

WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF BIOMEDICAL ENGINEERING

RNA STRUCTURAL ALIGNMENT USING
STOCHASTIC CONTEXT-FREE GRAMMARS

by

Robin D. Dowell B.S., B.S., M.S.

Prepared under the direction of Sean R. Eddy

A dissertation presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

Doctor of Science

December, 2004

Saint Louis, Missouri

WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF BIOMEDICAL ENGINEERING

ABSTRACT

RNA STRUCTURAL ALIGNMENT USING
STOCHASTIC CONTEXT-FREE GRAMMARS

by Robin D. Dowell

ADVISOR: Sean R. Eddy

December, 2004
Saint Louis, Missouri

Noncoding RNAs are any transcript that functions directly as RNA rather than being translated into protein. Many noncoding RNAs conserve intramolecular secondary structure. When a lot of data is available, comparative sequence analysis is a powerful approach to predicting RNA secondary structure. More frequently, only a small number of structurally homologous RNA sequences are known. In this case some evolutionary information is available, but insufficient for a pure comparative analysis. Predicting consensus RNA secondary structure for small datasets is an open problem. In this work we approach the problem by examining the case of predicting the consensus structure between two RNA sequences. We use a pairwise stochastic context-free grammar to model

secondary structure and alignment simultaneously. The resulting algorithm is computationally expensive, so we also develop a method of constraining the algorithm to reduce memory and runtime requirements.

Contents

List of Tables	v
List of Figures	vi
Acknowledgments	vii
1 Background	1
1.1 Noncoding RNAs	1
1.2 Secondary Structure	3
1.3 Thermodynamics	5
1.4 Covariation	8
1.5 Probabilistic Models	9
1.6 The Sankoff Algorithm	11
1.7 Outline of this work	13
2 Probabilistic Models of RNAs	15
2.1 Secondary Structure	15
2.1.1 Context-Free Grammars	17
2.1.2 Stochastic context-free grammars	19
2.2 Sequence Alignment	21
2.2.1 Pairwise Hidden Markov Models	22
2.2.2 Optimal Accuracy Alignment	24
2.2.3 Evolutionary Models	25
2.3 Ambiguity	27
2.3.1 Structural ambiguity	27
2.3.2 Alignment ambiguity in pairHMMs	30
2.3.3 Consensus Structure	31

2.4	Concluding Comments	31
3	Single Sequence RNA Secondary Structure Prediction	33
3.1	Overview	34
3.2	Impact of Structural Ambiguity	35
3.3	SCFG designs	37
3.4	Benchmarking	41
3.4.1	Grammar Comparison	42
3.4.2	Stacking Variations	45
3.4.3	Comparison to Mathews et al. benchmark	47
3.5	Concluding Remarks	49
4	Pairwise RNA Structural Alignment	51
4.1	Pairwise Grammar Design	52
4.2	Parameterization	54
4.3	Benchmarks	56
4.4	Concluding Remarks	60
5	A Constrained Structural Alignment Algorithm	63
5.1	General Methods of Constraint	63
5.2	Constrain on Alignment Pins Algorithm	64
5.3	A Pin Generating System	68
5.4	Concluding Remarks	71
6	Future Work	78
	Appendix A Availability	84
	References	85
	Vita	104

List of Tables

3.1	Parameters for simple grammars	39
3.2	Parameters for stacking grammars	40
3.3	Single Sequence Grammar Performance	44
3.4	Stacking Variations	46
4.1	Single sequence versus Pairwise Performance	57
5.1	Performance using Pins	67
5.2	Comparing Structural Alignment Programs	68
5.3	Accuracy of Posterior Probabilities	70
5.4	Performance using Predicted Pins	71
6.1	PairSCFG performance	82

List of Figures

1.1	Example RNA secondary structure	3
2.1	Example parse trees	19
2.2	Example pairHMM	23
2.3	Sources of structural ambiguity	28
3.1	Example parse trees for simple grammars	50
4.1	Example pairSCFG parse tree	61
4.2	Resource utilization of pairSCFG	62
5.1	Constrain on Alignment Pins Algorithm	73
5.2	Single pin memory improvements	74
5.3	Resource utilization of constrained pairSCFG	75
5.4	Example posterior probability table	76
5.5	Number Quality Posteriors vs Percent Identity	76
5.6	Types of Constraint Information	77

Acknowledgments

If I have accomplished anything or learned anything it is, as Sir Isaac Newton said, “because I have stood on the shoulders of giants!” I was privileged to work in the laboratory of a great scientist, Sean Eddy. I thank him for his guidance, insightful criticism, unwaivering support and broad shoulders. I am also grateful to Elena Rivas, a friend and mentor. I thank the members of my committee, Gary Stormo, Michael Brent, David Sept, and Jeremy Buhler for their insight and encouragement.

Bjarne Knudsen, David Mathews, and Ian Holmes deserve special mention for the numerous discussions, source code, data sets, and friendly competition they each provided. I have had the privilege and opportunity to work with them and many other wonderful people including Ivo Hofacker, Michael Zuker, Lincoln Stein, Rodney Jokerst, Todd Lowe, Takis Benos, and Sam Griffiths-Jones.

All of the Eddy lab people have been great to work with over the years. I particularly thank Zhirong Bao, Steve Johnson, Robbie Klein, and Shawn Stricklin for discussions, encouragement, and advice. Goran Ceric is a superman. I was supported financially by a Howard Hughes Predoctoral Fellowship and a Spencer T. Olin Fellowship, both expertly administered by the unflappable Nancy Pope.

Finally I want to thank my friends and family members for their love and support through the years. This could not have been done without them. I dedicate this work to two Davids: my father and hero David Dowell and my husband and close friend David Deen.

“Knowledge is in the end based on acknowledgment.”

Ludwig Wittgenstein

Robin D. Dowell

Washington University in Saint Louis
December 2004

Chapter 1

Background

1.1 Noncoding RNAs

A noncoding ribonucleic acid (ncRNA) is any RNA transcript that functions directly as RNA rather than being translated into protein. The list of ncRNAs is extensive and growing [32, 35, 107, 161]. They are a diverse collection, ranging in size from 21 nt (miRNAs) to >10,000 nt (*Xist*). They play roles as catalytic, regulatory, or structural components in the cell. A few deserve special mention here as they are important to this work.

Ribosomes were originally identified as discrete particles in the cytoplasm [133] that were soon determined to be the site of protein synthesis [19]. Speculation about the function of ribosomal RNA (rRNA) began when cleavage of a single phosphodiester bond in 16S rRNA was shown to cause inactivation of ribosome function [10, 153]. The crystal structure of the large subunit of the ribosomal RNA confirmed that the ribosome is in fact a ribozyme [6, 126].

Crick famously identified the need for an adapter between the genetic code and amino acid encodings [19]. Serendipitously, the same year Hoagland noted that specific RNAs pick up radio-labeled leucine, transferring it to a growing peptide chain [67]. The first transfer RNA (tRNA) sequence was determined in 1965 [75]. Just one year later, with only four comparative sequences, the secondary structure of tRNA was speculated to be the cloverleaf [113, 142, 186].

Enzymatic digestion [186], X-ray crystallography [94], and nuclear magnetic resonance [155] confirmed the now famous cloverleaf.

Ribonuclease P (RNase P) was first characterized as an enzyme required for the processing of the 5' termini of precursor tRNA in *Escherichia coli* [3]. Purified RNase P was found to contain both RNA and protein, both essential for function [45,160]. Later experiments showed that the catalytic subunit of this enzyme is the RNA subunit [52].

The signal recognition protein was renamed to signal recognition particle (SRP) when it was found to contain 7S RNA. The SRP RNA is involved in targeting secretory and membrane proteins to the appropriate membrane in the cell [92,173].

The tmRNA combines the functions of mRNA and tRNA in one molecule. The 3' and 5' ends come together to form a tRNA-like structure that can be charged with alanine. It can then occupy the ribosomal A site of a stalled ribosome. The alanine is donated to the growing peptide chain in the P site and the mRNA portion of the tmRNA becomes the new message. This results in an 11 amino-acid tag to the growing chain which tags it for degradation [5,66]. To accomplish this interesting mechanism requires tmRNA to take a complex shape with at least four pseudoknots [37].

Numerous databases have been developed which are devoted to one particular type of ncRNA. Examples include the RNase P database [11], the Signal Recognition Particle database [149], the tmRNA database [193], the Spritzl tRNA database [158], and the European ribosomal RNA database [182]. The quality and formatting of the information available is quite variable. Consequently, there have also been recent efforts to collect and standardize ncRNA information including the comparative RNA website [13] and the Rfam database [51]. The availability of these data collections is vital to both computational and experimental progress in RNA biology.

The best known ncRNAs have complex three-dimensional structures. Biochemists utilize X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy studies to elucidate the tertiary structure of an RNA molecule. NMR methods were used to solve the tertiary structure of tetraloop

motifs [65,90,171]. In general, NMR studies are limited to roughly 20 kD (60 nucleotides) [139]. X-ray crystallography has produced many of the most spectacular confirmations of the enzymatic activity of ribozyme RNAs [6,29,74,126]. Unfortunately, tertiary structure studies are time consuming, expensive, and complex.

1.2 Secondary Structure

Doty and colleagues showed that RNA demonstrates changes in optical density with increasing temperature [28] that are characteristic of secondary structure. They suggested that about half of the bases in the RNA molecules examined are involved in helical regions. (Interestingly, this estimate has held over time as roughly half of the nucleotides in known RNA secondary structures participate in base pair interactions [118].) A year later they proposed that RNA base pairs are the canonical Watson-Crick A·U and G·C pairs [43,44]. Crick proposed, after examining how tRNAs might recognize the genetic code, that G·U is also a valid base pair in RNA secondary structure [20].

