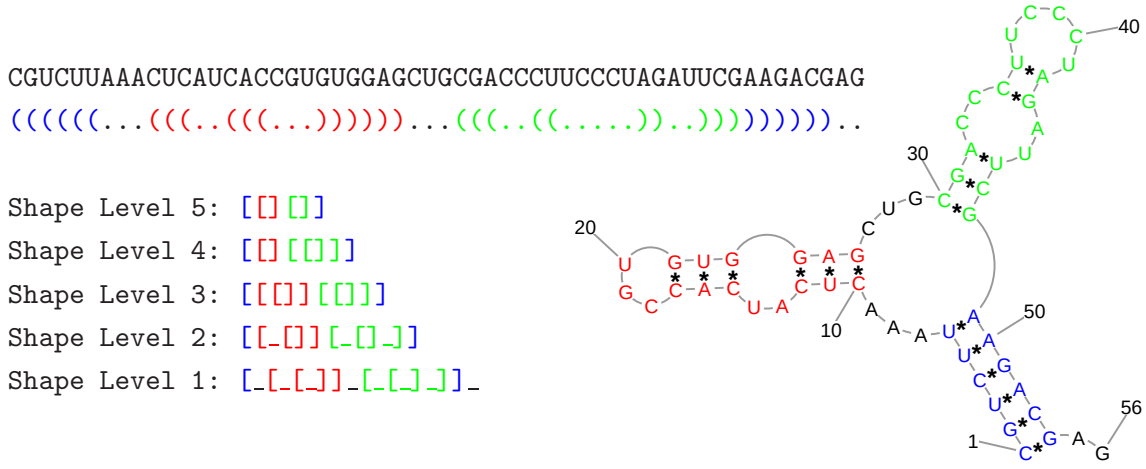


Figure 6.1: An example secondary structure and its five shape representations, implemented in the tool *RNAshapes* [Steffen *et al.*, 2006].

from the more familiar string representations of structures to string representations of shapes. Structures are represented as dot-bracket strings, e.g. `((((((...(((..(((...))))))...(((..((.....))..)))))))))...` The level-5-shape of this structure is represented as "`[[][]]`", its level-3-shape as "`[[[]][[]]]`". In Table 6.1, I provide equations defining shape abstractions π_5 and π_3.

Rank of true shape	1	2	3	4	5	5-9	10-19	20+	total
lin4	9	0	0	0	0	0	0	0	9
IRES	5	2	0	0	0	0	0	0	7
tRNA	3	1	5	2	0	0	0	0	11
srp RNA	2	2	0	0	0	0	0	0	4
riboswitch	7	0	0	0	0	0	0	0	7
S box	4	5	2	0	0	0	0	0	11
5S rRNA	1	2	0	1	0	1	0	0	5
U12 RNA	0	0	0	0	1	0	1	4	6
U1 RNA	1	1	0	1	0	0	1	0	4
U2 RNA	0	0	0	0	0	3	0	2	5

Table 6.2: Ranks of the true shape in the list of near-optimal shapes using *RNAshapes*. The true shape is on one of the first ten ranks in 61 out of 69 cases.

Based on my experience, I generally suggest to work with the less abstract Level 3 except for long molecules, where a stronger abstraction speeds up the program, because the shape space is reduced further.

6.3.2 Rankings of true shapes

In order to evaluate whether shape analysis bears promise towards consensus prediction, I performed two preliminary studies using several sequence families from Rfam [Griffiths-Jones *et al.*, 2003] and other databases (Table A.2 in the appendix), where the "true" structure s is known. From this true structure, I computed the "true" shape $p^* = \pi(s)$.

The first question then asks for the rank i such that $p_i = p^*$ in the list of shapes returned by shape analysis. Table 6.2 shows the outcome. The average rank of the true shape is 5.06, and in 32 out of 69 cases (46%) the true shape has rank one.

The advantage of shape analysis over complete suboptimal folding [Wuchty *et al.*, 1999] is confirmed by two detailed observations: For one of the tRNA sequences, the true shape has rank three, while the true structure has rank 104 in the complete enumeration. In the worst case observed, a U12 RNA sequence, the true shape has rank 28, while its associated true structure has rank 3 695 033. This confirms my intuition that the shape space is small enough to completely enumerate its interesting part. Yet, it also confirms that the reliability (in terms of correctly predicted shapes) of single sequence folding is around 46% – not useless, but not dependable either.

Secondly, I investigated whether this improves when we move towards a comparative

Rank of true shape	1	2	3	4	5	≥ 6	total
lin4	36	0	0	0	0	0	36
IRES	11	10	0	0	0	0	21
tRNA	22	22	11	0	0	0	55
srp RNA	5	1	0	0	0	0	6
riboswitch	21	0	0	0	0	0	21
S box	21	31	3	0	0	0	55
5S rRNA	8	1	1	0	0	0	10
U12 RNA	0	3	0	0	0	12	15
U1 RNA	4	0	1	0	1	0	6
U2 RNA	0	0	0	0	0	10	10

Table 6.3: The table shows the rank of the reference shape in all pairwise comparative predictions. The average rank of the true shape improves from 5.06 for single sequence prediction to 3.06.

approach by using pairs of sequences. In Table 6.3, all pairs of predictions (within each family) are considered, and the rank of the true shape in the list of all *common* shapes is reported. In the pairwise approach, the average rank of the true shape improves to 3.13, and the true shape now has rank one in 128 out of 235 cases (53%). I conclude that the power of comparative analysis is well captured by my approach, and I expect even better performance from using three or more sequences. Based on these observations, I developed the method of consensus shape prediction.

6.3.3 Consensus shape prediction

For a set of sequences $\{s_1, \ldots, s_k\}$, intentionally a family of related RNA sequences, I enumerate their shape spaces $\mathcal{P}(s_1), \ldots, \mathcal{P}(s_k)$. Upon those, I define:

Definition 1 *A shape p is a **common shape** of $\{s_1, \ldots, s_k\}$ if $p \in \bigcap_{i=1}^{k} \mathcal{P}(s_i)$.*

Definition 2 *The **consensus shape** for sequences $\{s_1, \ldots, s_k\}$ is the common shape p that minimizes $rank(\hat{p}_1, \ldots, \hat{p}_k)$.*

Here, *rank* is a scoring function that combines the individual shrep scores. I discuss several meaningful scoring functions in the next section.

Computing the intersection of $\mathcal{P}(s_1), \ldots, \mathcal{P}(s_k)$, once those are generated, is a trivial task. Usually, there are far fewer common shapes than there are shapes in $\mathcal{P}(s_1), \ldots, \mathcal{P}(s_k)$.

Then, I sort all common shapes by their rank and find the consensus shape as the first one in the list.

Note that the above definitions do not only yield the consensus shape, but moreover, from shape analysis, one also gets the set of shreps – the resulting output is a $(k+1)$-tuple $(p, [\hat{p}_1, \ldots, \hat{p}_k])$. These shreps constitute an (unaligned) multiple RNA structure prediction for the input sequences.

6.4 Algorithm implementation

6.4.1 The program *RNAcast*

The above method has been implemented in the program *RNAcast*[1] which stems from "**RNA** **c**onsensus **a**bstract **s**hapes **te**chnique". Although most of the method is clear from the definition of the consensus shape, a few details remain to be fixed.

The algorithm proceeds over three distinct phases:

Step 1: The algorithm starts with sequences $s_1, \ldots, s_k$ as input and an energy threshold R. Let n be their average length. It runs *RNAshapes* on each individual sequence with the provided energy range R. Theoretically, every sequence could have its own R, but in practice, only one is used.

Step 2: Within the k resulting lists (the shape spaces), it identifies all shapes that occur in all the lists. Hashing techniques are used for fast identity matching of shapes. Thus, this phase runs in time proportional to $k \cdot n \cdot |\mathcal{P}(s_1)|$, since it iterates only over the list $\mathcal{P}(s_1)$, and the hash table is used to look for co-occurrences in the other shape spaces $\mathcal{P}(s_2), \ldots, \mathcal{P}(s_k)$. After this step, the program holds a list of all l common shapes together with their shreps: $[(p_1, [\hat{p}_1^1, \ldots, \hat{p}_k^1]), \ldots, (p_l, [\hat{p}_1^l, \ldots, \hat{p}_k^l])]$.

Step 3: Finally, it evaluates each common shape with a scoring function and produces a sorted list of all common shapes. The first shape of this list is returned as the consensus shape along with its shreps. If desired, the $r \leq l$ best common shapes can be reported as well.

I suggest to use the output of *RNAcast* as input for *RNAforester* [Höchsmann *et al.*, 2004], a multiple RNA structure alignment program. (Another use of *RNAforester* for improving *RNAcast* predictions is given in Section 6.7). The unaligned *RNAcast* output is shown in Figure 6.2. A structural alignment is shown graphically in Figure 6.3.

[1] Available at `http://bibiserv.techfak.uni-bielefeld.de/rnacast/`

```
Shape: [[[]][][]]       Score: -223.50     relative Score: 0.99
CCUUUGCAGGCAGCGGAAAUCCCCACCUGGUAACAGGUGCCUCUGCGGCCAAAAGCCACGUGUAUAAGAUACACCUGCAAAGG
((((((((((..(((((......((((((....))))))..))))))(((.....)))..((((.....)))).)))))))))   (-34.10) R = 2
CCUUUGCAGGCAGCGGAAUCCCCCACCUGGUGACAGGUGCCUCUGCGGCCGAAAGCCACGUGUGUAAGACACACCUGCAAAGG
((((((((((..(((((......((((((....))))))..))))))(((.....)))..((((.....)))).)))))))))   (-39.10) R = 2
GCACGCAAGCCGCGGGAACUCCCCCUUGGUAACAAGGACCCGCGGGGCCGAAAGCCACGUUCUCUGAACCUUGCGUGU
(((((((((((((((.......(((((....)))))).)))))))(((.....)))..((((...))))))))))))        (-34.10) R = 2
GCAUGAUGGCUGUGGGAACUCCCCCUUGGUAACAAGGACCCACGGGGCCAAAAGCCACGUCCUCACGGACCCAUCAUGC
(((((((((((((((.......(((((....)))))).)))))))(((.....)))..((((....))))))))))))        (-34.70) R = 3
GCAUGACGGCCGUGGGAACUCCUCCUUGGUAACAAGGACCCACGGGGCCAAAAGCCACGCCCACACGGGCCCGUCAUGU
(((((((((((((((......((((((....))))))))))))))))(((.....)))..((((....))))))))))        (-41.90) R = 1
GCAUGUUGGCCGUGGGAACACCUCCUUGGUAACAAGGACCCACGGGGCCGAAAGCCAUGUCCUAACGGACCCAACAUGU
(((((((((((((((......((((((....))))))))))))))))(((.....)))..((((....))))))))))        (-39.60) R = 1
```

Figure 6.2: Example output for a family of IRES elements of picornaviruses. First, it shows the common shape and the achieved score (absolute and relative). Thereafter, for each input RNA, the sequence, the predicted shrep, its energy, and its individual rank in the shape space is given. Note that the sequences are not aligned.

Left to be defined is the scoring function *rank*. I propose and test four different possibilities:

1. Rank sum:
$$rank_1(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i) = rank(\hat{p}_1^i) + \cdots + rank(\hat{p}_k^i)$$
Each shrep contributes with its individual rank in the sorted shape space of its sequence.

2. Sum of energies:
$$rank_2(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i) = E(\hat{p}_1^i) + \cdots + E(\hat{p}_k^i)$$

3. Normalized sum of energies[2]:
$$rank_3(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i) = \frac{E(\hat{p}_1^i)}{E(mfe(s_1))} + \cdots + \frac{E(\hat{p}_k^i)}{E(mfe(s_k))}$$

4. Sum of probabilities:
$$rank_4(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i) = Prob(\hat{p}_1^i) + \cdots + Prob(\hat{p}_k^i)$$
where $Prob(...)$ are the structure probabilities from the partition function [McCaskill, 1990], requiring additional $O(k \cdot n^3)$ steps to compute. In this case, of course, we have to maximize over all scores.

[2]This feature has been suggested by the audience of the 2005 Benasque RNA meeting.