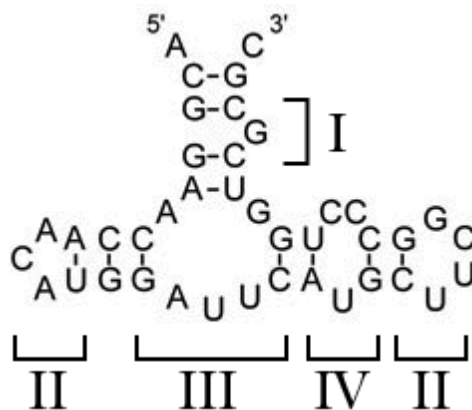


Figure 1.1: **Example RNA secondary structure.** An example RNA secondary structure, courtesy of [30]. The black lines connect base pairs. The four kinds of loops are labeled by roman numerals where I is a bulge loop, II are hairpin loops, III is a multibranch loop, and IV is an internal loop.

Secondary structure interactions are typically stronger than tertiary [84,114]. An example secondary structure is shown in Figure 1.1. A

secondary structure can be decomposed into a number of motifs known as internal loops, bulge loops, hairpin loops, multibranch loops, and base pairs. Many ncRNAs conserve secondary structure rather than primary sequence.

By convention, a base pair is described between positions i and j by $i \cdot j$ where $i < j$. When considering two base pairs $i \cdot j$ and $i' \cdot j'$, $i = i'$ if and only if $j = j'$. In other words, each nucleotide can take part in at most one base pair. Base triples and G-quartets violate this definition, therefore they are considered tertiary features. Adjacent base pairs ($i' = i + 1$, $j' = j - 1$) are said to be ‘neighbors’ and participate in stacking. A base pair without any neighboring pairs is a lone pair. A pseudoknot is when any two base pairs satisfy $i < i' < j < j'$. Pseudoknot base pairs are not as stable as base pairs in other motifs, often depending greatly on the Mg^{2+} concentration [48, 184].

At least two definitions of secondary structure exist. The most commonly utilized definition considers a secondary structure to be a set of base pairs [189]. An alternative definition is as a set of loops, where adjacent base pairs are considered a loop of size zero [40]. The distinction is perhaps pedantic, but becomes important when comparing structures, counting unique structures, or enumerating all possible structures. The set of base pairs definition is assumed here.

Accurate RNA secondary structure predictions help in understanding an RNA’s function, in identifying novel functional RNAs in genome sequences, and in recognizing evolutionarily related RNAs in other organisms. Often the sequence of a proposed ncRNA is known, but its secondary structure is not known. Extensive work has been directed toward predicting the secondary structure of a RNA sequence. The influence of tertiary interactions is generally ignored when predicting secondary structure.

The problem of RNA secondary structure prediction is an optimization problem. Therefore, the goal is to find the best of all possible solutions. It is important then to define “best”. An *objective function* is any function which determines how good a particular solution is. Given a perfect objective function, any reasonable *algorithm* could be utilized to generate accurate RNA secondary

structure predictions. Unfortunately, knowledge about the cellular forces that determine secondary structure is limited.

1.3 Thermodynamics

Secondary structures are stabilized by noncovalent interactions such as stacking and hydrogen-bonding. It was quickly recognized that understanding the energetics of these interactions would lead to prediction of secondary structure [168]. Consequently, tremendous effort has been made toward measuring the nucleic acid energy parameters. The main two methods are absorbance melting curves and microcalorimetry [88]. The earliest free energy parameters were referred to as “Tinoco rules” [8, 167, 168], but in the mid-1980’s these were supplanted by the “Turner rules” [42, 118, 170, 172, 185].

The current state of the art parameters are composed of base pair and loop components. The stability of helices is considered well determined [185]. But the sequence dependence of thermodynamic parameterization means that loops cannot be exhaustively studied experimentally. Three approaches to loop parameterization have been undertaken. The first uses representative sequences as models to a wider variety of sequences [118]. Alternatively, observed frequencies are considered a proxy to the free energy [83, 118, 172]. The third method, used mostly for multibranch loops, adjusts the thermodynamic parameters to improve the accuracy on known structures [55, 118]. The full set of thermodynamic parameters are therefore an approximation based on a finite set of experiments [116].

Collectively these energies are referred to as ‘nearest neighbor’ rules because individual terms in helices depend not on single base pairs but instead condition on adjacent pairs. The nearest neighbor model is used to predict the standard free-energy ΔG at 37° C of forming secondary structure. Free energies are assumed to be additive and the lowest free energy structure is considered the optimal structure.

One of the first computerized approaches to predicting RNA secondary structure utilized the free energy parameters to score all possible combinations of

stems [138]. The runtime for this combinatorial algorithm grows exponentially with the length of the sequence. Maximizing the number of base pairs was manually utilized to help determine the structure of one of the first tRNA sequences [112]. The first dynamic programming algorithm, by Ruth Nussinov, for predicting RNA secondary structure uses the same scoring system [129]. The technique was quickly extended to the optimization of free energy instead [128, 174, 191].

The most popular algorithm for utilizing the free energy parameters is the dynamic programming method of Zuker [191]. The algorithm computes the minimal free energy (mfe) structure. The algorithm assumes the secondary structure is at equilibrium, that there is a single conformation for the RNA, and that the thermodynamic parameters are error-free. At least three separate implementations of the algorithm exist, *mfold* [190, 191], RNAstructure [116, 118], and the Vienna RNA package [69, 73]. These algorithms are typically $O(L^3)$ in time and $O(L^2)$ in space where L is the length of the sequence [110, 191].

An alternative to a single lowest free energy structure is to compute suboptimals close to the lowest free energy structure. The exact approach utilized varies including finding all substantially different solutions within a given range [177, 187], enumerating all possible solutions within the given range [175, 176, 180], or sampling from the posterior probability distribution [24, 30].

The energy parameters have also been utilized to compute the partition function for secondary structure formation [73, 116, 120]. The partition function provides the normalizing coefficient necessary to convert the free energy of a particular structure into a probability. A massively parallel implementation was undertaken for the study of viral genomes [36]. The partition function can be utilized to calculate structures containing only consistent high probability base pairs [116].

Coaxial stacking has proven to be an important component to the thermodynamic parameters [172]. To manage coaxials, a second energy function, *efn2*, calculation was written to rescore proposed structures. The recommended thermodynamic method for predicting structure was therefore to predict some

number of suboptimals and rescore using *efn2* to select the lowest free-energy structure [118]. Only recently has the RNAstructure implementation included some coaxials in the dynamic programming formulation and no longer depends on *efn2* rescoring [115].

Pseudoknots are known to exist in natural structures such as group I introns [21] and tmRNA [37, 178]. A limited set of energy parameters has been suggested for pseudoknots [54]. A thermodynamic dynamic programming solution has been proposed [145] which is $O(L^6)$ in time and $O(L^4)$ in memory. Work continues on the pseudoknot prediction problem [26, 87, 109, 143, 150]. In the meantime, pseudoknots are excluded from most secondary structure prediction algorithms.

Experimental Constraints

Experience has shown that small changes in thermodynamic parameters can result in very large changes in predicted folding [188]. Independent evaluation of the nearest-neighbor energy parameters concludes that they do not work well for predicting the secondary structure of large RNAs [27, 39, 57]. Consequently, it is generally accepted [115, 189] that predicting a single “correct” fold requires auxiliary information.

The need for auxiliary information was recognized early [140, 191]. This auxiliary information constrains the space of possible secondary structures. Constraints come in many different forms including knowledge about location of binding proteins, intra-RNA cross-linking, chemical modifications, and enzymatic cleavage [34]. Proposed structures can be tested by site-directed mutagenesis and chemical-interference studies.

For example, cobra venom (RNase VI) cleaves RNA in double-stranded regions without consideration of sequence. Several nucleases cleave single-stranded regions without sequence preference. Flavin derivatives can be used to cleave RNA specifically at uridines involved in G·U base pairs via photo-induced mechanisms [12].

Chemical modifications such as CMCT, DMS, and kethoxal react with the hydrogen-bonding groups of solvent-exposed bases. CMCT reacts with U and G, DMS reacts with A and C, and kethoxal with G [34]. To be accessible to chemical modification, the base must be in a single-stranded region, at the end of a helix, or in adjacent G·U pairs. Utilization of chemical modification data as constraints improves prediction accuracy significantly [115, 116].

1.4 Covariation

Luckily, in most cases we have more than a single sequence. With many related sequences, comparative sequence analysis can be utilized to extract information about the common underlying structure. Conserved base pairing interactions are revealed by compensatory mutations in large RNA multiple sequence alignments [2, 18, 53, 59, 100, 125]. Arguably, this is the most powerful source for RNA structure prediction in absence of a crystal structure [56, 57, 59, 131].

The first successful utilization of comparative sequence analysis identified the cloverleaf structure of tRNA using (surprisingly) only four sequences [75, 113, 130, 142, 186]. A non-exhaustive list of RNAs with a secondary structure determined in this fashion includes tRNA, 5S [41], 16S [179], 23S [127], group I introns [22, 122], group II introns [122], RNase P RNA [85], SRP [192], and the telomerase RNA [148]. Lone pairs, non-canonical base pairs, base triples, coaxial stacking, and pseudoknots have all been identified by comparative sequence analysis [121].

Comparative sequence analysis is extremely reliable, and has produced strikingly accurate RNA structure predictions [56, 123]. For example, the predicted secondary structure of ribosomal RNA has been essentially confirmed by recent crystal structures; 97-98% of the predicted base pairs are confirmed by experimental structures [56]. The trouble is that rRNA predictions were refined by experts over twenty years, ultimately utilizing data from about 7000 small subunit rRNA sequences and 1050 large subunit rRNA sequences [56].

There are many RNA structures of biological interest [32, 161], consequently there is a need for automated approaches that combine evolutionary information from comparative sequence analysis with biophysical knowledge of what structures are most plausible. Statistical methods are used to detect covarying base pairs, ranging from mutual information criteria [18, 58] to more sophisticated phylogenetic models [2, 53, 125]. Given an accurate multiple alignment, a large number of sequences, and sufficient sequence diversity, statistical comparative analysis alone is sufficient to produce very accurate structure prediction [56].