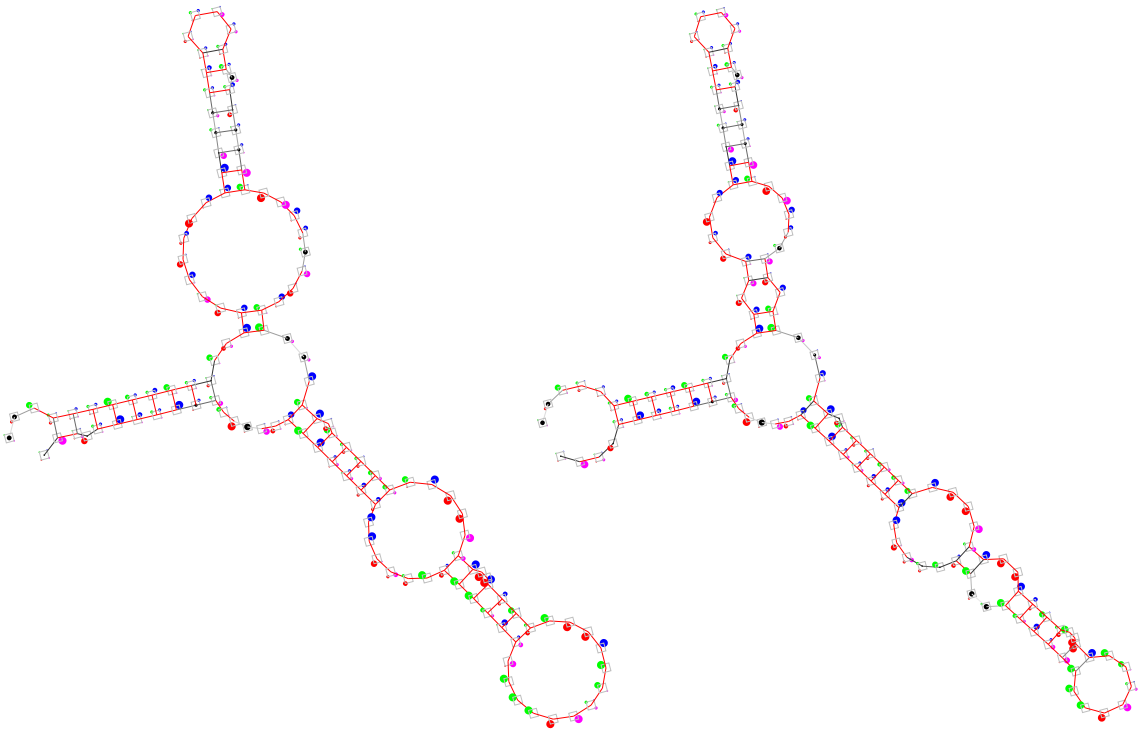


Figure 6.3: Two multiple RNA structure alignments of five 5S rRNA computed by *RNAforester*. On the left-hand side, the alignment of the structures as found in the database [Szymanski *et al.*, 2000] is depicted. On the right-hand side, the output shreps of *RNAcast* served as input for the alignment. Obviously, the structures are similar, with *RNAcast* predicting a few additional *compatible* base pairs. The alignment visualization should be interpreted as follows: The frequencies of the bases A, C, G, U are proportional to the radius of circles which are arranged clockwise on the corners of a square for each residue, starting at the upper left corner. Additionally, these circles are colored red, green, blue, and magenta for the bases A, C, G, and U, respectively. The frequency of a gap is proportional to a black circle growing at the center of the square.

Functions $rank_2, rank_3$, and $rank_4$ can easily be normalized to the interval $[0 \ldots 1]$. This simplifies comparisons between different RNA families under evaluation. Note that rescaling of the scores does not have an influence on the order of common shapes. Therefore, *RNAcast* also reports the relative score in its output (see Figure 6.2).

Overall, it turned out that $rank_2$ and $rank_3$ perform equally good and slightly better than the other two scoring functions. $rank_3$ may be superior to $rank_2$, if the sequences under evaluation exhibit large differences in the MFE values. In that case, the relative distance to the respective MFE value (considered by $rank_3$) may be more meaningful than

the absolute distance (considered by $rank_2$). However, I did not observe any differences in the predictions with the test set. As expected, $rank_4$ performs better than the simple rank sum score. But, the prediction accuracies for all four scoring functions do not differ much. The method seems to be relatively robust concerning the choice of scoring function. I decided to use $rank_2$ for all computations discussed in this chapter.

6.4.2 Technical limitations

Mathematically, at least one common shape always exists for any set of sequences. This does not hold when the search space is limited, and hence, the choice of the energy range R is critical: If the true shape is missing in the enumerated shape space of one of the input sequences, (because the energy of its shrep is too high), this shape can neither become a common shape nor the consensus. The predicted consensus will be wrong in this case. This has consequences: Theoretically, there is no upper bound for the number k of sequences to be considered, and from the efficiency point of view, k can be quite large. But with larger data sets, it is more likely that one of the sequences is an outlier, and the chosen energy range is just not large enough. Then, the true consensus shape is missed due to the effect explained above. Increasing k implies a tendency to increase R, unless we know in advance that the data set is very homogeneous. Whenever more sequences require us to look deeper into the shape space, these two sources of increased efforts multiply. In practice, I suggest to be cautious with more than ten input sequences, unless it is guaranteed that no outlier is in the input set.

Another point of advice can be given as a rule of thumb: All predicted consensus shreps have the same number of helices. If their native structures are suspected to have different numbers of helices (as with a mixture of four and five stem tRNAs), the energy threshold should be large enough to accommodate the loss of the extra stem.

6.5 Evaluation

Based on the preliminary tests, I concluded that my method is capable of identifying the correct shape in 53% of all pairwise predictions. When using *RNAcast* in the multiple way, the correct shape is predicted for six out of ten families, and for three further families the true shape is on rank two or three. Still, predicting the correct shape alone is not good enough. Within a shape class, considerable structural variation can occur. Since shapes abstract from concrete helix positions and sizes, it is theoretically possible that the shrep of a correct shape does not share a single base pair with the true structure.

In this section, I evaluate the accuracy achieved by *RNAcast* on the base pair level and compare it to other tools.

In particular, I answer the following questions:

1. How accurate are the shreps given the correct shape?
2. What is the improvement over single sequence folding algorithms?
3. How does *RNAcast* perform compared to other pairwise and multiple folding algorithms?
4. What are the reasons for wrong predictions?

I analyze the structure predictions in terms of sensitivity, selectivity, and the Matthews correlation coefficient (MCC):

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Selectivity = \frac{TP}{TP + FP}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

True positive base pairs (TP), true negatives (TN), and false negatives (FN) are counted as usual. No slipping of helices is allowed. For false positives (FP), I use the counting method proposed by [Gardner and Giegerich, 2004]: Predicted base pairs that do not occur in the reference structure, but are *compatible* with it, are not counted for FP. A base pair $i \bullet j$ is *compatible* if neither i nor j is paired to another base in the reference, and there is no other base pair $k \bullet l$ that violates the nesting convention (i.e. $k < i < l < j$). This assumption is meaningful, since the reference structures used in this study are often based on a consensus. The members of a specific RNA family share most of the base pairs in the consensus but may have additional ones.

6.5.1 Accuracy of the true shrep

Let us first assume that we already know the correct shape of the family under evaluation. We are then asking for the corresponding shreps. We can either look them up in the *RNAshapes* output, or we can generate an RNA folding program restricted to that specific shape and compute the optimal structure directly. I employed this strategy for the same

set of RNA families as used in the preliminary study and evaluated the accuracy of the shreps.

On average, sensitivity is 78.2% and selectivity is 78.6%, compared to 65.4% and 65% respectively for the MFE-prediction of the single sequence. This shows that knowing the correct shape improves secondary structure prediction of single sequences significantly. However, in most realistic cases, we do not know the true shape in advance. The best we can do then is to rely on the consensus shape computed by *RNAcast*. Note that even when the predicted consensus shape is incorrect, it may still be close to the correct shape, in which case the predicted structures may also resemble the truth closely.

6.5.2 Improvement over single sequence prediction

Next, I determined the accuracy of the structures predicted by *RNAcast*, regardless of the predicted shape being correct or not.

I folded each RNA family in five different ways:

1. Single sequence prediction using *RNAfold*,

2+4. *RNAcast* on all pairwise combinations using shape abstractions π_5 and π_3, and

3+5. *RNAcast* in a multiple way on all family members at once, again in each case with π_5 and π_3. The energy threshold R was set to 10 kcal/mol.

Figure 6.4 shows the (average) MCC of each prediction method. One can see that "going comparative" pays off: In all cases but one (multiple tRNA folding with π_5), *RNAcast* performs better (or equal) than single sequence prediction. The clover leaf prediction for tRNA failed - one arm of the clover leaf was missed. However, using the less abstract shape mapping π_3 yields the correct shape and a higher accuracy. One can further see that using multiple sequences increases the reliability of the prediction. Overall, π_3 gives the highest accuracy especially for shorter sequences ($\leq$ 150 bases), where additional bulges or internal loops may be more important than in longer sequences. The averaged MCC for *RNAcast* multiple with shape abstraction π_3 is 0.77. This is an obvious increase compared to 0.64 for single sequence prediction. With π_5, the MCC is 0.72. Pairwise predictions using π_3 have an average MCC of 0.73 (0.7 for π_5) – lower than for the multiple approach, but still better than the single sequence approach.

While these results are already promising, I also want to relate *RNAcast*'s performance to existing comparative tools.

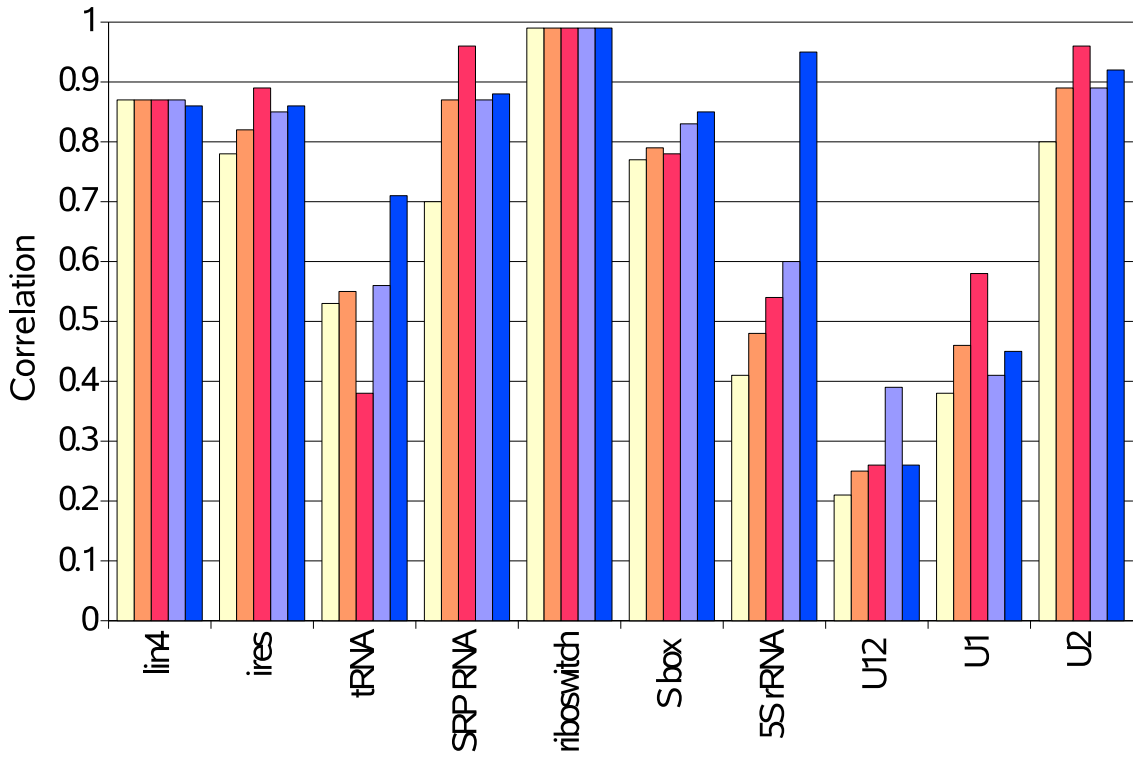


Figure 6.4: Accuracy (MCC) of *RNAcast* on a set of RNA families sorted by their average size. The bars correspond to *RNAfold* (light yellow), *RNAcast*-π_5 pairwise (orange), *RNAcast*-π_5 multiple (red), *RNAcast*-π_3 pairwise (light blue), and *RNAcast*-π_3 multiple (dark blue).

6.5.3 Comparison to the Sankoff approach

Comparison to *Dynalign*

Dynalign was chosen as a representative of the (pairwise) Sankoff approach. In [Mathews and Turner, 2002], the sensitivity of *Dynalign* is measured on a set of 5S rRNA. I found a secondary structure for five sequences of that set in the database [Szymanski *et al.*, 2000] and applied *RNAcast* on them. Single sequence prediction performs relatively bad on this data set (see row *RNAfold* in Table 6.4). Using *RNAcast* in a pairwise fashion clearly improves the accuracy, but the results are still not satisfying. The average sensitivity of (pairwise) *Dynalign* is 84.2% which is only topped by running *RNAcast* multiple on all five sequences simultaneously. Then, a sensitivity value of 92% and an almost perfect selectivity of 97.8% can be achieved.