One is usually not blessed with the large number of sequences or the expertise that a purely comparative approach requires. An outstanding problem is consensus RNA secondary structure prediction for a small number of structurally homologous RNA sequences. In these cases some evolutionary information is available, but insufficient for a pure comparative analysis. Approaches to this problem described thus far fall into two classes: the RNA sequence alignment is treated as known [14, 18, 60, 71, 72, 89, 98, 100, 102, 134, 162], or the sequences are given unaligned, leading to an even harder problem of simultaneous folding and alignment [49, 50, 80, 87, 108, 119, 135, 137, 152].

1.5 Probabilistic Models

However, it is not clear how best to combine probabilistic evolutionary information with the thermodynamic parameters of the standard energy minimization model into a meaningful, mathematically defensible objective function. Clearly one can use the Gibbs-Boltzmann equation to convert an *overall* ΔG for a structure into a probability of that structure in an ensemble of all possible structures [24, 120], but it is not possible to exactly interpret individual energy *parameters* in the thermodynamic model as log probabilities. Consequently, nearly all consensus RNA structure prediction methods that have been introduced optimize an *ad hoc* weighted combination of thermodynamic parameters and comparative sequence analysis terms.

Basing the overall objective function purely on thermodynamics seems problematic. It is hard to see how to express inherently stochastic evolutionary events in terms of free energies. For example, it does not make much sense to speak of the free energy of a deletion conditional on a divergence time. That said, Mathews and Turner’s Dynalign program does do this [119], and performs well: using a dynamic programming approach to calculate the consensus structure for a pair of RNA sequences by optimizing the sum of the two structures’ predicted free energies, while using an *ad hoc* pseudoenergy penalty for indels.

In contrast, deriving a combined objective function in terms of probability theory is straightforward. Consensus RNA structure prediction can be treated as a Bayesian statistical inference problem [23, 30]. One can find a consensus structure with maximum posterior probability by finding the structure that maximizes the joint probability of both the sequences and the structure; this is the product of an evolutionary model expressing the probability of the homologous sequences given some secondary structure and a model expressing the *a priori* probability of that structure [100].

If one expresses RNA secondary structure plausibility directly as a full probabilistic model, the desired joint probability can be optimized easily, identifying optimal (or sampling suboptimal) consensus structures by dynamic programming. Knudsen and Hein were the first to address consensus RNA secondary structure prediction using a consistent probabilistic model that combines an explicit stochastic evolutionary model with a stochastic context-free grammar based probabilistic model of structure plausibility. They find a consensus structure that optimizes a joint probability of the structure and multiple aligned homologous sequences [100, 102].

The stochastic evolutionary model of Knudsen and Hein is a general stationary time-reversible model [106, 163]. It can be regarded as a time-continuous Markovian process characterized by a rate matrix \mathbf{R} . The probabilities of the process for time t units are generally given by $\mathbf{Q}(t) = \exp^{\mathbf{R}t}$. Knudsen and Hein utilize this model for both the unpaired nucleotides and base paired nucleotides [100, 125, 157]. The model treats any observed change in a base pair as a single mutation, a simplification that appears valid for at least

rRNA [166]. In more recent work, Knudsen et al. show that the rate matrix approach to parameterizing base pairing is robust [99].

The Knudsen and Hein approach uses a class of probabilistic models called *stochastic context-free grammars* (SCFGs) that are a convenient basis for probabilistic modeling of non-pseudoknotted RNA structure [30, 33, 151]. The dynamic programming algorithms for working with SCFGs are essentially identical to the standard dynamic programming algorithms for RNA folding.

In addition to the Knudsen and Hein approach, at least three other SCFG-based approaches to RNA secondary structure prediction have been described. These include an SCFG-based mirror of the standard Zuker algorithm for single-sequence structure prediction [146], and two “pairSCFG” approaches for simultaneous folding and alignment of two homologous RNAs [80, 147]. All four groups use different underlying SCFG designs. No group appears to have explored different possible SCFG designs before settling on the one they used. Only Knudsen and Hein reported any benchmark results for the accuracy of their secondary structure predictions [100, 102]. It is not known how different designs affect the accuracy of SCFG-based secondary structure prediction.

1.6 The Sankoff Algorithm

The Knudsen and Hein approach requires a multiple sequence alignment as input. Unfortunately, to produce an accurate RNA multiple alignment requires knowledge of the secondary structure. Therefore, comparative approaches which require a multiple sequence alignment as input are problematic. Ultimately the problems of alignment and folding should be addressed simultaneously. Several approaches to this problem have been proposed.

One approach is to calculate the base pair probability matrix for every sequence and then seek to align these matrices by simulated annealing [93], a semi-manual approach [108], or by dynamic programming [70]. Unfortunately, this is related to the threading problem which is known to be NP-complete [105]. Others seek to find a set of shared helices between the sequences utilizing genetic

algorithms [17, 82, 137, 169], pattern recognition [9, 135], or a graph-theoretic approaches [87]. All of these approaches require heuristics to be made tractable.

In 1985, David Sankoff [152] proposed a dynamic programming algorithm to combine the problems of alignment and folding by optimizing a linear combination of the objective functions utilized when the problems are treated separately. Sankoff described the problem generally, detailing an algorithm which is $O(L^{3n})$ in time and $O(L^{2n})$ in space, where L is the length of the sequences and n is the number of sequences in the set. When the Sankoff algorithm is limited to two sequences, it is $\mathcal{O}(L^4)$ and $\mathcal{O}(L^6)$ in space and time respectively. The original paper was theoretical and well in advance of the compute resources available at the time.

A historical aside is worth mentioning. While Sankoff defined secondary structure as a set of base pairs, his simultaneous fold and alignment algorithm did not require the sequences to share a common secondary structure. Instead, he sought to identify a common “shape” between the sequences of interest, where shape was loosely defined as a function of the branching pattern of stems. In general, the issue of shapes is one of defining similarity metrics between structures [47, 124]. It is somewhat philosophical in nature and beyond the scope of this work.

It can be argued that any algorithm which considers folding and alignment simultaneously, calculating a structural alignment, is a variation of “the Sankoff algorithm”. Sankoff is a prolific and talented researcher. As such, “the Sankoff algorithm” is somewhat vague. For example, the Sankoff algorithm referred to here differs from the Sankoff algorithm discussed in Felsenstein’s *Inferring Phylogenies* [38]. In the context of this work, the Sankoff algorithm and structural alignment are interchangeable and refer to the work of [152].

The first practical implementation of the Sankoff algorithm was Gorodkin’s FOLDALIGN [49] which utilized a simple base pair maximization scoring scheme and did not permit multibranch loops. FOLDALIGN performs alignment between pairs of sequences and then does the multiple alignment through a greedy approach. The resulting algorithm is $O(L^4)$ in time and memory [49, 50].

Mathews and Turner utilize the nearest neighbor free energy parameters and permit multibranch loops, but largely ignore sequence homology information. Their implementation, Dynalign, is limited to pairwise alignment. Dynalign depends on three input parameters: an *ad hoc* pseudoenergy penalty for gaps ΔG_{gap}^0 , a maximum alignment offset \mathcal{M} , and an indicator for if single nucleotide pairs are permitted. Dynalign uses a banded alignment approach where a nucleotide at position i in one sequence aligns to j which is restricted to $(i - \mathcal{M}) < j < (i + \mathcal{M})$. When single nucleotide pairs are allowed, a single inserted base pair can be included uniquely in either structure if it is between two conserved base pairs. The algorithm is $O(\mathcal{M}^3 L^3)$ in time and $O(\mathcal{M}^2 L^2)$ in memory where L is the shorter sequence [119].

Holmes and Rubin describe a general pairwise stochastic context-free grammar approach that utilizes “fold envelopes” to restrict the set of base pairings allowed to be considered. Therefore, rather than considering all possible structures, the algorithm is limited to the set consistent with precomputed secondary structures [80]. Their original paper did not provide an implementation.

1.7 Outline of this work

The problem of interest in this work is the simultaneous alignment and folding of two structurally homologous sequences. My approach is to utilize a fully probabilistic framework. To do this requires first examining the problem of how to design a small simple probabilistic approach to single sequence RNA folding.

In Chapter 2 I outline the general algorithmics of probabilistic models, focusing on stochastic context-free grammars for secondary structure prediction. I also introduce the concept of structural ambiguity and discuss its implications. In Chapter 3 I explore the impact of a number of different SCFG designs on single sequence RNA secondary structure prediction accuracy. The goal is to identify a small compact SCFG design that can serve as the core for the development of an alignment and structurally unambiguous pairSCFG.

In Chapter 4 I extend a single sequence SCFG to the pairwise case and take a look at the performance of the full pairwise structural alignment in accuracy, runtime, and memory requirements. Finally, in Chapter 5 I describe an algorithm for constraining the structural alignment. My constraint algorithm depends on knowing a few positions (pins) of the alignment *a priori*. Consequently, I also outline a way to predict high quality pins. In the final chapter I take a look at future directions implied by this work.

Chapter 2

Probabilistic Models of RNAs

To develop a model of pairwise RNA structural alignment, we must first understand the individual problems of RNA secondary structure prediction and pairwise sequence alignment. Here we examine separately the algorithmics behind probabilistic models of RNA secondary structure and of pairwise sequence alignment. We then introduce the issue of model ambiguity with respect to both structure and alignment. This chapter outlines the algorithmic background relevant to this work.