Program	Sensitivity	Selectivity	Correlation
RNAfold	43.08	41.2	0.41
Dynalign	84.20	-	-
RNAcast (pairwise) π_3	59.10	62.40	0.60
RNAcast (multiple) π_3	91.98	97.82	0.95

Table 6.4: Prediction accuracy for a set of 5S rRNA. Note: The evaluation of *Dynalign*, taken from [Mathews and Turner, 2002], allows for slipping helices which we do not allow in our evaluation. Selectivity and correlation were not given in [Mathews and Turner, 2002].

	RNAcast pairwise		*RNAcast* multiple		*Carnac*		*Dynalign*	
	Sens.	Corr.	Sens.	Corr.	Sens.	Corr.	Sens.	Corr.
11 tRNA-PHE	45.2	0.49	71.4	0.75	71.4	0.81	54.78	0.54
5 RNAse P	61.3	0.58	65.6	0.63	64.9	0.79	31.95	0.32

Table 6.5: Comparison to the Gardner study. *RNAcast* uses shape abstraction π_5. The *Dynalign* RNAse P results may improve for a larger window size. For detailed *Carnac* and *Dynalign* parameter sets see [Gardner and Giegerich, 2004].

Comparison to the Gardner study

In [Gardner and Giegerich, 2004], several multiple RNA folding algorithms were evaluated. The study included three different approaches, where "Plan B" referred to tools that approximate the Sankoff approach of simultaneous alignment and folding. I chose the *S.cerevisiae* tRNA-PHE (11 sequences, high sequence similarity) and the *E.coli* RNAse P (5 sequences, medium similarity) data sets from that study and compared the prediction accuracies. Since *Dynalign* permits only pairwise folding, Gardner *et al.* folded the reference sequence with each of the other sequences at a time. I applied *RNAcast* in the same fashion. The corresponding results are in column "*RNAcast* pairwise" and "*Dynalign*" in Table 6.5. *Dynalign* performs better on the tRNA set, but *RNAcast* predicts better structures on the RNAse P set. *Carnac* [Touzet and Perriquet, 2004] can fold multiple sequences and performed quite well in the Gardner study. Naturally, my method yields much better results for a multiple sequence input than for only two sequences (see column "multiple"). The average correlation increases from 53.5% to 69%. The sensitivity is comparable to *Carnac* which in turn is almost perfectly selective and thus has a better correlation.

Comparison with *FoldalignM* study

Recently, a pairwise progressive multiple version of *Foldalign* has been introduced in [Torarinsson *et al.*, 2007]. The authors tested their algorithm and several others including *RNAcast* on a set of 17 Rfam families. Unfortunately, they tested *RNAcast* with its default parameters which sets the energy threshold R to 10% of the MFE. This threshold turned out to be too low for 7 families, where *RNAcast* was not able to predict a consensus structure. In order to allow for a fair comparison, I re-evaluated *RNAcast*'s accuracy with a higher threshold (10 kcal/mol) and obtained a consensus structure for all families. I also included *LocARNA* into the evaluation. Since it predicts a single consensus structure for all sequences, I had to map the structure back onto the individual sequences. Doing so, I removed all non-canonical base pairs, since the test set was constructed in the same manner. The results are shown in Table 6.6. *RNAcast* performs better than the combinations of *ClustalW+Pfold* and *ClustalW+RNAalifold*. Yet, *FoldalignM*, *CMfinder*, and *LocARNA* (in global, multiple mode), which are all computationally more expensive than *RNAcast*, perform even better on the test data set.

It should be noted that this evaluation uses a different measure of accuracy than the previous ones. Compatible base pairs are counted as false positives and thus reduce the accuracy. I believe that *RNAcast* suffers disproportionally from this difference. By algorithm construction, *RNAalifold*, *FoldalignM*, and *CMfinder* do not predict base pairs which could be folded only in the minority of the sequences. Those base pairs are usually the ones which are also not present in the reference consensus structure. In contrast, *RNAcast* employs energy minimization to obtain the shreps and thus tends to predict additional base pairs in accordance with the consensus shape which might not be conserved in other family members. In order to test this hypothesis, I re-ran the evaluation with my previous definition of false positives which does not count additional compatible base pairs. As expected, the accuracy rises from 0.74 to 0.81. Thus, it can be concluded that a significant amount of predicted base pairs do not contradict the consensus but are merely additions to it. I did not re-evaluate the other programs, so one has to keep in mind that their accuracy might also rise, but I expect their gain to be lower than *RNAcast*'s.

Overall, the evaluations show that *RNAcast* outclasses single sequence prediction methods and to a lesser extent, procedures which fold an independently computed sequence alignment. Depending on the test data set and the evaluation method, *RNAcast* has a comparable or slightly worse performance than algorithms implementing the Sankoff-style which in turn, are naturally a lot slower. Thus, the following directives may be given as a rule of thumb:

Family	#seq	FoldalignM	CMfinder	LocARNA	Clustal/ Pfold	Clustal/ Alifold	RNAcast
Entero CRE	56	0.75	0.77	0.81	0.95	0.71	0.77
Histone 3	63	1	1	1	1	1	1
IRE	29	0.83	0.92	0.95	0.67	0.64	0.71
Intron gpII	75	0.73	0.71	0.78	0.78	0.76	0.71
Lysine	48	0.60	0.77	0.76	0.59	0.22	0.74
Purine	29	0.89	0.90	0.74	0.76	0	0.77
RFN	47	0.73	0.73	0.80	0.72	0.77	0.62
Rhino CRE	12	0.82	0.96	0.69	0.66	0.77	0.73
SECIS	63	0.73	0.68	0.65	0	0	0.65
S box	64	0.73	0.81	0.81	0.77	0.72	0.71
tRNA-like	22	0.74	0.77	0.87	0.62	0.73	0.79
ctRNA pGA1	17	0.94	0.89	0.95	0.86	0.92	0.96
glmS	14	0.72	0.74	0.76	0.62	0.49	0.53
let-7	9	0.83	0.79	0.87	0.84	0.80	0.82
lin-4	9	0.78	0.76	0.84	0.72	0.73	0.78
mir-10	11	0.81	0.85	0.71	0.75	0.85	0.79
s2m	23	0.79	0.68	0.80	1	0.64	0.54
AVG	35	0.79	0.81	0.81	0.72	0.63	0.74

Table 6.6: Re-evaluation of *RNAcast* on several Rfam families with a higher energy threshold. The performance is much better than reported in [Torarinsson *et al.*, 2007]. Accuracy in this study is measured with the approximate correlation coefficient and counting of compatible base pairs as false positives. All values except those for *RNAcast* and *LocARNA* are from [Torarinsson *et al.*, 2007].

1. If a good sequence alignment is available (usually with a mean pairwise identity > 60–70%), use alignment folding.
2. Use *RNAcast* below 60-70% mean pairwise identity.
3. If running *RNAcast* does not yield a consensus shape, turn to a Sankoff-style algorithm.

6.5.4 Detailed analysis of mispredictions

In the preliminary study, it turned out that the true shape is detected in 53% of all pairwise predictions. Nevertheless, the MCC for pairwise predictions is over 70%. This raises the question of how bad the wrong shapes really are? By visual inspection, I could classify a few recurring situations, listed in Table 6.7.

Error classification	
additional hairpin predicted	3
multiloop enclosure	42
one hairpin missed	22
otherwise	41
total	107

Table 6.7: Recurring situations, for which *RNAcast* predicts the wrong shape. In general, prediction accuracy on the base pair level for the first three cases is still rather high.

In three cases, all tRNAs, *RNAcast* predicts an additional hairpin not mentioned in the database. This hairpin is the variable arm and therefore predicted correctly. On other sequences (42), I found that parts of the reference structure were enclosed by an additional helix, thus forming a multiloop. In my point of view, this situation cannot be classified as false, either. It further confirms my choice not to count *compatible* base pairs as false positives. Another point of error was the loss of one hairpin in 22 cases. Instead of the hairpin, there is either a single-stranded region, or the region is consumed by the prolongation of a neighboring helix. Nevertheless, the remaining structure is accurate. All remaining cases (41) differ substantially from the reference and have to be admitted as wrong predictions without an excuse.

In general, the accuracy for the first three situations is still rather high on the base pair level. In fact, it is higher than single sequence predictions. Usually, sequences for which *RNAcast* predicts a wrong shape have a low accuracy and are poorly predicted by *RNAfold* as well.

6.5.5 Efficiency

Let me now recapitulate *RNAcast*'s efficiency.

The enumeration phase runs in $O(k \cdot n^3 + |\mathcal{P}(s_1)| + \cdots + |\mathcal{P}(s_k)|)$. Step 2 runs in time proportional to $(k \cdot n \cdot |\mathcal{P}(s_1)|)$, since the algorithm needs to iterate only over the list $\mathcal{P}(s_1)$ and uses the hash table to look for co-occurrences in the other shape spaces $\mathcal{P}(s_2), \ldots, \mathcal{P}(s_k)$. Finally, scoring and sorting of candidates can be done in $O(k \cdot l + l * \log l)$ time where l is the number of common shapes. In general, the $\mathcal{P}(s_i)$ are rather small and the first step dominates the computation time.

As is to be expected from the asymptotic analysis, the efficiency of *RNAcast* is quite good. It strongly depends on *RNAshapes*' efficiency which has recently been optimized [Steffen *et al.*, 2006]. For reason of completeness, I also report the values for the old versions

Original measurements published in [Reeder and Giegerich, 2005]:

	tRNA 72 and 75		U2 RNA 188		RNAse P 245 and 248	
Program (Parameters)	Time	Memory	Time	Memory	Time	Memory
RNAcast ($\pi_5, R = 10$)	0.6	19	4.5	29	22.5	80
RNAcast ($\pi_3, R = 10$)	0.8	20	5.0	31	58.0	257
Dynalign (M=15)	488	20	7631	92	12718	141

Measurements using the current versions as of 2007:

RNAcast ($\pi_5, R = 5$)	0.1	1	0.5	16	1.3	18
RNAcast ($\pi_3, R = 5$)	0.1	3	0.5	16	1.5	20
RNAcast ($\pi_5, R = 10$)	0.2	13	1.2	16	5.0	24
RNAcast ($\pi_3, R = 10$)	0.2	13	1.5	19	25.5	143
LocARNA ($p^* = 0.01$)	0.3	2	2.7	3	14.9	3
LocARNA ($p^* = 0.001$)	0.5	2	4.8	3	73.0	4
Dynalign	5.7	10	54	17	127.1	23
Foldalign	2.5	6	66	33	164.5	62

Table 6.8: Time (in seconds) and memory (in MB) requirements for three pairwise predictions, measured on a 2.8 GHz Dual Xeon system with 2 GB RAM. Depending on the actual parameter settings, *RNAcast* is faster than the Sankoff-style programs, except for the case of π_3 and a high R and *LocARNA* running with a low significance threshold.

of pairwise *RNAcast* and *Dynalign* in Table 6.8 which were used in the benchmark of the original publication [Reeder and Giegerich, 2005], on which this chapter is based. While at the time of publication, the difference in runtime between *RNAcast* and *Dynalign* (and other Sankoff-like algorithms not benchmarked back then) was quite tremendous, the recent improvements for Sankoff-style alignment programs (especially the heuristic employed in *LocARNA*) reduce this gap considerably. Still, the consensus shapes approach is faster, however, depending on the sequence length and the actual choice of shape abstraction level and energy range. Only *LocARNA* with its default parameters (which might be too optimistic in some cases) is faster than *RNAcast* (with conservative settings) on the RNAse P sequence pair. Note that for long sequences and a high energy threshold, the memory requirements of *RNAcast* are larger than those of the other programs. However, most of the memory is already required by *RNAshapes* for the enumeration of the shape spaces. The actual consensus analysis adds only a little overhead.

Of course, the true strength of *RNAcast* lies in its linear runtime dependence on the

	5 U12 RNA	11 Sbox	29 Purine	75 Intron gpII
RNAcast (π_5)	2.0	2.4	4.9	8.9
RNAcast (π_3)	3.4	3.4	7.0	15.3
LocARNA	10.1	21.0	92.4	242.1
FoldalignM	355.4	482.8	2140.5	9297.2

Table 6.9: Runtime in seconds measured for predictions of multiple RNA sequences. With increasing number of sequences, *RNAcast*'s relative speed advantage becomes more obvious. For *RNAcast*, R was set to 10; the other programs used their default values.

number of sequences. Therefore, I also benchmarked it against *LocARNA* and *FoldalignM* in a multiple fashion. The results are shown in Table 6.9. The runtime of *FoldalignM* is at least a factor of 100 larger than my method and *LocARNA*'s around a factor of five to more than ten. Obviously, the latter program's heuristic is much more effective than the first one's. Nevertheless, both *FoldalignM* and *LocARNA* suffer from their quadratic dependence on the number of input sequences, witnessed by the increasing relative difference to *RNAcast*'s runtime.