2.1 Secondary Structure

Dynamic programming algorithms for non-pseudoknotted RNA secondary structure prediction work by calculating scores for optimal foldings for all subsequences $x_i \dots x_j$, starting with subsequences of zero length and working outward recursively on increasingly longer sequences [189]. For example, an example of an RNA folding algorithm [129] is:

Initialization:

$$\gamma(i, i - 1) = 0$$

$$\gamma(i, i) = 0$$

Iteration:

$$\gamma(i, j) = \max \begin{cases} \gamma(i + 1, j - 1) + \delta(x_i, x_j) \\ \gamma(i + 1, j) + \delta(x_i) \\ \gamma(i, j - 1) + \delta(x_j) \\ \max_{i-1 \leq a \leq j} \{ \gamma(i, a) + \gamma(a + 1, j) \} \end{cases}$$

$\delta(x_i)$ and $\delta(x_j)$ are scores for single stranded nucleotides and $\delta(x_i, x_j)$ are scores for base pairs. For a sequence of length L , the score calculation terminates when $\gamma(1, L)$ is calculated, the score of the best structure on the complete sequence. The optimal structure itself is retrieved by a traceback of the dynamic programming matrix.

When $\delta(x_i, x_j) = 1$ for base pairs and all other scores are zero, this algorithm finds the structure that maximizes the number of base pairs with the final score $\gamma(1, L)$ as the number of base pairs in the optimal structure [129]. In the more complicated loop-dependent algorithms (e.g. the Zuker algorithm), the algorithm is fundamentally the same (though with more terms, more matrices, and minimization instead of maximization); but the scores are energy parameters, and the final score is ΔG , the calculated thermodynamic stability of the optimal structure [189].

In a probabilistic approach, the algorithm again remains fundamentally the same, but the parameters are log probabilities (we call the parameter set Θ) and the final score is the log probability of the optimal structure $\hat{\nu}$ and the sequence \mathbf{x} given the model parameters, $\log P(\hat{\nu}, \mathbf{x} | \Theta)$. This is the quantity we need for statistical inference approaches [86, 111]. Probabilities must sum to one, so we have to be sure that for our model, $\sum_{\nu} \sum_{\mathbf{x}} P(\nu, \mathbf{x} | \Theta) = 1$ over all possible sequences \mathbf{x} and all possible structures ν for those sequences. But there are an infinite number of possible sequences and a combinatorial explosion of possible structures, so we can't enumerate all these possibilities; we need a formal system to be confident that we can form a correct probabilistic model.

Stochastic context-free grammars (SCFGs) are one such formal system. SCFGs are probabilistic models capable of capturing the long range, nested, pairwise correlations, such as those induced by base pairing in non-pseudoknotted

RNA secondary structures [30, 33, 151]. Here we give a somewhat informal introduction to SCFGs, specifically as they apply to RNA folding, starting with (nonstochastic) context-free grammars. For a review of the use of SCFGs for RNA folding, see [30]. For more formal descriptions of CFGs, see [62, 81, 156].

2.1.1 Context-Free Grammars

A context-free grammar \mathcal{G} can be defined by $\mathcal{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, S)$ where:

- \mathbf{V} is a finite set of nonterminal symbols (“states”),
- \mathbf{T} is a finite set of terminal symbols (for RNA: $\{a, c, g, u\}$),
- \mathbf{P} is a finite set of *production rules* (described below), and
- S is the initial (start) nonterminal ($S \in \mathbf{V}$).

Production rules describe how the grammar generates an observed symbol string in steps, starting from the initial start nonterminal S . Production rules take the form $A \rightarrow \alpha$ where $A \in \mathbf{V}$ and α is any combination of terminals \mathbf{T} and/or nonterminals \mathbf{V} . By convention, capital letters denote nonterminal symbols and lower-case letters are terminals. $S \rightarrow gSc$ is an example of a CFG production rule; it generates a ‘g’ and a ‘c’ terminal symbol in one correlated step. The ability to generate or score two or more correlated symbols in a single step is what gives CFGs the power to deal with base pairing.

At each step of a *production* from the grammar, one has a current string of terminals and/or nonterminals; one chooses a nonterminal, and transforms that nonterminal into a new substring using a valid production rule. This process starts with the initial string S and iterates until one arrives at a string consisting solely of terminal symbols.

For example, consider a “palindrome grammar” $\mathbf{V} = \{S\}$, $\mathbf{T} = \{a, b\}$, and $\mathbf{P} = \{S \rightarrow aSa, S \rightarrow bSb, S \rightarrow \epsilon\}$, where ϵ is a null string used as an ending production¹. This little grammar generates only strings consisting of palindromes

¹Rules such as $A \rightarrow \epsilon$ are shrinking productions. Technically this is only a legal production in a Turing machine (an unrestricted grammar). The grammars presented here can always be

of a and b symbols. RNA base pairing is essentially palindromic, except pairings are complements rather than identities. An example production is $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbSbba \Rightarrow abbbba$, resulting in the string $abbbba$. Note that the grammar produces this string in an nested fashion and that this CFG efficiently describes the set of all palindromes over the alphabet $\{a, b\}$.

Now consider an example CFG that generates RNA structures: $\mathbf{V} = \{S\}$, $\mathbf{T} = \{a, c, g, u\}$, and $\mathbf{P} =$

$$\begin{aligned} S &\rightarrow aSu \mid uSa \mid cSg \mid gSc \\ S &\rightarrow aS \mid cS \mid gS \mid uS \\ S &\rightarrow Sa \mid Sc \mid Sg \mid Su \\ S &\rightarrow SS \\ S &\rightarrow \epsilon \end{aligned}$$

where ‘|’ represents ‘or’ between production rules. The productions can be rewritten in a shorthand as:

$$S \rightarrow aSa' \mid aS \mid Sa \mid SS \mid \epsilon$$

where we are now using a generically to represent any single terminal symbol in \mathbf{T} , and the rule $S \rightarrow aSa'$ implies a base pairs with a' . We could also allow G·U pairs here. In general we will have probability scores for all 16 base pairs including noncanonical ones, or in the case of single sequence grammars that take base pair stacking into account, the full 16x16 matrix of possibilities.

A CFG derivation has an elegant representation known as a *parse tree*, π . Figure 2.1 shows an example RNA structure and two example parse trees for the RNA CFG. Given appropriate production rules, a parse tree has a natural correspondence with an RNA secondary structure.

Nonstochastic CFGs are used in pattern search applications, where one represents an RNA structural consensus as a CFG and ask if a particular sequence matches or doesn’t match the query. They are not particularly useful

expanded to eliminate the ϵ rule, so they do not present a problem. See [30] or [81, 156] for detailed treatment.

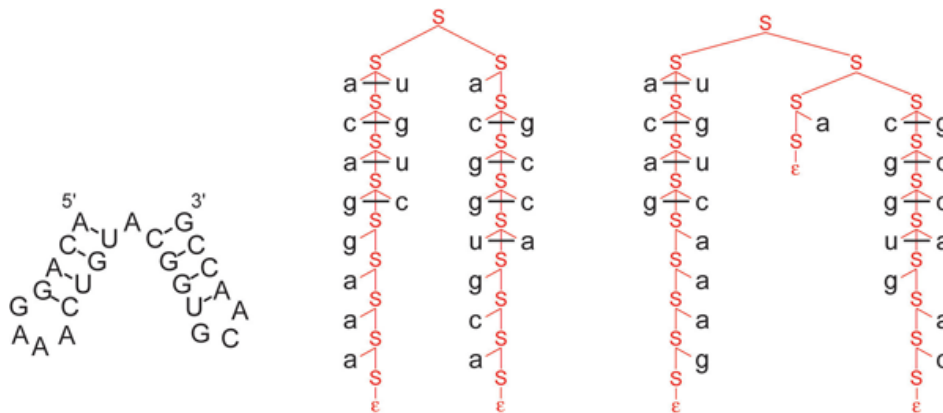


Figure 2.1: **Examples of CFG parse trees for an example RNA structure.** Left: an RNA secondary structure with two stems. Middle: a parse tree for that structure using the grammar $S \rightarrow aSa' \mid aS \mid Sa \mid SS \mid \epsilon$, with nonterminals in red and terminals in black. Note the correspondence between the RNA structure and the structure of the parse tree; that the individual steps in the grammar correspond to base pairs and single nucleotides (that is, the grammar is used to factor the structure down into individually scored steps); and that the RNA sequence can be read off the parse tree by following the margin of the tree counterclockwise. Right: an alternative parse tree for the same structure, demonstrating that this grammar is *structurally ambiguous*.

for structure prediction. For the CFG above, for example, for any RNA sequence there will be a huge number of valid parse trees, each of which corresponds to a possible RNA secondary structure. However, our problem in structure prediction is not to determine whether an RNA sequence has at least one possible structure. Given a sequence, we want to score and rank the possible parse trees for that sequence to infer the optimal one. To score and rank parse trees, we need to use *stochastic* context free grammars. In addition, we need efficient algorithms for finding the optimal SCFG parse tree for a given sequence.

2.1.2 Stochastic context-free grammars

In an SCFG \mathcal{G} , we associate a probability with each CFG production rule. The probabilities of the set of productions from any given nonterminal must sum to one. We refer to the full set of probabilities (the parameters of a model) as Θ .

The probability $P(\mathbf{x}, \pi \mid \mathcal{G}, \Theta)$ is the product of all the probabilities of the production rules used in a parse tree π for sequence \mathbf{x} . An SCFG is a probabilistic model that describes a joint probability distribution $P(\mathbf{x}, \pi \mid \mathcal{G}, \Theta)$ over all RNA sequences \mathbf{x} and all possible parse trees π .