6.6 Discussion

6.6.1 Differences to the Sankoff notion of consensus

Let me once more relate *RNAcast* to *Dynalign* which is one of the best available approximations to the Sankoff algorithm. It is important to keep in mind that while the Sankoff algorithm can, in principle, maximize sequence similarity alongside with free energy minimization, its *Dynalign* implementation minimizes gap penalties, but otherwise ignores sequence content.

The quantitative results in the previous section show that the consensus shape method is comparable or better in the quality of predictions and much faster computationally. In that section, results from both tools were compared to a "gold standard" which is much easier than comparing them to each other, because they pursue different objectives.

Remember that I have not presented another approach to implement the Sankoff algorithm, but I have significantly changed the problem definition: While the Sankoff approach determines a sequence alignment reflecting a common set of base pairs, consensus shape prediction produces a consensus abstract shape together with its shrep for each sequence, but no alignment. Since this deviates from the traditional and accepted notion, let us discuss common aspects as well as differences from a conceptual point of view.

Dynalign produces sequences aligned according to the predicted common base pairs, hence with the same Level 5 shape. However, their Level 3 shapes may be different, as some sequences may have gaps where others have bulges. In either case, the structures reported are not necessarily the shreps of their respective shapes. One may refold the structures individually with the consensus base pairs fixed, but then, the refolded structures may be "out of shape", because they exhibit additional hairpins.

RNAcast predictions are unaligned. Using the predicted shreps, a multiple structure alignment may be obtained via *RNAforester* or similar structure alignment tools. From the structure alignment, a sequence alignment, consistent with the consensus shape, may be easily derived. The structure alignment also minimizes the number of gaps, but in contrast to the Sankoff approach, it does so *after* structure prediction and not simultaneously. Hence, one may expect cases where the Sankoff approach produces results that fix the relative positions of helices more strongly, while with *RNAcast*, conserved helices may move more flexibly. However, I have not observed this effect to a significant amount in my studies. I demonstrate the combination of *RNAcast* and *RNAforester* in the next section.

6.7 Alignments of consensus structures

The evaluations performed so far clearly show that the consensus shape approach reliably predicts most of the annotated base pairs. In order to mimic the usual outcome of Sankoff-style algorithms, *RNAcast*'s unaligned consensus structure prediction can be aligned with the structural alignment tool *RNAforester* [Höchsmann *et al.*, 2004]. A benchmark suite for such alignments of structural RNAs is the BRAliBase [Gardner *et al.*, 2005]. It provides pairwise and multiple alignments of RNAs with mean pairwise sequence identity ranging from 10 – 100 %. Performance of an algorithm to be tested on BRAliBase is measured in terms of sum-of-pairs-score (SPS) and structure conservation index (SCI). The SPS measures the amount of correctly aligned residues, while the SCI measures the amount of conserved structural information in the alignment. Specifically, the SCI is defined as the ratio of the alignment-folding score (computed by *RNAalifold*) and the average *RNAfold*-MFE of the sequences contained in the alignment. Dataset II of BRAliBase contains over one hundred pairwise tRNA alignments and was used in [Gardner *et al.*, 2005] to assess the performance of several (structural) alignment tools.

I designed a simple pipeline (called *RNACaFo*) which redirects *RNAcast*'s output into *RNAforester* and evaluated its performance on the data set II[3]. The outcome is shown in Figures 6.5 and 6.6 (dotted red line). In the moderate to high homology region, the use of

[3] Joint work with Alexander Hosfeld, a diploma student under my supervision.

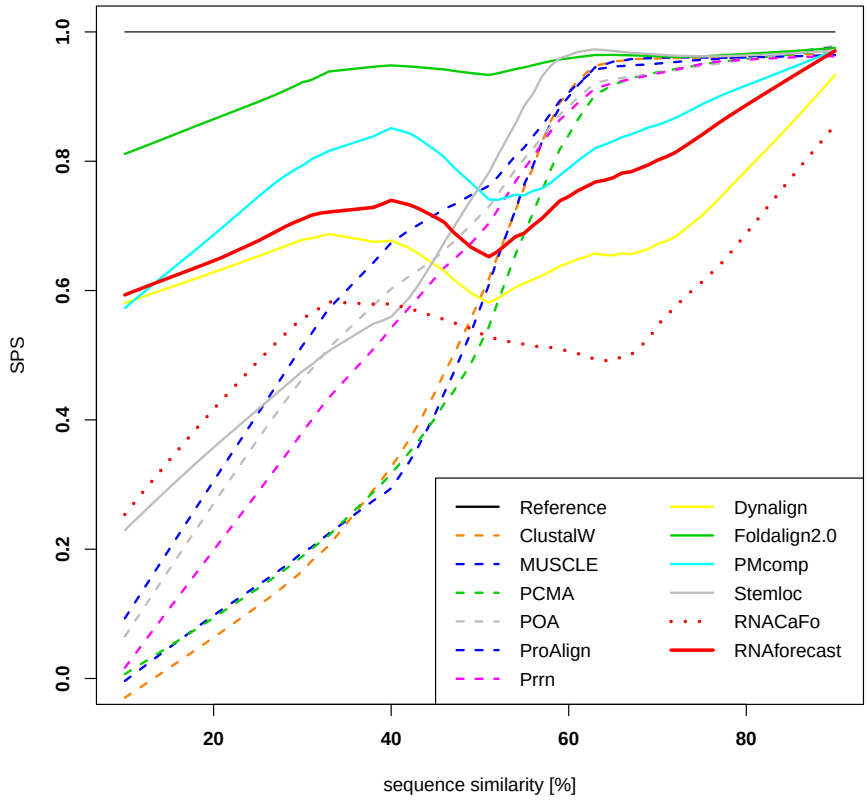


Figure 6.5: SPS curves for the programs in the BRAliBase study augmented with curves for *RNACaFo* (dotted red line) and *RNAforecast* (solid red line). While the former fails to clearly separate from the pure sequence alignment programs, the latter outperforms even *Dynalign*. In this study, α was set to 0.5, and shape abstraction π_5 was used.

structural alignment programs is dispensable, since pure sequence alignments are usually good enough. In the low homology region, *RNACaFo* shows little improvement in terms of SPS over *ProAlign*, the best pure sequence alignment tool. Considering the SCI, the situation looks more promising. Here, a clear separation of *RNACaFo* and the sequence alignment tools is apparent. However, the accuracy of Sankoff-style approaches is not reached. The significant difference of *RNACaFo*'s accuracy in terms of SCI compared to SPS suggests that the alignments are more accurate in the helical parts. A misalignment of conserved base pairs may prevent *RNAalifold* from assigning a particular base pair and thus, results in a lower SCI score. On the other hand, misaligned loop regions only have a minor influence on the SCI, but contribute on equal terms to the SPS as helical regions.

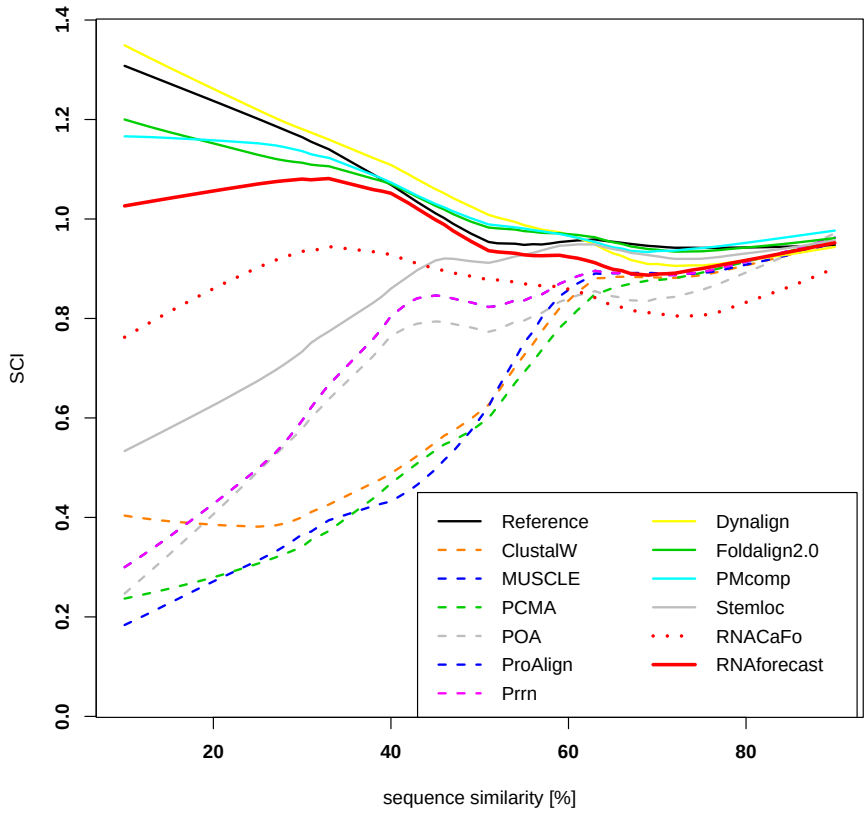


Figure 6.6: SCI curves from the BRAliBase benchmark. Both *RNACaFo* and *RNAforecast* surpass all pure sequence alignment tools in the interesting low homology region. The latter even approaches the performance of the Sankoff-style alignment tools.

6.7.1 *RNAforecast*

Motivated by the results of the simple pipeline, the idea of an improved pipeline, called *RNAforecast*, was born. It uses the information obtained from the structural alignments to select better consensus shapes from the list of common shapes. The idea is formalized by a further rank function:

5. *RNAforecast* score:

   A linear combination of *RNAcast* and relative *RNAforester* score, where the α balances the influence of each score.

$$rank_5(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i) = \alpha \cdot \frac{\frac{rank_2(p_i, \hat{p}_1^i, \ldots, \hat{p}_k^i)}{k}}{\sum_{j=1} E(mfe(s_j))} + (1-\alpha) \cdot forester\_score(\hat{p}_1^i, \ldots, \hat{p}_k^i)$$

The first operand (multiplied by α) is the $rank_2$ function defined earlier, normalized with the sum of MFEs, thus a value from the interval $[0 \ldots 1]$. (One could as well as use $rank_3$ instead of $rank_2$ and change the denominator to k for rescaling to one.) The second operand (weighted by $(1 - \alpha)$) is the relative *RNAforester* score which is upper-bounded by 1 but has no lower bound. For details on the scoring function and definition of the relative score see [Höchsmann *et al.*, 2004].

Of course, computing a multiple structural alignment for all common shapes adds a considerable amount of computation time. Computing a single alignment of k structures with maximal sequence length n requires $O(n^2 d^2 k^2)$ time, where d is the maximum degree of a tree node. (Structures are handled as trees/forests in the model underlying *RNAforester*.) Particularly note that *RNAforecast*'s runtime does not scale linearly with the number of sequences as *RNAcast*'s, but quadratically.

Thus in practice, I restrict the alignment procedure to only the best, say ten, consensus shapes, according to the *RNAcast* score. To give some real numbers: Consensus shape prediction of five U2 RNAs (∼190 nucleotides) takes five seconds with *RNAcast*, while the *RNAforecast* prediction takes about 100 seconds. Still, this is much faster than e.g. *FoldalignM*, which takes over ten minutes on this particular data set, but in the same range as *LocARNA*. For shorter sequences, say five typical tRNAs, the difference lies in 0.3 seconds for *RNAcast* versus 36 seconds for *RNAforecast*.

This raises the question whether the extension is justified by improved predictions. Figures 6.5 and 6.6 (solid red line) give the answer. Both, SPS and SCI, values are lifted to the level of Sankoff-style alignment programs. Again, the SCI is significantly higher than SPS. Remember that the BRAliBase data set II consists of only pairwise alignments, but *RNAcast* and *RNAforecast* can handle multiple sequences. It would be worthwhile to set up a new test set or use the BRAliBase data set I and to evaluate the alignment performance of the multiple *Foldalign* version, *CMfinder*, *LocARNA*, and *RNAforecast*.

The *RNAforecast* pipeline is conceptually similar to the *MARNA* approach [Siebert and Backofen, 2005], but not identical for the following reason: *MARNA* independently predicts, for each sequence, a small (hopefully representative) set of structures. It uses either randomly sampled RNAs, generated by the probabilistic backtrace procedure of *RNAsubopt*, or *RNAshapes*, as in my approach. However, in the next step, they perform an all-against-all pairwise structural alignment and use the resulting scores as basis for a multiple alignment with the *T-Coffee* system. Assuming k sequences of length n and E structures *per* sequence, this approach requires $O(E^2 \cdot k^2)$ structural alignments, each taking $O(n^4)$ time. Therefore, the number of structures per sequence E is set to a rather small constant, say three, to keep the runtime practical.

However, in the extended consensus shape approach, the expensive structural alignments are limited to structures exhibiting the same shape. Since we consider, for each sequence, only one structure per shape, *RNAforecast* requires a factor of E less structural comparisons. Hence, this allows us to look deeper into the suboptimal shape (structure) space.

Note that another difference lies within the underlying model for structure comparison. While *RNAforester* aligns trees, *MARNA* aligns arc-annotated sequences. Of course, I could as well have used the latter for the pipeline.

6.8 Amalgamating *pknotsRG* and *RNAcast*

The original *RNAshapes* implementation considers only properly nested secondary structure in its folding space. However, the technique of abstract shape analysis can (at least conceptually) easily be taught to any DP folding routine. To demonstrate this fact, I implemented a prototype algorithm which employs abstract shape analysis for canonical simple recursive pseudoknots. The algorithm uses an extended definition of the previously introduced structure-to-shape mapping. Pseudoknots are indicated by square brackets for the first stem (as regular helices in the previous shape notation) and curly brackets for the second stem. Unpaired loop regions in pseudoknots are handled in the same way as internal loops – they will appear only in shape Level 2 and lower. For a precise definition of π_5 and π_3 see Table 6.10.

At the present time, the prototype implementation is, in practice, a lot slower than the original *pknotsRG* for two reasons. First, it cannot be compiled with the ADP compiler, since it requires the use of a classifying shape algebra in addition to the regular energy and prettyprint algebras. Hence, it relies on the more powerful, but slower Haskell embedding of ADP. Second, it does not employ the efficient backtracing procedure of *RNAshapes* (for details see [Steffen, 2006], Chapter 5.1), but rather applies shape abstraction already in the matrix fill stage. This is why the asymptotic runtime lies in $O(n^4 \cdot |\mathcal{P}_R|)$ (with $|\mathcal{P}_R|$ being the number of suboptimal shapes residing within the chosen energy range R) rather than in $O(n^4 + |\mathcal{P}_R|)$, achievable with a separation of matrix fill and shape abstraction stage. To give some concrete numbers: Computing all shapes within 10 kcal/mol of the MFE for a sequence of length 100 nucleotides takes about 40 minutes. Reducing the energy threshold to 5 kcal/mol, reduces runtime to 140 seconds. Note that the higher runtime does not only stem from the less efficient implementation, but also from the drastic increase in the size of the shape space when pseudoknots are considered. While a random sequence of 100 bases has 41 shapes within its 10 kcal/mol suboptimal *nested* shape space, the same

$\pi_5(\mathbf{.})$	$= \varepsilon$	$\varrho_5(\mathbf{.})$	$= \varepsilon$
$\pi_5(\mathbf{.}s)$	$= \pi_5(s)$	$\varrho_5(\mathbf{.}s)$	$= \varrho_5(s)$
$\pi_5(s\mathbf{.})$	$= \pi_5(s)$	$\varrho_5(s\mathbf{.})$	$= \varrho_5(s)$
$\pi_5(\mathbf{(}s\mathbf{)})$	$= [\varrho_5(s)]$	$\varrho_5(\mathbf{(}s\mathbf{)})$	$= \varrho_5(s)$
$\pi_5(\mathbf{(}s\mathbf{)}s')$	$= [\varrho_5(s)]\pi_5(s')$	$\varrho_5(\mathbf{(}s\mathbf{)}s')$	$= \pi_5(\mathbf{(}s\mathbf{)}s')$
$\pi_5([^k s\{^l s']^k s''\}^l)$	$= [\pi_5(s)\{\pi_5(s')]\pi_5(s'')\}$	$\varrho_5([^k s\{^l s']^k s''\}^l)$	$= [\pi_5(s)\{\pi_5(s')]\pi_5(s'')\}$

$\pi_3(\mathbf{.})$	$= \varepsilon$	$\varrho_3(\mathbf{(}s\mathbf{)})$	$= \varrho_3(s)$
$\pi_3(\mathbf{.}s)$	$= \pi_3(s)$	$\varrho_3(s)$	$= \pi_3(s)$ *in all other cases*
$\pi_3(s\mathbf{.})$	$= \pi_3(s)$		
$\pi_3(\mathbf{(}s\mathbf{)})$	$= [\varrho_3(s)]$		
$\pi_3(\mathbf{(}s\mathbf{)}s')$	$= [\varrho_3(s)]\pi_3(s')$		
$\pi_3([^k s\{^l s']^k s''\}^l)$	$= [\pi_3(s)\{\pi_3(s')]\pi_3(s'')\}$		

Table 6.10: Definition of level-5 (π_5) and level-3 (π_3) shape abstractions including rules for canonical simple recursive pseudoknots.

sequence can fold into 541 different possibly *pseudoknotted* shapes.

Nevertheless, this first version of *pknotsRG-shape* allows us to perform single sequence shape analysis for moderate size sequences and of course, also consensus shape analysis. The techniques described earlier in this chapter can directly be applied on the pseudoknot containing shape space. *RNAcast* can easily be modified to use *pknotsRG-shape* for the enumeration of the shape spaces in step 1. Step 2 and step 3 require no modifications at all. However, since *RNAforester* is unable to align pseudoknotted structures, the *RNAforecast* approach cannot be used with pseudoknots.

To my knowledge, this is the first comparative pseudoknot folding algorithm which has no prerequisites other than the pure sequences. Previous comparative pseudoknot folding algorithms [Witwer *et al.*, 2004; Ruan *et al.*, 2004] require an initial sequence alignment and thus, inherit all the problems of programs such as *RNAalifold* with misalignments of low homology sequences.

I did not perform a thorough evaluation of this new approach, since there are only very few RNA families in Rfam and other databases which contain pseudoknots and are short enough to be folded with *pknotsRG-shape*. Instead, I demonstrate its applicability by comparative folding of the ribosomal frameshift inducing element conserved in the coronavirus family (Rfam ID: RF00507). While *pknotsRG* fails to predict the correct topology for six out of eleven family members, the pseudoknot consensus method succeeds

in finding it. The corresponding structures are given in Figure 6.7.

Considering the fact that there are many uncertainties in the pseudoknot energy model, which poses an implicit disturbance in pseudoknot prediction from single sequences, I am convinced that this method will be a leap forward in the detection of pseudoknots. This, of course, assumes that the here described method will be re-implemented (semi-automatically using the ADP compiler) in the most-efficient manner, employing the tricks mentioned earlier with *RNAshapes*.

6.9 Potential improvements

Reality differs from my evaluation scenario. Database families can be considered reliable homologues, but when a new (putative) family is investigated, one cannot be sure whether structure is preserved. With consensus shape prediction, I would like to implement a safeguard against members in the sequence set that really do not share the common shape with the rest. Such a situation will most likely result in consensus garbage. In my opinion, leave-one-out tests can be designed to recognize this situation. Such tests can be implemented efficiently, because only steps 2 and 3 of the *RNAcast* algorithm, but not the most costly step 1 must be iterated.

It should be kept in mind that the calls to *RNAshapes* are independent of each other and thus can, in principle, be executed in parallel. Depending on the number of sequences, this will reduce the runtime considerably. However, this feature is currently not implemented.

Another point of disturbance arises, if the sequences under evaluation are, in fact, members of an RNA family but are flanked by non-conserved sequence parts. I am convinced, that my approach is relatively robust, as long as the conserved RNA sequences are flanked by only a few bases. However, with larger, unrelated flanking sequences, the outcome will usually not be correct. For such situations, one would like to have an adaptive window approach which automatically chooses the interesting region for each sequence. Working with an adaptive window size is a current research problem in RNA gene prediction. My approach will certainly benefit from advances in this direction.

I have performed an overall evaluation of the new method but have not tried to optimally adjust it to particular data sets. For example, when studying short molecules like microRNA precursors, Level 2 abstraction, which distinguishes 5'-, 3'- bulges and internal loops, might be more conclusive than Level 3. More systematic study and experience is needed to provide guidance about the most conclusive level of shape abstraction to be used in a particular context.

```
Shape:               [{][]}     Score: -381.815827      relative Score: 0.97
> L22089.1/6839-6925 RF00507; Corona_FSE;
CGCUUCCAAAUUCCAGACCCAACCGGGUCUGGAAUUGGGCUCUGCAAUUGGGUGUGACCCAGACGUGCAUUGGACAGCCUUUGGUGU
........[[[[[[[[[[[[.{{{]]]]]]]]]]]].((((.((((.(((((.....)))))...))))......)))))...}}}.. (-38.40) R = 1
> M22457.1/6904-6990 RF00507; Corona_FSE;
UGCAUCGAAGUUCCAGACCCAACCGGGUCUGGAGCUUGGUUCUGCUAUUGGAUGUGAUCCUGAUGUUCACUGGACCGCUUUUGGUGU
.......[[[[[[[[[[[[[.{{{]]]]]]]]]]]]](((((.((....((((...))))....)).....)))))......}}}.. (-35.60) R = 2
> X74312.1/6937-7023 RF00507; Corona_FSE;
UGCAUCAAAGUUCCAGACCCAACCGGGUCUGGAACUAGGAUCAGCCAUUGGAUGUGACCCAGAUGUACACUGGACUGCCUUCGGUGU
........[[[[[[[[[[[[.{{{]]]]]]]]]]]].((..((((((.((.((((.......)))).)).)))).))))...}}}.. (-34.40) R = 23
> AF029248.1/13602-13686 RF00507; Corona_FSE;
GGGUUCGGGGUACAAGUGUAAAUGCCCGUCUUGUACCCUGUGCCAGUGGCUUGGACACUGAUGUUCAAUUAAGGGCAUUUGACAU
.....[[[[[[[[[[..{{{{{{{{{{..]]]]]]]]]]...(((((.......)))))............}}}}}}}}}.... (-35.30) R = 2
> AF391541.1/13342-13426 RF00507; Corona_FSE;
GGGUUCGGGGUACGAGUGUAGAUGCCCGUCUCGUACCCUGUGCCAGUGGUUUAUCUACUGAUGUACAAUUAAGGGCAUUUGAUAU
......[[[[[[[[[[..{{{{{{{{{{..]]]]]]]]]](((((((((((.....))))))..))))......}}}}}}}}}.... (-39.30) R = 1
> AY613949.1/13349-13430 RF00507; Corona_FSE;
GGGUUUGCGGUGUAAGUGCAGCCCGUCUUACACCGUGCGGCACAGGCACUAGUACUGAUGUCGUCUACAGGGCUUUUGAUAU
......[[[[[[[[[[...{{{{{..]]]]]]]]]](((((((((((....)).))).))))))).....}}}}}........ (-31.60) R = 1
> AY338732.1/12302-12383 RF00507; Corona_FSE;
GGGUACGGGGUAGCAGUGAGGCUCGGCUGAUACCCCUUGCUAAUGGAUGUGAUCCUGAUGUUGUAAAGCGAGCCUUUGAUGU
......[[[[[[.....{{{{{{{{{.....]]]]]].(((....((((...))))).......)))...}}}}}}}}..... (-25.10) R = 1
> AJ271965.2/12339-12417 RF00507; Corona_FSE;
GAGUGCGGGGUUCUAGUGCAGCUCGACUAGAACCCUGCAAUGGUACUGAUCCAGACCAUGUUAGUAGAGCUUUUGACAU
...[[[[[[[[[[[[[..{{{{{.]]]]]]]]]]]]](((((.(((...)))))))).......}}}}}........ (-36.40) R = 1
> X69721.1/12521-12599 RF00507; Corona_FSE;
GAGUCCGGGGCUCUAGUGCCGCUCGACUAGAGCCCUGUAAUGGUACAGACAUAGAUUACUGUGUCCGUGCAUUUGACGU
.....[[[[[[[[[[[[......{{]]]]]]]]]]]].....((((.(((((((....)))))))).))))......}}. (-35.40) R = 4
> AY518894.1/12426-12504 RF00507; Corona_FSE;
GAGCAAGGGGUUCUAGUGCAGCUCGACUAGAACCCUGUAAUGGCACGGACAUCGAUAAGUGUGUUCGUGCUUUUGACAU
......[[[[[[[[[[[.....{{{]]]]]]]]]]].....((((((((((.........))))))))))..}}}... (-31.60) R = 5
> AF353511.1/12621-12699 RF00507; Corona_FSE;
GAGUACGGGGCUCUAGUGCAGCUCGACUAGAGCCCUGUAAUGGUACUGAUACACAACAUGUGUAUCGUGCUUUUGACAU
...[[[[[[[[[[[[[.....{{{]]]]]]]]]]]]..(((((.(((((((.....)))))))))))..}}}... (-38.00) R = 2
```

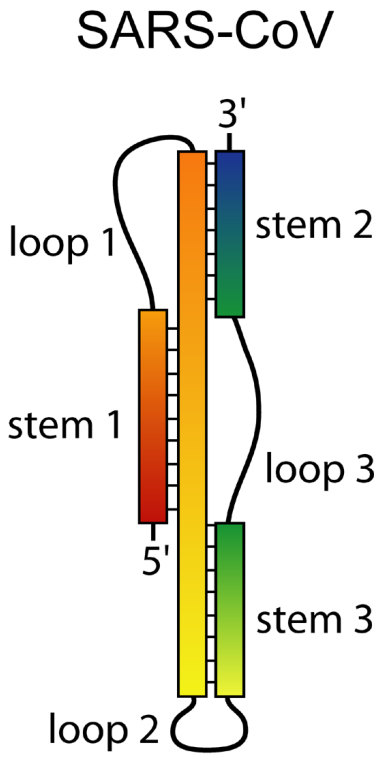


Figure 6.7: Consensus shape prediction for the coronavirus frameshifting stimulation element. The three-stem pseudoknot topology (schematic drawing on the right side, reproduced from [Staple and Butcher, 2005]) is predicted for all sequences. Note that for some sequences, this shape is not ranked on position 1, i.e. it will not be predicted by *pknotsRG* as MFE.