Given a parameterized SCFG (\mathcal{G}, Θ) and a sequence \mathbf{x} , the Cocke-Younger-Kasami (CYK) dynamic programming algorithm finds an optimal (maximum probability) parse tree $\hat{\pi}$ for a sequence \mathbf{x} ,

$$\hat{\pi} = \operatorname{argmax}_{\pi} P(\pi, \mathbf{x} \mid \mathcal{G}, \Theta).$$

For notational and formal reasons, the CYK algorithm is usually described for SCFGs in “Chomsky normal form” with only two kinds of production rules, $A \rightarrow AA$ and $A \rightarrow a$; any SCFG can provably be converted to a set of production rules of this form [30]. But in applications, it is convenient to avoid this conversion and express the CYK algorithm directly in terms of the grammar’s own production rules. For an RNA SCFG based on the example grammar in the previous section, the CYK algorithm is:

Initialization:

$$S(i, i - 1) = \log P(S \rightarrow \epsilon)$$

Iteration:

$$S(i, j) = \max \begin{cases} S(i + 1, j - 1) + \log P(S \rightarrow x_i S x_j) \\ S(i + 1, j) + \log P(S \rightarrow x_i S) \\ S(i, j - 1) + \log P(S \rightarrow S x_j) \\ \max_{i-1 \leq a \leq j} \{S(i, a) + S(a + 1, j) + \log P(S \rightarrow SS)\} \end{cases}$$

When the algorithm terminates, $S(1, L)$ is $\log P(\mathbf{x}, \hat{\pi} \mid \mathcal{G}, \Theta)$, the log probability of the most probable parse tree $\hat{\pi}$ for the sequence \mathbf{x} given the grammar \mathcal{G} and parameters Θ . A traceback recovers this optimal parse tree.

Take note of the near-exact correspondence between the CYK algorithm and standard dynamic programming algorithms for RNA folding (page 15). SCFG algorithms are essentially the same as existing RNA folding algorithms,

but the scoring system is probabilistic, based on factoring the score for a structure down into a sum of log probability terms, rather than factoring the structure into a sum of energy terms or arbitrary base pair scores. The thermodynamic scoring parameters for energy minimization are largely derived by experimental melting studies of small model structures [117]; in contrast, SCFG log probability parameters are derived from frequencies observed in training sets of known RNA secondary structures. That is, instead of scoring a G·C pair stacked on a C·G base pair by adding a term for the free energy contribution of the G·C/C·G stack, an SCFG would add a log probability that G·C/C·G stacks are observed in known RNA structures.

A related SCFG algorithm, the Inside algorithm, is used to obtain the total probability of the sequence given the model summed over all parse trees,

$$P(\mathbf{x}|\mathcal{G}, \Theta) = \sum_{\pi} P(\mathbf{x}, \pi|\mathcal{G}, \Theta).$$

The Inside algorithm replaces the max operations in CYK with sums and additions of terms with multiplications. It is analogous to the McCaskill algorithm for calculating partition functions using the thermodynamic model of RNA folding [30, 120]. Suboptimal parse trees can be sampled from their posterior probability distribution by a probabilistic traceback of the Inside matrix, analogous to techniques used for energy minimization algorithms [24, 25, 30].

The RNA secondary structure grammars we are interested in are limited to bifurcations, splitting rules such as $S \rightarrow SS$, that have not more than two nonterminals on the right side. For all grammars of this type, the CYK and Inside algorithms are $O(ML^2)$ and $O(ML^3)$ in memory and time respectively, for M total nonterminals in the grammar and a sequence of length L .

2.2 Sequence Alignment

As with structure prediction, the field of sequence alignment algorithms grew from dynamic programming formulations. A formal modeling system for

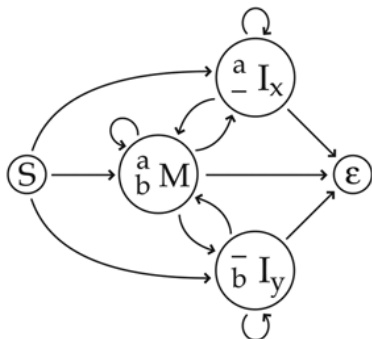
probabilistically describing sequence alignment is known as a hidden Markov model. Because HMMs are a deep and broad field of study, I will only scratch the surface here with emphasis on the aspects of sequence alignment relevant to the development of a model of structural alignment. For a review of HMMs in biological sequence analysis, see [31]. For a more detailed treatment refer to the Durbin book [30]. Finally, for a more formal description of regular grammars, see [62, 81, 156].

2.2.1 Pairwise Hidden Markov Models

A hidden Markov model is one method of modeling a stochastic regular grammar. Regular grammars are a subset of context-free grammars which are restricted to rules of the form $A \rightarrow aA$, $A \rightarrow a$, or $A \rightarrow \epsilon$. Consequently, regular grammars can not efficiently model long range interactions such as base pairing. However, they are excellent models of primary sequence. Often, HMMs are not written grammatically but rather pictorially as finite state automata.

We are particularly interested in HMMs designed to align two sequences. By convention [30] an HMM which emits a pairwise alignment is referred to as a pairHMM. A pairHMM is a probabilistic model that describes a joint probability distribution $P(\mathbf{x}, \mathbf{y}, \pi \mid \mathcal{G}, \Theta)$ over all possible parse trees π between sequence \mathbf{x} and sequence \mathbf{y} .

Figure 2.2 shows a finite state automaton representation for a typical pairHMM description of sequence alignment. The automaton representation assumes a Moore machine in which emissions are associated with states. In a Moore machine, transitions between states and emissions from states are typically treated separately. The grammar notation used thus far to describe SCFGs assumed Mealy machines, systems that emit on transition. In a Mealy machine, the rules of the grammar effectively lump together transition and emission into one probability distribution. In fact, these two representations are interconvertible [81, 156]. In the grammatical notation for this model, we define the start terminal as M .



$$\begin{aligned}
 M &\rightarrow \begin{matrix} a \\ b \end{matrix} M \mid \begin{matrix} a \\ - \end{matrix} I_x \mid \begin{matrix} - \\ b \end{matrix} I_y \mid \epsilon \\
 I_x &\rightarrow \begin{matrix} a \\ b \end{matrix} M \mid \begin{matrix} a \\ - \end{matrix} I_x \mid \epsilon \\
 I_y &\rightarrow \begin{matrix} a \\ b \end{matrix} M \mid \begin{matrix} - \\ b \end{matrix} I_y \mid \epsilon
 \end{aligned}$$

Figure 2.2: **Example pairHMM.** On the left is a typical finite state automaton representation for a pairHMM. The arrows represent the transitions between states and the circles represent the states, or nonterminals. Emissions are associated with each state, which can emit zero or more symbols. On the right is the same pairHMM represented in grammatical notation. By convention, grammars written in this notation are typically said to emit on transition. A state-based automaton, as on the left, is called a Moore machine and a transition based system, as on the right, is called a Mealy machine. In the Mealy notation for this model, we define the start terminal as M .

We can write the pairHMM analog to CYK, known as Viterbi, as follows:

$$\begin{aligned}
 M(i, k) &= \max \left\{ \begin{array}{l} M(i-1, k-1) + \log P \left(M \rightarrow \begin{matrix} x_i \\ y_k \end{matrix} M \right) \\ I_x(i-1, k) + \log P \left(M \rightarrow \begin{matrix} x_i \\ - \end{matrix} I_x \right) \\ I_y(i, k-1) + \log P \left(M \rightarrow \begin{matrix} - \\ y_k \end{matrix} I_x \right) \end{array} \right. \\
 I_x(i, k) &= \max \left\{ \begin{array}{l} M(i-1, k-1) + \log P \left(I_x \rightarrow \begin{matrix} x_i \\ y_k \end{matrix} M \right) \\ I_x(i-1, k) + \log P \left(I_x \rightarrow \begin{matrix} x_i \\ - \end{matrix} I_x \right) \end{array} \right. \\
 I_y(i, k) &= \max \left\{ \begin{array}{l} M(i-1, k-1) + \log P \left(I_y \rightarrow \begin{matrix} x_i \\ y_k \end{matrix} M \right) \\ I_y(i, k-1) + \log P \left(I_y \rightarrow \begin{matrix} - \\ y_k \end{matrix} I_y \right) \end{array} \right.
 \end{aligned}$$

when the algorithm terminates, $M(M, N)$ is the $\log P(\mathbf{x}, \mathbf{y}, \hat{\pi} \mid \mathcal{G}, \Theta)$, the log probability of the most probable parse tree $\hat{\pi}$ for the sequences \mathbf{x} and \mathbf{y} (lengths M and N , respectively) given the grammar \mathcal{G} and parameters Θ . A traceback recovers the best alignment. Note that this description of the algorithm is based on the grammar notation, for the automata based description of the algorithm see [30].

The direct analogy between the Viterbi algorithm for pairHMMs and the CYK algorithm for SCFGs should be readily apparent. The Inside algorithm analog in HMMs is known as the Forward algorithm. We can calculate suboptimal alignments by sampling from the posterior probability distribution of parse trees by a probabilistic traceback of Forward. In general HMM algorithms are simpler than their SCFG analogs. Viterbi and Forward algorithms are typically $O(ML^2)$ in memory and time, for M total nonterminals in the grammar and a sequence of length L . Algorithms exist to reduced the memory requirements to $O(ML)$ for a slight (roughly 2-fold) increase in runtime.

2.2.2 Optimal Accuracy Alignment

The probability of any single alignment being entirely correct is relatively small. Often parts of the alignment are reliable, whereas other sections are more uncertain. If we can assess the reliability of each column in the alignment, we can determine which portions are solid. Reliable portions of a primary sequence alignment between two RNAs may represent regions where the sequence is conserved. This information could then be utilized as constraint information to the structural alignment problem.

Posteriors probabilities are the desired reliability measure. Given a fully probabilistic model of sequence alignment, we can calculate $P(x_i \diamond y_k)$, the posterior probability that x_i is aligned to y_k . To do so requires both the Forward and Backward algorithms. Forward calculates the collective probability of all alignments up to and including the current position of interest. Backward is the counterpoint to Forward, calculating the collective probability of the remainder of the sequence given the position of interest. For the details of the Backward algorithm, refer to [30, 141].

With the posteriors, we can now calculate the alignment with the highest predicted accuracy, known as the optimal accuracy alignment [76, 79]. The method is a Viterbi algorithm which utilizes the posteriors as scores of alignment columns:

$$A(i, j) = \max \begin{cases} A(i + 1, j - 1) + P(x_i \diamond y_k) \\ A(i + 1, j) + P(x_i \diamond -) \\ A(i, j - 1) + P(- \diamond x_j) \end{cases}$$

where $P(x_i \diamond -)$ and $P(- \diamond x_j)$ indicate the probability of positions x_i and x_j being an indel and a standard traceback recovers the alignment.