Structural alignments of common shapes can improve the quality of predictions, as shown with the *RNAforecast* pipeline. However, this comes with an increase in asymptotic time efficiency. Since *RNAforester* performs a pairwise all-against-all structural alignment during the guide tree construction, the linear dependence on the number of input sequences is lost. Therefore, improvements in this step will have a major impact on the overall computation time. Here, I propose two alternative methods for the alignment step in the *RNAforecast* pipeline: First, a faster "string-like-alignment" method can be used for the guide tree construction, solely. The full structural alignment would then be performed only for the progressive alignments along the tree. A similar approach has been suggested in [Hofacker *et al.*, 2004], where a fast (sequence) alignment of probability profiles is used as a substitute for the expensive base pair matrix alignment. The authors claim that the quality of the fast alignments is sufficient to construct the guide tree. Second, I propose to restrict the structural alignments to those parts of the structures which correspond to the same part in the shape string. To give an example, let us assume we are aligning two structures both exhibiting the two-hairpin-shape `[][]`. Then, we can effectively divide both sequences somewhere between the 3'-most paired base of the left hairpin and the 5'-most paired base of the right hairpin. The two smaller subsequence pairs can now be aligned independently. Depending on the number of cut-points and how balanced the subsequence lengths are, we can expect a significant reduction in runtime.

The Sankoff algorithm and its heuristic implementations principally score both sequence/structure conservation and free energy. However, the actual choices taken in current implementations differ. *Dynalign* minimizes the combined score of energy and a (positive) penalty for gaps. *Foldalign* maximizes a combined score of sequence similarity (using a substitution matrix similar to RIBOSUM [Klein and Eddy, 2003]) and free energy values multiplied by -10. *LocARNA* and *PMmulti* score sequence conservation and base pair probabilities which of course, are derived from the free energy model. To my knowledge, no exhaustive study has been conducted which aims on finding the optimal combination or weighting of the two components. I am confident that *RNAforecast*, as well as the above mentioned programs, will benefit from a such a study.

Chapter 7

Conclusion

In this thesis, I presented the algorithm *pknotsRG-mfe*, based on the MFE-model, for finding the best RNA structure including the pseudoknot class csr-PK. The algorithm achieves time complexity $O(n^4)$ and space complexity $O(n^2)$. The runtime improvement, compared to *pknotsRE*, results from the idea of canonization, while the space improvement results from disallowing chained pseudoknots. The algorithm variant *pknotsRG-enf* returns the energetically best structure that contains a pseudoknot (interesting when the MFE structure is unknotted), while *pknotsRG-loc* reports the best pseudoknot (under a length-normalized energy score) *somewhere* in a sequence. It achieves a high prediction accuracy for moderate length sequences, whereas long sequences, at least when pseudoknots are involved, seem to have a folding scheme that cannot be modeled with minimum free energy folding. *pknotsRG* has been successfully applied in several biological publications, e.g. [Roberts *et al.*, 2004; Zeenko and Gallie, 2005; Dmitriev *et al.*, 2007; Hansen *et al.*, 2007; Nibert, 2007]. Furthermore, it has been incorporated into various computational pipelines [Huang *et al.*, 2005; Moon *et al.*, 2007; Taufer *et al.*, 2007].

The algorithm *pknotsRG* is based on a simpler grammar model than the crossed interaction grammars [Rivas and Eddy, 2000] underlying *pknotsRE*, as well as the communicating grammars underlying the approach by Cai *et al.* [2003]. It requires only a minor extension over the ADP tree grammars which are applicable to a wide range of sequence analysis problems [Giegerich *et al.*, 2004a]. Furthermore, the grammar is not only a theoretical backup explaining the underlying model. With minor annotation for the sake of efficiency, the grammar actually constitutes executable code. This means that *pknotsRG* can serve as a template for a new class of programs which I call thermodynamic matchers. I demonstrated this fact by showing the development and application of a thermodynamic matcher for the detection of -1 ribosomal frameshift signals. Thermodynamic matchers, which can include the class of pseudoknot defined in this thesis, can easily be defined by

drawing them with the interactive *Locomotif* system [Reeder *et al.*, 2007].

The current state-of-the-art RNA homology search tool, *Infernal* [Nawrocki and Eddy, 2007], ignores pseudoknot base pairs and thus, looses their covariance information. In some cases, this may lead to misclassifications, especially if the amount of pseudoknotted bases is rather high. The program *pknotsRG* can help researchers in the future to identify known, pseudoknot containing RNA families in newly sequenced genomes. Family specific thermodynamic matchers can be defined, either automatically from the consensus structure annotation or semi-automatically using the *Locomotif* system. In conjunction with *Infernal* or even stand-alone, a matcher can give a strong hint at the thermodynamic stability of a sequence forced into the family consensus structure.

A significant speedup for comparative RNA folding algorithms can be achieved by using heuristics relying on base pair probabilities. In this thesis, this method is employed successfully for the first time in the context of pseudoknot prediction. The sparse version of *pknotsRG* achieves an asymptotic runtime of around $O(n^3)$ (with a significance threshold of $p^* = 1 \cdot 10^{-8}$) – the same asymptotic efficiency class as regular folding of nested RNA secondary structures.

In the second part of my thesis, I presented a new method for the consensus prediction of multiple RNA sequences. It is based on the rationale that RNA molecules exhibiting the same function often have a similar overall shape or structure. Building on the abstract shapes approach, I devised a fast method which independently enumerates, for each sequence, the suboptimal shape space and quickly identifies shapes common to all input sequences. I showed that this method performs superior to single sequence prediction and alignment folding programs. In comparison to (heuristic) algorithms implementing the much more expensive Sankoff approach, the accuracy in terms of correctly predicted base pairs is similar. In general, *RNAcast* is much faster than other comparative prediction algorithms. Its main advantage is its linear dependence in runtime on the number of input sequences. This allows us to use the full comparative power hidden in a set of related sequences.

In order to strengthen *RNAcast*'s predictions, I designed a pipeline, *RNAforecast*, which employs the score of the structural *RNAforester* alignment in combination with the original score for consensus shape ranking. This method shows a good performance in the low-homology region below 60 % pairwise sequence identity – an area previously only accessible with the much more expensive Sankoff implementations.

Finally, I implemented a functional prototype of a comparative pseudoknot prediction algorithm by amalgamating the two independent approaches of this thesis. Assuming that

a dramatically faster version of *pknotsRG-shape* will be available soon (which is straightforward, but time consuming to implement), this method will have a definite impact on the prediction of RNA pseudoknots.

Appendix

Nussinov78

The ADP implementation of Nussinov's base pair maximization algorithm as explained in section 3.2.1.

```
-- Haskell header  --

import ADPCombinators  -- download at
      -- http://bibiserv.techfak.uni-bielefeld.de/adp/src/ADPCombinators.lhs
import Array
import List

-- The signature  --

data M  =  Nil ()
        |  Unpaired      M Char
        |  Pair     Char M Char
        |  Split    M    M
                      deriving (Eq, Show)


-- Algebra type  --

type Algebra alphabet answer = (
  ()       -> answer,
  answer   -> alphabet -> answer,
  alphabet -> answer -> alphabet -> answer,
  answer   -> answer -> answer,
  [answer] -> [answer]
  )


-- Enumeration algebra  --

enum :: Algebra Char M
enum = (nil, unpaired, pair, split, h) where
  nil      = Nil
  unpaired = Unpaired
  pair     = Pair
  split    = Split
  h        = id
```

```
-- Counting algebra  --

count :: Algebra Char Int
count = (nil, unpaired, pair, split, h) where
  nil a        = 1
  unpaired a b = a
  pair a b c   = b
  split a b    = a * b
  h []         = []
  h xs         = [sum xs]


-- Base pair maximization algebra  --

bpmax :: Algebra Char Int
bpmax = (nil, unpaired, pair, split, h) where
  nil ()       = 0
  unpaired a b = a
  pair   a b c = b + 1
  split  a b   = a + b
  h []         = []
  h xs         = [maximum xs]


-- Pretty printing algebra  --

pp :: Algebra Char String
pp = (nil, unpaired, pair, split, h) where
  nil ()       = ""
  unpaired a b =      a ++ "."
  pair   b c d = '(': c ++ ")"
  split    a b =      a ++ b
  h            = id


-- Algebra product operation  --

infix ***
(***) :: Eq answer1 =
   (Algebra Char answer1) -> (Algebra Char answer2) ->
    Algebra Char (answer1,                answer2)

alg1 *** alg2 = (nil, unpaired, pair, split, h) where
  (nil1, unpaired1, pair1, split1, h1) = alg1
  (nil2, unpaired2, pair2, split2, h2) = alg2

  nil a                 = (nil1 a, nil2 a)
  unpaired (a1,a2) b    = (unpaired1 a1 b, unpaired2 a2 b)
  pair a (b1,b2) c      = (pair1 a b1 c, pair2 a b2 c)
  split (a1,a2) (b1,b2) = (split1 a1 b1, split2 a2 b2)

  h xs = [(x1,x2)| x1 <- nub $ h1 [ y1 | (y1,y2) <- xs],
                   x2 <-       h2 [ y2 | (y1,y2) <- xs, y1 == x1]]
```

```
-- The yield grammar --

nussinov78 alg f = axiom m where
  (nil, unpaired, pair, split, h) = alg

  m    = tabulated(
         nil      <<< empty                                          |||
         unpaired <<< m  ~~- base                                    |||
         split    <<< m  ~~! (pair <<< base -~~ m ~~- base )
                                          `with` basepairing ... h)
       where
       infixl 7 ~~!
       (~~!) = (*~*) 0 3

Bind input:

  z         = mk f
  (_,n)     = bounds z
  base      = achar' z
  axiom     = axiom' n
  tabulated = table n

  basepairing :: Filter
  basepairing  = match z
  match  inp (i,j) = i+1<j && basepair (z!(i+1), z!(j))
  basepair ('a','u') = True
  basepair ('u','a') = True
  basepair ('c','g') = True
  basepair ('g','c') = True
  basepair ('g','u') = True
  basepair ('u','g') = True
  basepair ( x , y ) = False
```

pknotsRG

This code shows the complete yield grammar underlying the algorithm *pknotsRG*. The start symbol (axiom) is `struct`.

```
struct      = listed (
               sadd <<< base    -~~ struct |||
               cadd <<< dangle' ~~~ struct |||
               nil  <<< empty'             ... h_l)

dangle'     =  dangle ||| dangleknot ... h
dangle      =  edl  <<< base -~~ closed ~~. loc  |||
               edr  <<<  loc .~~ closed ~~- base |||
               edlr <<< base -~~ closed ~~- base |||
               is   <<<  loc .~~ closed ~~. loc  ... h

dangleknot  =  kndr <<<          knot   ~~- base |||
               kndl <<< base -~~ knot            |||
               kndlr<<< base -~~ knot   ~~- base |||
               pk   <<<          knot            ... h
```

```
closed  = tabulated (
          (stack ||| hairpin ||| leftB ||| rightB ||| iloop ||| multiloop)
                                                        `with` stackpair ... h)

stack   = sr <<< base -~~ closed ~~- base
hairpin = hl <<< base -~~ base --~ (region `with` (minloopsize 3))
                                                        ~~- base ~~- base
leftB   = bl <<< base -~~ base --~ region ~~~ closed    ~~- base ~~- base ... h
rightB  = br <<< base -~~ base --~ closed ~~~ region    ~~- base ~~- base ... h
iloop   = il <<< base -~~ base --~ region ~+~ (closed, region)
                                                        ~~- base ~~- base ... h

multiloop =
   mldl <<< base -~~ base --~ base --- ml_comps1        ~~- base ~~- base |||
   mldr <<< base -~~ base --~          ml_comps1 ~~- base ~~- base ~~- base |||
   mldlr<<< base -~~ base --~ base --- ml_comps1 ~~- base ~~- base ~~- base |||
   ml   <<< base -~~ base --~          ml_comps1        ~~- base ~~- base ... h
                    where
ml_comps1 = tabulated (
            sadd  <<< base     -~~ ml_comps1             |||
            cadd  <<< mldangle ~~~ ml_comps              |||
            cor   <<< (region `with` (minloopsize 3))
                      ~~~ (ul <<< (pkml <<< dangleknot))  |||
            addss <<< (pkml <<< dangleknot)
                      ~~~ (region `with` (minloopsize 3)) ... h_l)
                          -- a single pseudoknot inside a multiloop

ml_comps  = tabulated (
            sadd <<< base     -~~ ml_comps |||
            cadd <<< mldangle ~~~ ml_comps |||
            addss<<< mldangle ~~~ uregion  ... h_l)

pk_comps  = tabulated (         -- in pk_comps unpaired bases yield a npp penalty
            cadd <<< singlestrand -~~ pk_comps ||| -- one  single base at a time
            cadd <<< mldangle     ~~~ pk_comps |||
            cadd <<< mldangle     ~~~ (ul <<< emptystrand) ... h_l)

mldangle  = mlstem <<< dangle     |||                   -- adds ml_penalty (40)
            pkml   <<< dangleknot ... h                 -- adds pkml_penalty (600)

knot      = tabulated ( pknot ... h_p)

-- Construction of the pseudoknot out of 4 regions (2 stems) and 3 internal
-- components. Note that one base in front and two bases in back are left
-- unpaired explicitly.

pknot (i,j) = [pk energy a u b v a' w b' | k<-[i+3 .. j-8], l<-[k+4 .. j-4],

                          -- look up precomputed helix lengths
                          (alphanrg, alphalen) <- stacklen (i,k),
                          (betanrg, betalen)   <- stacklen (l,j),

                          -- don't let a-a' and b-b' collide within u
                          let h = min alphalen (k-i-1),
                          h >= 2,
```

```
                        -- don't let a-a' and b-b' collide within w
                        let tmph' = min betalen (j-l-2),

                        -- don't let a-a' and b-b' collide within v
                        let h' = min tmph' (l-k-h),
                        h' >= 2,

                        a <- region             (i    , i+h    ),
                        u <- front  j           (i+h+1, k      ),
                        b <- region             (k    , k+h'   ),
                        v <- middle (j-h') (i+h) (k+h' , l-h    ),
                        a'<- region             (l-h  , l      ),
                        w <- back   i           (l    , j-h'-2 ),
                        b'<- region             (j-h' , j      ),

                        -- recalculate the energy of shrinked helices
                        (acorrectionterm, _) <- stacklen (i+h -1,l-h +1),
                        (bcorrectionterm, _) <- stacklen (k+h'-1,j-h'+1),
                        let energy = alphanrg - acorrectionterm
                                   + betanrg - bcorrectionterm
                    ] where

  -- The internal parts of a pseudoknot:

          front j    =           front'          |||
                       frd j <<< front' ~~- base ... h_l
                                    -- one base dangling of b,b'
          front'     = ul    <<< emptystrand     |||
                               pk_comps        ... h_l

          middle k l = emptymid k l <<< empty                |||
                       midbase  k l <<< base                 |||
                       middlro  k l <<< base -~~ base        |||
                       middl    k   <<< base -~~ mid         |||
                       middr      l <<<          mid ~~- base |||
                       middlr   k l <<< base -~~ mid ~~- base |||
                                                 mid         ... h_l
          mid        = ul           <<< singlestrand         |||
                                        pk_comps             ... h_l

          back i     =                    back' |||
                       bkd i <<< base -~~ back' ... h_l
                                    --  one base dangling of a,a'
          back'      = ul    <<< emptystrand    |||
                               pk_comps       ... h_l

singlestrand = pss <<< region
emptystrand  = pss <<< uregion
```

Data sets

Sequences used in the evaluation of *pknotsRG*. The first three entries each represent a set of sequences. # BP = number of base pairs in the reference structure.

Sequence ID	Description	Length	# BP	Reference
PseudoBase	Collection of pseudoknots extracted from the literature	variable	variable	[van Batenburg *et al.*, 2001]
7 HIVRT	Ligands of HIV reverse transcriptase isolated by SELEX	35	11	[Tuerk *et al.*, 1992]
11 tRNAs	Selected transfer RNAs	71-82	18-19	[Sprinzl *et al.*, 1998]
HDV	Self-cleaving HDV ribozyme	87	32	[Ferré-D'Amaré *et al.*, 1998]
TYMV	tRNA-like structure of 3' end of TYMV	86	24	[Deiman *et al.*, 1997]
TMV-up	Upstream pseudoknot domain of the 3' UTR of TMV	85	25	[van Belkum *et al.*, 1985]
TMV-down	tRNA-like downstream pseudoknot domain of the 3' UTR of TMV	105	34	[van Belkum *et al.*, 1985]
ORSV	3' UTR of ORSV possibly involved in virus replication	419	136	[Gultyaev *et al.*, 1994]
STNV	3' UTR of STNV involved in regulation of translation	252	69	[Danthinne *et al.*, 1991]
tmRNA	Transfer-messenger RNA of E.coli rescues stalled ribosomes by *trans*-translation	363	104	[Nameki *et al.*, 1999]

Table A.1: *pknotsRG* test data set.

The following data set has been used in the evaluation of *RNAcast*. A complete list of all sequences and structures is also available at `http://bibiserv.techfak.uni-bielefeld.de/rnacast/references.html`.

ID	length	#	Description	Source
lin4	70-72	9	microRNA precursor	Rfam [Griffiths-Jones *et al.*, 2003]
IRES	79-84	7	IRES regions of Picornaviridae viruses	[Witwer *et al.*, 2001]
tRNA	76-93	11	transfer RNA	[Sprinzl *et al.*, 1998]
srp RNA	78-107	4	RNA of the Signal Recognition Particle (Eubacterial)	SRPDB [Rosenblad *et al.*, 2003]
riboswitch	97-100	7	Purine riboswitch	Rfam
S box	103-114	11	SAM riboswitch (S box leader)	Rfam
5S rRNA	117-120	5	Component of the large ribosomal subunit	5S rRNA DB [Szymanski *et al.*, 2000]
U12 RNA	130-157	6	Small nuclear RNA, component of the spliceosome	Rfam
U1 RNA	157-163	4	See U12	Rfam
U2 RNA	188-197	5	See U12	Rfam

Table A.2: *RNAcast* test data set.

Bibliography

Akutsu, T. (2000) Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, **104**, 45–62.

Altuvia, S. and Wagner, E. G. H. (2000) Switching on and off with RNA. *Proceedings of the National Academy of Sciences*, **97**, 9824–9826.

Argaman, L. and Altuvia, S. (2000) fhlA repression by OxyS RNA: kissing complex formation at two sites results in a stable antisense-target RNA complex. *Journal of Molecular Biology*, **300**, 1101–1112.

Baranov, P. V., Gurvich, O. L., Hammer, A. W., Gesteland, R. F. and Atkins, J. F. (2003) RECODE 2003. *Nucleic Acids Research*, **31**, 87–89.

Barette, I., Poisson, G., Gendron, P. and Major, F. (2001) Pseudoknots in prion protein mRNAs confirmed by comparative sequence analysis and pattern searching. *Nucleic Acids Research*, **29**, 753–758.

van Batenburg, F. H. D., Gultyaev, A. P. and Pleij, C. W. A. (2001) PseudoBase: structural information on RNA pseudoknots. *Nucleic Acids Research*, **29**, 194–195.

Bekaert, M., Bidou, L., Denise, A., Duchateau-Nguyen, G., Forest, J.-P., Froidevaux, C., Hatin, I., Rousset, J.-P. and Termier, M. (2003) Towards a computational model for -1 eukaryotic frameshifting sites. *Bioinformatics*, **19**, 327–335.

van Belkum, A., Abrahams, J. P., Pleij, C. W. and Bosch, L. (1985) Five pseudoknots are present at the 204 nucleotides long 3' noncoding region of tobacco mosaic virus RNA. *Nucleic Acids Research*, **13**, 7673–7686.

Cai, L., Malmberg, R. L. and Wu, Y. (2003) Stochastic modeling of RNA pseudoknotted structures: a grammatical approach. *Bioinformatics*, **19 Suppl 1**, i66–i73.

Cech, T. (1988) Conserved sequences and structures of group I introns: building an active site for RNA catalysis—a review. *Gene*, **73**, 259–271.

Cech, T. R. (1990) Self-splicing of group i introns. *Annual Review of Biochemistry*, **59**, 543–568.

Chen, J.-L., Blasco, M. A. and Greider, C. W. (2000) Secondary structure of vertebrate telomerase RNA. *Cell*, **100**, 503–514.

Condon, A., Davy, B., Rastegari, B., Zhao, S. and Tarrant, F. (2004) Classifying RNA pseudoknotted structures. *Theoretical Computer Science*, **320**, 35–50.

Danthinne, X., Seurinck, J., van Montagu, M., Pleij, C. W. A. and van Emmelo, J. (1991) Structural similarities between the RNAs of two satellites of tobacco necrosis virus. *Virology*, **185**, 605–614.

Deiman, B., Kortlever, R. and Pleij, C. (1997) The role of the pseudoknot at the 3' end of turnip yellow mosaic virus RNA in minus-strand synthesis by the viral RNA-dependent RNA polymerase. *Journal of Virology*, **71**, 5990–5996.

Deogun, J., Donis, E., Komina, O. and Ma, F. (2004) RNA secondary structure prediction with simple pseudoknots. In *Proc. Second Asia-Pacific Bioinformatics Conference 2004*, pp. 239–246.

Dinman, J. D. (2006) Programmed ribosomal frameshifting goes beyond viruses: Organisms from all three kingdoms use frameshifting to regulate gene expression, perhaps signaling a paradigm shift. *Microbe*, **1**, 521–527.

Dirks, R. and Pierce, N. A. (2003) A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of Computational Chemistry*, **24**, 1664–1677.

Dmitriev, S. E., Andreev, D. E., Terenin, I. M., Olovnikov, I. A., Prassolov, V. S., Merrick, W. C. and Shatsky, I. N. (2007) Efficient translation initiation directed by the 900-nucleotide-long and GC-rich 5' untranslated region of the human retrotransposon LINE-1 mRNA is strictly cap dependent rather than internal ribosome entry site mediated. *Molecular and Cellular Biology*, **27**, 4685–4697.

Doshi, K., Cannone, J., Cobaugh, C. and Gutell, R. (2004) Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics*, **5**.

Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. F. (1990) Sparse dynamic programming. In *Proc. 1st Symp. Discrete Algorithms*, pp. 513–522. ACM and SIAM.

Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. F. (1992a) Sparse dynamic programming I: linear cost functions. *Journal of the ACM*, **39**, 519–545.

Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. F. (1992b) Sparse dynamic programming II: convex and concave cost functions. *Journal of the ACM*, **39**, 546–567.

Evers, D. and Giegerich, R. (2001) Reducing the conformation space in RNA structure prediction. In *German Conference on Bioinformatics*, pp. 118–124.

Ferré-D'Amaré, A. R., Zhou, K. and Doudna, J. A. (1998) Crystal structure of a hepatitis delta virus ribozyme. *Nature*, **395**, 567–674.

Gardner, P. P. and Giegerich, R. (2004) A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics*, **5**.

Gardner, P. P., Wilm, A. and Washietl, S. (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Research*, **33**, 2433–2439.

Giegerich, R. (2000) Explaining and controlling ambiguity in dynamic programming. In *Proc. Combinatorial Pattern Matching*, pp. 46–59. Springer Verlag.

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

Giegerich, R., Meyer, C. and Steffen, P. (2004a) A discipline of dynamic programming over sequence data. *Science of Computer Programming*, **51**, 215–263.

Giegerich, R. and Steffen, P. (2006) Challenges in the compilation of a domain specific language for dynamic programming. In *Proceedings of the 2006 ACM Symposium on Applied Computing*.

Giegerich, R., Voss, B. and Rehmsmeier, M. (2004b) Abstract Shapes of RNA. *Nucleic Acids Research*, **32**, 4843–4851.

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

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

Gultyaev, A. P., van Batenburg, F. and Pleij, C. (1999) An approximation of loop free energy values of RNA H-pseudoknots. *RNA*, **5**, 609–617.

Gultyaev, A. P., van Batenburg, F. and Pleij, C. W. A. (1994) Similarities between the secondary structure of satellite tobacco mosaic virus and tobamovirus RNAs. *Journal of General Virology*, **75**, 2851–2856.