It is worth noting that SCFGs have an analog to Backward known as the Outside algorithm. Using Inside and Outside, $P(x_i \sqcap x_j)$ the posterior probability of a base pair $x_i \cdot x_j$ can be calculated. The sum of this probability over all possible j would calculate the probability of i being paired. Using a CYK algorithm with base pair posteriors as scores, the optimal accuracy fold can be calculated. This is, in fact, the structure prediction method utilized by Pfold [103].

Experience shows that the optimal accuracy method of alignment and folding tend to be conservative. In other words, a position in the alignment (base pair in the fold) is incorporated into the alignment (folding) only if the model has strong evidence of it being correct. The result is that this approach tends to under predict, but with greater accuracy in the predicted areas than conventional Viterbi (CYK) [101, 116].

2.2.3 Evolutionary Models

It can be argued that any alignment should take into account the evolutionary relationship between the sequences. Two closely related sequences are expected to show more similarity and fewer indels than more distantly related sequences. The parameters should reflect these expectations. For closely related sequences, structural alignment should emphasize primary sequence alignment but at longer distances the emphasis should shift more toward conserved

structure. Estimating the evolutionary distance between two RNA sequences requires a stochastic model for nucleotide substitutions.

Time dependent models of substitution treat the alphabet as states in a Markov chain [68]. Transitions between the states correspond to substitutions and as time progresses the chance of a substitution increases. Let \mathbf{R} be a rate matrix whose ij th element r_{ij} is the substitution rate from nucleotide i to nucleotide j for $i \neq j$; the diagonal elements are given by $r_{ij} = -\sum_{j \neq i} r_{ij}$. By the Markov process, the matrix at time t units is generally given by:

$$\mathbf{Q}(t) = \exp^{t\mathbf{R}}$$

where the ij th element of $\mathbf{Q}(t)$ is $Q_{j|i}(t)$, the probability i has been substituted with j by t time units. Generally models of this sort are assumed to be stationary and time reversible. Stationary means that the expected nucleotide frequencies in the sequence do not change over time and are denoted by τ_i where $i \in \text{alphabet}$ ². Reversibility means $\tau_i r_{ij} = \tau_j r_{ji}$ holds for any i and j .

A number of classic models are special cases of this model [38, 132]. The Jukes and Cantor model [91] assumes equal probability for all possible substitutions. The Kimura two-parameter model [95] assumes transitions have a different probability than transversions. In both models the base composition of the sequences is expected to be uniform. The Hasegawa, Kishino, and Yano model extends the Kimura two-parameter model to unequal base composition [63].

Given a rate matrix \mathbf{R} , the joint probabilities P_{ij} can be readily calculated from the conditional probabilities $Q_{j|i}$ using $P_{ij} = Q_{j|i}\tau_i$ where $Q_{j|i}$ is calculated at time t and τ_i are the stationary probabilities. The joint probabilities are precisely the values of interest to describe a pairHMM's alignment distribution. Consequently, the parameters of the pairHMM can be tuned to the evolutionary distance implied by the sequences of interest.

²Be aware that I do not use the standard symbols of the literature on stochastic evolutionary models. I opted to redefine the symbols to avoid conflict with other definitions in this document.

2.3 Ambiguity

In formal language theory, a grammar is said to be *ambiguous* when there is more than one possible parse tree for some sequence [62, 81, 156]. Any SCFG that will be useful for RNA secondary structure prediction must be ambiguous in this sense. We are interested in looking at all possible base paired structures for the sequence, represented as a set of possible parse trees, and finding the optimal (highest probability) one.

2.3.1 Structural ambiguity

A different form of grammar ambiguity concerns us here. The CYK algorithm finds the mathematically optimal parse tree $\hat{\pi}$; we want the optimal secondary structure $\hat{\nu}$. We consider two structures to be identical if they have the same set of base pairs. The optimal parse tree gives us the optimal structure if and only if there is a one to one correspondence between parse trees and secondary structures. However, a given secondary structure does not necessarily have a unique parse tree. For instance, consider the two possible parse trees for the example in Figure 2.1, both of which express the same set of base pairs but use different series of production rules. When multiple valid parse trees describe the same secondary structure, we call the grammar *structurally ambiguous*. If a grammar is structurally ambiguous, then we cannot equate the probability of a parse tree with the probability of its structure [46]. The probability of a structure is a sum over the probabilities of all parse trees consistent with that structure. This summation is not reconcilable with the CYK algorithm; an optimal structure cannot be calculated efficiently if we need to do the summation over multiple possible parse trees for each structure. Thus, we will either have to use grammars that are structurally unambiguous (see Chapter 3), or we will have to assume that it is a valid approximation to consider an optimal parse tree gives us the optimal structure.

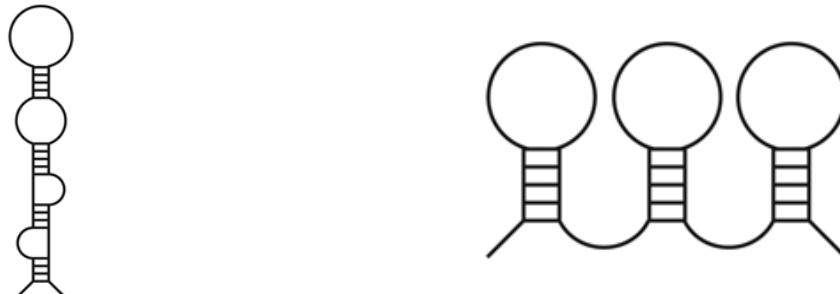


Figure 2.3: **Sources of structural ambiguity in single sequence.** Left: The stem structure is an example which contains all loop types possible in a stem, from the bottom up they are bulges (left and right), an internal loop, and a hairpin. We need $S \rightarrow aS$ to model a left bulge and $S \rightarrow Sa$ to model a right bulge. These same rules are utilized to describe both interior loops and hairpins. But in these cases, the ordering of these rules is unspecified. Right: The stem structure shows the other common source of ambiguity. When a structure presents multiple stems, a symmetric splitting rule such as $S \rightarrow SS$ does not distinguish between the case where the first nonterminal on the right handles one or two stems. Asymmetric bifurcations, $S \rightarrow LS$ can be carefully designed so as to avoid this ambiguity.

P(structure)

Figure 2.3 shows the two main sources of ambiguity in single sequence SCFGs. It can be tricky to write grammars that are structurally unambiguous, and it appears to be difficult to prove that they are so, except in simple cases.

We can, however, test empirically whether a particular grammar is unambiguous for a given sequence \mathbf{x} and structure ν . We do this with a *conditional Inside* algorithm that calculates:

$$P(\mathbf{x}, \nu | \mathcal{G}, \Theta) = \sum_{\pi \in \mathcal{C}(\nu)} P(\mathbf{x}, \pi | \mathcal{G}, \Theta)$$

by limiting the Inside calculation to $\mathcal{C}(\nu)$, those parse trees consistent with the given structure ν . For example, the conditional Inside algorithm for the RNA grammar described previously (page 18) is:

Initialization:

$$\gamma(i, i - 1) = P(S \rightarrow \epsilon)$$

Iteration:

$$\gamma(i, j) = \sum \begin{cases} \gamma(i+1, j-1) * P(S \rightarrow x_i S x_j) & \text{if } x_i, x_j \text{ paired in } \nu \\ \gamma(i+1, j) * P(S \rightarrow x_i S) & \text{if } x_i \text{ unpaired in } \nu \\ \gamma(i, j-1) * P(S \rightarrow S x_j) & \text{if } x_j \text{ unpaired in } \nu \\ \sum_{i-1 \leq a \leq j} \{ \gamma(i, a) * \gamma(a+1, j) * P(S \rightarrow SS) \} & \end{cases}$$

If $P(\mathbf{x}, \hat{\pi} | \mathcal{G}, \Theta)$ calculated by CYK equals $P(\mathbf{x}, \nu | \mathcal{G}, \Theta)$ calculated by conditional Inside then there is only one parse tree, namely $\hat{\pi}$, of nonzero probability for the structure ν . Note that this is an empirical test for a single sequence/structure pair. If this condition is maintained for a large number of sequences and structures, we can be reasonably confident that the grammar is unambiguous.

For an unambiguous grammar there is only one valid path (by definition) so conditional Inside simplifies to a scoring algorithm. In other words, given a structure ν , $P(\mathbf{x}, \nu | \theta)$ is the standard scoring algorithm for the parse tree π_ν . The structure ν can be thought of as a labeling of base pairs. Because it is ultimately this labeling that is the aim of structure prediction stochastic context-free grammars, unambiguous probabilistic context-free grammars may outperform their ambiguous counterparts.

An interesting difference between the thermodynamic and probabilistic approaches with respect to ambiguity is worth noting. The thermodynamic scoring scheme is not normalized, so structural ambiguity is not an issue for finding optimal structures; regardless of how many different ways there are of scoring the energy of a structure, the lowest energy structure still wins. However, ambiguity becomes a painstaking issue for calculating the equilibrium partition function [120], where one must be careful not to count any structure more than once. For SCFG-based methods, with normalized probabilities as scores, exactly the opposite is the case. Ambiguity is an issue for optimal structure prediction (i.e. interpreting $\hat{\pi} = \operatorname{argmax}_\pi P(\mathbf{x}, \pi | \Theta, \mathcal{G})$ as $\hat{\nu}$) but the Inside calculation (the analog of the summed partition function calculation) gives the correct result (i.e. $P(\mathbf{x} | \mathcal{G}, \Theta)$) even for ambiguous grammars.

2.3.2 Alignment ambiguity in pairHMMs

A similar ambiguity problem exists in sequence alignment. Two alignments are considered identical if they share the same set of aligned pairs, columns in the alignment which do not contain a gap character. Consider these two alignments:

$$\begin{array}{cc} \mathbf{xx-x} & \mathbf{x-xx} \\ \mathbf{y-yy} & \mathbf{yy-y} \end{array}$$

where x and y are any sequence character and $-$ is a gap character. It can be argued that these alignments are the same, differing only in the ordering of gaps in an indel region.

Consider a labeling strategy where ‘ c ’ is the consensus character (one of 16 possible) used to identify columns without gaps and ‘ \bullet ’ identifies columns containing a gap. Under such a labeling, both of these alignments can be described by ‘ $c_1 \bullet \bullet c_2$ ’. Two pairwise alignments which are identical (share the same alignment columns) and simplify to the same labeling are said to be *alignment ambiguous*.

We can test empirically for alignment ambiguity for a particular grammar on a given pair of sequences \mathbf{x} \mathbf{y} and alignment v . We do this with a *conditional Forward* algorithm that calculates:

$$P(\mathbf{x}, \mathbf{y}, v | \mathcal{G}, \Theta) = \sum_{\pi \in \mathcal{C}(v)} P(\mathbf{x}, \mathbf{y}, \pi | \mathcal{G}, \Theta)$$

by limiting the Forward calculation to $\mathcal{C}(v)$, those parse trees consistent with the given alignment v , specified by the labeling. This test is rarely employed in pairHMMs because alignment ambiguity can be easily avoided entirely. In our pairHMMs this is done by the assumption that a deletion can not immediately follow an insertion, represented in Figure 2.2 by the fact that no transition arrows connect I_x with I_y .

Structural and alignment ambiguity are similar problems. We seek a one-to-one correspondence between the grammar’s parse tree, π , and a labeling. In the case of structure prediction, that labeling is the identification of base

pairs. In alignment it is the identification of a consensus sequence. The importance of ambiguity depends entirely on what kinds of inference sought. In other words, if a grammar is only going to be used to calculate the probability of the sequence(s), $P(\mathbf{x} \mid \mathcal{G}, \Theta)$ or $P(\mathbf{x}, \mathbf{y} \mid \mathcal{G}, \Theta)$, then ambiguity is not an issue. However, if one is making statistical inference on secondary structure or sequence alignment, then ambiguity is arguably a concern.

2.3.3 Consensus Structure

We must consider additional sources of structural ambiguity when contemplating extending a single sequence SCFG to the pairwise case. In structural alignment, the information gained by utilizing a second sequence is the presence of conserved pairs, $\begin{matrix} x_i & S & x_j \\ y_k & & y_l \end{matrix}$ where this notation implies x_i aligns to y_k and base pairs with x_j . Similarly y_l aligns to x_j and base pairs with y_k . Consensus secondary structure is defined as the set of shared base pairings, quadruples. Pairwise SCFGs are discussed in more detail in Chapter 4.

One might imagine a pairSCFG in which base pairs are permitted in one sequence and not the other. However, such a grammar will be structurally ambiguous. Consequently, the pairSCFGs we consider will only consider a base pair if it is conserved. This is not an undue restriction as one can calculate the consensus secondary structure predicted by structural alignment and then use the predicted consensus base pairs as constraints to a single sequence folding algorithm. In this manner the substructures which are unique to our sequence of interest could be identified.

2.4 Concluding Comments

To calculate the structural alignment requires a model of both RNA folding and alignment. We propose to utilize a pairSCFG, essentially a stochastic context-free grammar which emits a correlated pair of sequences in a manner analogous to a pairHMM. It is straight forward to extend the algorithms of single sequence SCFGs to the case of a pairSCFG. The result is $O(ML^6)$ in time and

$O(ML^4)$ in memory for two sequences of length L and a grammar with M nonterminals.

Consequently, we seek to develop a pairwise SCFG for calculating the structural alignment. This can be accomplished by extending a single sequence SCFG to the pairwise case. However, at this point it is not known how different designs affect the accuracy of SCFG-based secondary structure prediction. Design decisions are particularly important in consensus structure prediction applications, because a natural trade-off arises. A complex RNA folding SCFG might predict structure better than a simpler model, but extending a complex model to deal with multiple evolutionarily correlated sequences can easily result in a combinatorial explosion of parameters, making the model impractical.

Chapter 3

Single Sequence RNA Secondary Structure Prediction

As already stated, the pairSCFG approach to structural alignment is $O(ML^6)$ in time and $O(ML^4)$ in memory for a grammar with M nonterminals. A grammar with fewer nonterminals will require less memory and time than a more complicated grammar, however, a more complicated grammar may better be able to model RNA secondary structure. One wants to build consensus prediction models on top of small simple SCFG designs that sacrifice as little RNA structure prediction accuracy as possible, relative to state-of-the-art energy minimization approaches.

A potential advantage of a probabilistic modeling approach to secondary structure prediction is that it is more readily extended to include other sources of statistical information that constrain a structure. Flexibility in model design comes from the fact that SCFG probability parameter estimation can be done by counting frequencies in databases of trusted RNA secondary structures, so it is easy to parameterize different models that vary in complexity and capture different features of RNA structure. In contrast, energy minimization algorithms are based on a standard set of thermodynamic parameters, most of which are determined experimentally [118, 189], so it would take substantial effort to develop a radically new thermodynamic model.

It is not known how different designs affect the accuracy of SCFG-based secondary structure prediction. In this chapter we explore the impact of different SCFG designs on single-sequence RNA secondary structure prediction accuracy. Our goal is to identify a small simple SCFG model design that can serve as the core to a more complex integrated approach.

3.1 Overview

The simplest RNA SCFGs factor a secondary structure into scoring terms for each individual base pair and each individual unpaired residue. We will examine five grammars of this type. However, state of the art thermodynamic models use a loop-dependent thermodynamic model that factors a structure in a more complex way, into nearest-neighbor base stacking terms (as opposed to individual base pairs) and tables of penalties for different lengths of different kinds of loops (bulge, interior, hairpin, and multibranch). SCFG methods can also be designed to capture more sophisticated folding features.

Base pair stacking is a first order Markov dependence. It can be modeled in a grammar by capturing a limited amount of context dependency, a technique called *lexicalization* in natural language processing. One uses production rules of the type, $P^{bb'} \rightarrow aP^{aa'}a'$, where the probability of emitting a new pair $a \cdot a'$ is dependent on the previous base pair $b \cdot b'$. The notation $P^{aa'}$ indicates that the P nonterminal will “remember” that it just generated an $a \cdot a'$ pair. Formally, in the production rules this means 16 $P^{aa'}$ nonterminals, one for each pair of nucleotides, each one of which can generate 16 possible pairs: thus, we have the desired 16x16 table for base stacking parameters in terms of our production rules. However, lexicalized nonterminals do not incur any extra storage nor additional computational cost when parsing, because the lexicalization depends completely on the local sequence context; the 16 $P^{aa'}$ nonterminals can just be treated as one “meta” P nonterminal, with scores conditioned on the local context around the base pair $x_i \cdot x_j$. In this chapter we will consider four grammars of this type.

The simplest (and most common) model of loop lengths in a grammar uses recursive rules like $S \rightarrow aS$, which imply a geometric length distribution.

Explicit loop length distributions can be modeled by enumerating a set of production rules, one for each possible length. A set of such loop length probabilities is no different in effect than the lookup table of loop length penalties in the thermodynamic parameters. Unfortunately, modeling explicit loop lengths results in larger, more complex grammars which would explode in parameter space when made pairwise. Therefore, all the grammars we explore utilize the simple geometric loop distribution rather than the more involved explicit loop length distributions.

Base stacking and explicit loop lengths are the most important two features of SCFGs that would closely mirror the current thermodynamic model. Most other desirable features, such as parameterizing stacked terminal mismatches for hairpin and internal loops, dangled bases, and special stable tetraloops, can also be dealt with using techniques like the above. Coaxial stacking [93,172] can also be accommodated [145], but adds to complexity.

3.2 Impact of Structural Ambiguity

It seems possible that structural ambiguity might be only a formal concern. This would be the case if, in a set of possible parse trees for any particular structure, one parse tree has a probability that dominates the others, and approximates the summed probability of the ensemble. For example, many of the alternative parse trees are pathological cases that invoke fruitless cycles of bifurcation and destruction; any $S \Rightarrow \epsilon$ derivation in a parse tree could also be $S \Rightarrow SS \Rightarrow \epsilon S \Rightarrow \epsilon\epsilon$, but these more elaborate trees have probabilities that asymptote toward zero. Additionally, our parameterization of ambiguous grammars tends to be biased toward concentrating probability in a single possible parse tree, because we nonrandomly choose only one possible parse tree for each training structure, favoring smaller, more sensible parse trees where ambiguous choices are possible. And finally, if all structures have about the same amount of probability diffusion to ambiguous parses, the rank order of the best parse trees might still reflect the rank order of the best structures, so that a CYK algorithm that finds the best parse tree would still recover the best structure.

We first implemented two structurally ambiguous grammars:

$$G1 : S \rightarrow aSa' \mid aS \mid Sa \mid SS \mid \epsilon$$

$$G2 : S \rightarrow aP^{aa'}a' \mid aS \mid Sa \mid SS \mid \epsilon \\ P^{bb'} \rightarrow aP^{aa'}a' \mid S$$

$G1$ is the simple grammar mentioned in Chapter 2 (Section 2.1.1). It is also the SCFG analog to Nussinov’s original algorithm [30, 129]. $G2$ extends it to include base pair stacking parameters.