Gutell, R. R., Power, A., Hertz, G. Z., Putz, E. J. and Stormo, G. D. (1992) Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *Nucleic Acids Research*, **20**, 5785–5795.

Hammell, A. B., Taylor, R. C., Peltz, S. W. and Dinman, J. D. (1999) Identification of Putative Programmed -1 Ribosomal Frameshift Signals in Large DNA Databases. *Genome Research*, **9**, 417–427.

Hansen, T. M., Reihani, S. N. S., Oddershede, L. B. and Sørensen, M. A. (2007) Correlation between mechanical strength of messenger rna pseudoknots and ribosomal frameshifting. *Proceedings of the National Academy of Sciences*, **104**, 5830–5835.

Harmanci, A., Sharma, G. and Mathews, D. (2007) Efficient pairwise rna structure prediction using probabilistic alignment constraints in dynalign. *BMC Bioinformatics*, **8**, 130.

Havgaard, J. H., Lyngsø, R. B., Stormo, G. D. and Gorodkin, J. (2005) Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics*, **21**, 1815–1824.

Havgaard, J. H., Torarinsson, E. and Gorodkin, J. (2007) Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Computational Biology*, **3**, 1896–1908.

Höchsmann, T., Höchsmann, M. and Giegerich, R. (2006) Thermodynamic matchers: strengthening the significance of RNA folding energies. *Comput Syst Bioinformatics Conf*, pp. 111–121.

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

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

Hofacker, I. L., Fekete, M. and Stadler, P. F. (2002) Secondary structure prediction for aligned RNA sequences. *Journal of Molecular Biology*, **319**, 1059–1066.

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, S., Tacker, M. and Schuster, P. (1994) Fast folding and comparison of RNA secondary structures. *Monatshefte f. Chemie*, **125**, 167–188.

Huang, C.-H., Lu, C. L. and Chiu, H.-T. (2005) A heuristic approach for detecting RNA H-type pseudoknots. *Bioinformatics*, **21**, 3501–3508.

Isambert, H. and Siggia, E. D. (2000) Modeling RNA folding paths with pseudoknots: Application to hepatitis delta virus ribozyme. *Proceedings of the National Academy of Sciences*, **97**, 6515–6520.

Jacobs, J. L., Belew, A. T., Rakauskaite, R. and Dinman, J. D. (2007) Identification of functional, endogenous programmed -1 ribosomal frameshift signals in the genome of Saccharomyces cerevisiae. *Nucleic Acids Research*, **35**, 165–174.

Kikinis, Z., Eisenstein, R. S., Bettany, A. J. E. and Munro., H. N. (1995) Role of RNA secondary structure of the iron-responsive element in translational regulation of ferritin synthesis. *Nucleic Acids Research*, **23**, 4190–4195.

Klein, R. J. and Eddy, S. R. (2003) RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, **4**, 44.

Knudsen, B. and Hein, J. (1999) RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, **15**, 446–454.

Kontos, H., Napthine, S. and Brierley, I. (2001) Ribosomal pausing at a frameshifter RNA pseudoknot is sensitive to reading phase but shows little correlation with frameshift efficiency. *Molecular and Cellular Biology*, **21**, 8657–8670.

Lagos-Quintana, M., Rauhut, R., Lendeckel, W. and Tuschl, T. (2001) Identification of novel genes coding for small expressed RNAs. *Science*, **294**, 853–858.

Lau, N. C., Lim, L. P., Weinstein, E. G. and Bartel, D. P. (2001) An abundant class of tiny RNAs with probable regulatory roles in caenorhabditis elegans. *Science*, **294**, 858–862.

Lee, R. and Ambros, V. (2001) An extensive class of small RNAs in caenorhabditis elegans. *Science*, **294**, 862–864.

Lefebvre, F. (1996) A grammar-based unification of several alignment and folding algorithms. In *Proceedings 4th ISMB*, pp. 143–154. AAAI Press, Menlo Park, CA, USA.

Lyngsø, R. B. and Pedersen, C. N. (2000) Pseudoknots in RNA secondary structures. In *Proceedings of the fourth annual international conference on computational molecular biology*, pp. 201–209. ACM Press.

Lyngsø, R. B. and Pedersen, C. N. (2001) RNA pseudoknot prediction in energy based models. *Journal of Computational Biology*, **7**, 409–428.

Lyngsø, R. B., Zuker, M. and Pedersen, C. N. (1999) Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, **15**, 440–445.

Macke, T. J., Ecker, D. J., Gutell, R. R., Gautheret, D., Case, D. A. and Sampath, R. (2001) RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Research*, **29**, 4724–4735.

Mathews, D. H., Disney, M. D., Childs, J. L., Schroeder, S. J., Zuker, M. and Turner, D. H. (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, **101**, 7287–7292.

Mathews, D. H., Sabina, J., Zuker, M. and Turner, D. H. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, **288**, 911–940.

Mathews, D. H. and Turner, D. H. (2002) Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, **317**, 191–203.

Mattick, J. S. (2003) Challenging the dogma: the hidden layer of non-protein-coding RNAs in complex organisms. *Bioessays*, **25**, 930–939.

McCaskill, J. S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Moon, S., Byun, Y. and Han, K. (2007) FSDB: a frameshift signal database. *Computational Biology and Chemistry*, **31**, 298–302.

Moon, S., Byun, Y., Kim, H.-J., Jeong, S. and Han, K. (2004) Predicting genes expressed via -1 and +1 frameshifts. *Nucleic Acids Research*, **32**, 4884–4892.

Nameki, N., Feldan, B., Atkins, J. F., Gesteland, R. F., Himeno, H. and Muto, A. (1999) Functional and structural analysis of a pseudoknot upstream of the tag-encoded sequence in E. coli tmRNA. *Journal of Molecular Biology*, **286**, 733–744.

Nawrocki, E. P. and Eddy, S. R. (2007) Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Computational Biology*, **3**, e56.

Nibert, M. L. (2007) '2A-like' and 'shifty heptamer' motifs in penaeid shrimp infectious myonecrosis virus, a monosegmented double-stranded RNA virus. *Journal of General Virology*, **88**, 1315–1318.

Nussinov, R., Pieczenik, G., Griggs, J. R. and Kleitman, D. J. (1978) Algorithms for loop matchings. *SIAM Journal of Applied Mathematics*, **35**, 68–82.

Pedersen, J. S., Bejerano, G., Siepel, A., Rosenbloom, K., Lindblad-Toh, K., Lander, E. S., Kent, J., Miller, W. and Haussler, D. (2006) Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Computational Biology*, **2**, e33.

Plant, E. P. and Dinman, J. D. (2005) Torsional restraint: a new twist on frameshifting pseudoknots. *Nucleic Acids Research*, **33**, 1825–1833.

Plant, E. P., Pérez-Alvarado, G. C., Jacobs, J. L., Mukhopadhyay, B., Hennig, M. and Dinman, J. D. (2005) A three-stemmed mRNA pseudoknot in the SARS coronavirus frameshift signal. *PLoS Biology*, **3**, e172.

Reeder, J. and Giegerich, R. (2004) Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, **5:104**.

Reeder, J. and Giegerich, R. (2005) Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. *Bioinformatics*, **21**, 3516–3523.

Reeder, J., Reeder, J. and Giegerich, R. (2007) Locomotif: From graphical motif description to RNA motif search. *Bioinformatics*, **23**, i392–400.

Ren, J., Rastegari, B., Condon, A. and Hoos, H. H. (2005) HotKnots: Heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, **11**, 1494–1504.

Rijk, P. D., Wuyts, J. and Wachter, R. D. (2003) RnaViz2: an improved representation of RNA secondary structure. *Bioinformatics*, **19**, 299–300.

Rivas, E. and Eddy, S. R. (1999) A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, **285**, 2053–2068.

Rivas, E. and Eddy, S. R. (2000) The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, **16**, 334–340.

Roberts, M. D., Martin, N. L. and Kropinski, A. M. (2004) The genome and proteome of coliphage T1. *Virology*, **318**, 245–266.

Rosenblad, M. A., Gorodkin, J., Knudsen, B., Zwieb, C. and Samuelsson, T. (2003) SRPDB: Signal Recognition Particle Database. *Nucleic Acids Research*, **31**, 363–364.

Ruan, J., Stormo, G. D. and Zhang, W. (2004) An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, **20**, 58–66.

Sankoff, D. (1985) Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal of Applied Mathematics*, **45**, 810–825.

Schlax, P. J., Xavier, K. A., Gluick, T. C. and Draper, D. E. (2001) Translational repression of the Escherichia coli α Operon mRNA. Importance of an mRNA conformational switch and a ternary entrapment complex. *Journal of Biological Chemistry*, **276**, 38494–38501.

Searls, D. (1997) Linguistic approaches to biological sequences. *Computer Applications in the Biosciences*, **13**, 333–344.

Shapiro, B. A. and Wu, J. C. (1997) Predicting RNA H-type pseudoknots with the massively parallel genetic algorithm. *Computer Applications in the Biosciences*, **13**, 459–471.

Siebert, S. and Backofen, R. (2005) MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, **21**, 3352–3359.

Sprinzl, M., Horn, C., Brown, M., Ioudovitch, A. and Steinberg, S. (1998) Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Research*, **26**, 148–153.

Staple, D. W. and Butcher, S. E. (2005) Pseudoknots: RNA structures with diverse functions. *PLoS Biology*, **3**, e213.

Steffen, P. (2006) *Compiling a Domain Specific Language for Dynamic Programming*. Ph.D. thesis, University of Bielefeld, Germany.

Steffen, P. and Giegerich, R. (2005) Versatile and declarative dynamic programming using pair algebras. *BMC Bioinformatics*, **6**.

Steffen, P., Voß, B., Rehmsmeier, M., Reeder, J. and Giegerich, R. (2006) RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics*, **22**, 500–503.

Szymanski, M., Barciszewska, M. Z., Barciszewski, J. and Erdmann, V. A. (2000) 5S ribosomal RNA database Y2K. *Nucleic Acids Research*, **28**, 166–167.

Tabaska, J. E., Cary, R. B., Gabow, H. N. and Stormo, G. D. (1998) An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, **14**, 691–699.

Taufer, M., Leung, M.-Y., Johnson, K. and Licon, A. (2007) RNAVLab: A unified environment for computational RNA structure analysis based on grid computing technology. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pp. 1–8.

Tinoco, I. and Bustamante, C. (1999) How RNA folds. *Journal of Molecular Biology*, **293**, 271–281.

Torarinsson, E., Havgaard, J. H. and Gorodkin, J. (2007) Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, **23**, 926–932.

Torarinsson, E., Sawera, M., Havgaard, J. H., Fredholm, M. and Gorodkin, J. (2006) Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure. *Genome Research*, **16**, 885–889.

Touzet, H. and Perriquet, O. (2004) CARNAC: folding families of related RNAs. *Nucleic Acids Research*, **32**, 142–145.

Tuerk, C., MacDougal, S. and Gold, L. (1992) RNA pseudoknots that inhibit human immunodeficiency virus type 1 reverse transcriptase. *Proceedings of the National Academy of Sciences*, **89**, 6988–6992.

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

Washietl, S., Hofacker, I. L., Lukasser, M., Hüttenhofer, A. and Stadler, P. F. (2005a) Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nature Biotechnology*, **23**, 1383 – 1390.

Washietl, S., Hofacker, I. L. and Stadler, P. F. (2005b) Fast and reliable prediction of noncoding RNAs. *Proceedings of the National Academy of Sciences*, **102**, 2454–2459.

Will, S., Reiche, K., Hofacker, I. L., Stadler, P. F. and Backofen, R. (2007) Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Computational Biology*, **3**, e65.

Witwer, C., Hofacker, I. L. and Stadler, P. F. (2004) Prediction of consensus RNA secondary structures including pseudoknots. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **01**, 66–77.

Witwer, C., Rauscher, S., Hofacker, I. L. and Stadler, P. F. (2001) Conserved RNA secondary structures in Picornaviridae genomes. *Nucleic Acids Research*, **29**, 5079–5089.

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

Xayaphoummine, A., Bucher, T., Thalmann, F. and Isambert, H. (2003) Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. *Proceedings of the National Academy of Sciences*, **100**, 15310–15315.

Yao, Z., Weinberg, Z. and Ruzzo, W. L. (2006) CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics*, **22**, 445–452.

Zeenko, V. and Gallie, D. R. (2005) Cap-independent translation of tobacco etch virus is conferred by an RNA pseudoknot in the 5'-leader. *Journal of Biological Chemistry*, **280**, 26813–26824.

Zuker, M. (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, **31**, 3406–3415.

Zuker, M. and Sankoff, S. (1984) RNA secondary structures and their prediction. *Bulletin of Mathematical Biology*, **46**, 591–621.

Zuker, M. and Stiegler, P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary informations. *Nucleic Acids Research*, **9**, 133–148.