To test whether structural ambiguity made a practical impact on structure prediction, we performed the following “reordering” experiment. For a given RNA sequence, we use CYK to find the optimal parse tree $\hat{\pi}$. We sample N more suboptimal parse trees from the posterior distribution (using Inside with stochastic traceback). We rank the $N + 1$ trees by their probabilities $P(\mathbf{x}, \pi | \mathcal{G}, \Theta)$, where the optimal CYK parse is ranked first by definition. Then for each parse tree π , we get the base-paired structure ν and use conditional Inside to calculate the $P(\mathbf{x}, \nu | \mathcal{G}, \Theta)$ summed over all parse trees for that structure. We then ask two questions: first, if the rank order of the $N + 1$ parse trees is the same as the rank order of their (up to) $N + 1$ structures; second, if the structure of the optimal CYK parse tree becomes suboptimal in the ranked list of structure probabilities $P(\mathbf{x}, \nu | \mathcal{G}, \Theta)$. If ambiguity does not matter in practice, we should not see many differences in rank order, and we hope to never see the top-ranked structure differ from the structure implied by the top-ranked CYK parse.

Rfam v5.0 [51] was utilized to build a large test set containing a wide variety of different RNA sequences. This set was constructed by filtering the seed alignments at 80% identity and then imposing the family’s consensus structure onto each sequence to infer individual secondary structures. Any sequence containing ambiguous bases was removed. The resulting dataset contains 2455 sequences from 174 different RNA families. We utilized this dataset for the reordering experiment, for N varying from 10 to 3000 suboptimal parse trees.

The results showed that even in a small number of samples, the rank order of the parse tree probabilities and the structural probabilities were nearly always significantly different. For $N = 10$ suboptimals, for grammar $G1$ a better structure than the CYK parse was found for 2% of the sequences (41/2455), and for $G2$, a better structure was found for 69% (1704/2455). Sampling deeper into the posterior probability distribution for parse trees increases the chance of finding a more optimal structure; for $N = 1000$ suboptimals, better structures than the CYK parse were found for about 20% of the sequences for $G1$ (495/2455), and 98% of the sequences for $G2$ (2417/2455).

One important aspect of this experiment is that it does not rely on a comparison between different grammars. The changes to the rules of a grammar necessary to disambiguate it can never be assured to produce a grammar which utilizes the same information within the training set. In other words, the grammar with more parameters may perform better just because there are more parameters. Consequently the reorder experiment is more conclusive that ambiguity affects the ability of a particular grammar to properly order potential structures than a direct comparison between two apparently similar grammars, one ambiguous and one unambiguous.

These results show that the CYK algorithm often does not give the optimal secondary structure if one is using a structurally ambiguous grammar. Structural ambiguity appears to be a practical concern. We therefore focused on developing several small simple unambiguous grammars.

As outlined in Chapter 2 (Section 2.3.1), we can empirically test for ambiguity by confirming that the score $P(\mathbf{x}, \hat{\pi} | \mathcal{G}, \Theta)$, calculated by CYK equals $P(\mathbf{x}, \nu | \mathcal{G}, \Theta)$ calculated by conditional Inside for the structure implied by $\hat{\pi}$. Operationally, we consider a grammar to be structurally unambiguous if this condition holds for all the sequences of our Rfam v5.0 dataset.

3.3 SCFG designs

It is something of a challenge to make unambiguous grammars with a small number of productions. We do not know of a way to systematically

generate and explore the space of unambiguous grammars. There appear to be many different ways to make unambiguous RNA grammars that have the same simple emission parameterization (4 probabilities for unpaired residues, 16 probabilities for base pairs). We used the following four unambiguous grammars:

$$\begin{aligned}
 G3 : \quad & S \rightarrow aSa' \mid aL \mid Ra \mid LS \\
 & L \rightarrow aSa' \mid aL \\
 & R \rightarrow Ra \mid \epsilon
 \end{aligned}$$

$$\begin{aligned}
 G4 : \quad & S \rightarrow aS \mid T \mid \epsilon \\
 & T \rightarrow Ta \mid aSa' \mid TaSa'
 \end{aligned}$$

$$G5 : \quad S \rightarrow aS \mid aSa'S \mid \epsilon$$

$$\begin{aligned}
 G6 : \quad & S \rightarrow LS \mid L \\
 & L \rightarrow aFa' \mid a \\
 & F \rightarrow aFa' \mid LS
 \end{aligned}$$

$G3$ was developed by RDD. We challenged Graeme Mitchison to make a smaller one, and he produced $G4$ (G. Mitchison, personal communication). The ultra compact $G5$ grammar is from Ivo Hofacker (personal communication). $G6$ is the Knudsen grammar utilized in the Pfold package [100,102]. Each of the four grammars was conjectured to be unambiguous by inspection. Each one also passed our empirical test for ambiguity using the test set of 2455 Rfam sequences.

Each grammar imposes slightly different restrictions on the “language” of possible structures. $G3$ imposes a minimum hairpin loop length of one nucleotide, $G6$ has a minimum of two, and $G4$ and $G5$ do not impose minimum hairpin loop lengths. Also, $G3$ and $G6$ can not emit an empty string, ϵ , whereas $G4$ and $G5$ can.

Figure 3.1 shows parse trees for an example RNA structure using these four grammars $G3$ - $G6$. The figure shows how each grammar factors a structure into elementary scorable steps in a different manner.

Table 3.1: **Simple grammar specifications** Each simple grammar requires a different number of nonterminals (NT) and parameters (with free parameters in parenthesis). Memory requirements were determined empirically by measuring the memory each grammar utilizes to fold a single *C. elegans* large subunit ribosomal RNA sequence (3662 nucleotides). Under “notes”, we give some of the implications each grammar has on the language it describes.

Grammar	NT	Parameters		<i>C. elegans</i> LSU Memory (MB)	Notes
		Total	Tied		
<i>G1</i>	1	29 (25)	25 (22)	26.9	ambiguous
<i>G3</i>	3	56 (47)	28 (23)	79.4	min 1 nt loop; no ϵ string
<i>G4</i>	2	46 (40)	26 (22)	53.2	none
<i>G5</i>	1	23 (20)	23 (20)	26.9	none
<i>G6</i>	3	42 (36)	26 (21)	79.4	min 2 nt loop; no ϵ string

Table 3.1 shows the resource requirements of these four grammars as well as the simple ambiguous grammar *G1*. Each nonterminal of a grammar requires storage of a dynamic programming matrix, so memory usage scales linearly with nonterminal number.

Stacking Grammars

The *G6* grammar is readily extended to include stacking parameters, by changing the *F* nonterminal parameterization to a first order Markov chain, $F^{bb'} \rightarrow aF^{aa'}a' \mid LS$. We call this grammar *G6^S*. We also restructured two of the simple backbones (*G3* and *G4*) to include base pair stacking parameters, resulting in grammars *G7* and *G8*:

$$\begin{aligned}
G7: \quad S &\rightarrow aP^{aa'}a' \mid aL \mid Ra \mid LS \\
L &\rightarrow aP^{aa'}a' \mid aL \\
R &\rightarrow Ra \mid \epsilon \\
P^{bb'} &\rightarrow aP^{aa'}a' \mid aNa' \\
N &\rightarrow aL \mid Ra \mid LS
\end{aligned}$$

$$\begin{aligned}
G8: \quad S &\rightarrow aS \mid T \mid \epsilon \\
T &\rightarrow Ta \mid aP^{aa'}a' \mid TaP^{aa'}a' \\
P^{bb'} &\rightarrow aP^{aa'}a' \mid aNa' \\
N &\rightarrow aS \mid Ta \mid TaP^{aa'}a'
\end{aligned}$$

Table 3.2 shows the general resource requirements for each of the stacking grammars, including the ambiguous $G2$ grammar described earlier. Grammars $G6^S$, $G7$, and $G8$ pass the empirical ambiguity test for the set of 2455 Rfam sequences, and are therefore considered unambiguous.

Table 3.2: Stacking Grammar Specifications. Stacking adds to grammars more nonterminals (NT) and total (free) parameters. This translates into increased memory requirements, as shown by the folding of *C. elegans* large subunit ribosomal RNA (3662 nucleotides). It also may introduce other restrictions such as not permitting lone pairs.

Grammar	NT	Parameters		<i>C. elegans</i> LSU Memory (MB)	Notes
		Total	Tied		
$G2$	2	287(281)	283(278)	53.2	ambiguous
$G7$	5	341(326)	289(281)	128	min 1 nt loop; no lone pairs; no ϵ string
$G8$	4	347(334)	287(280)	103	min 1 nt loop; no lone pairs
$G6^S$	3	282(276)	282(276)	79.4	min 2 nt loop; no ϵ string

3.4 Benchmarking

The parameters of each SCFG are estimated from frequencies observed in annotated secondary structures. The training data are large and small subunit rRNAs, obtained from the European Ribosomal Database [181,183]. Sequences containing more than 5% ambiguous bases or with less than 40% base pairing are discarded. The resulting data set was then filtered to remove sequences with greater than 80% identity. The final training set contains randomly chosen equal numbers of LSU and SSU sequences from the filtered data, totaling 278 sequences, 586,293 nucleotides, and 146,759 base pairs.

For a given grammar, a parse tree for each structure is determined from the secondary structure annotation, and the number of occurrences of each production type is counted. When determining a parse tree for an ambiguous grammar, one possible parse tree was arbitrarily chosen. Production probabilities are then estimated from these counts using a Laplace (plus-one) prior [30].

A benchmark data set was built from the Ribonuclease P database [11], the Signal Recognition Particle database [149] and the tmRNA database [193]. Each of these databases provides high quality *individual* sequence structure information and curated structural alignments in a readily parsable form. Each structural alignment was filtered to remove sequences with greater than 80% identity. The structure for each member of the filtered families were then retrieved from the databases. Any sequence containing ambiguous bases was removed. The resulting test set contains 403 total sequences, consisting of 225 RNase P's, 81 SRP's, and 97 tmRNA's.

We count the number of base pairs that are correctly predicted, given a predicted structure and the trusted benchmark structure. That is, if we predict a base pair $i \cdot j$, it is correct if and only if base pair $i \cdot j$ is present in the trusted structure. No helix offsetting/sliding is allowed, so our accuracy percentages are systematically lower than other published benchmarks that count base pairs as correct if they are within one base of a trusted base pair [115,118].

Predicted base pairs that are in the trusted structure are true positives (TP); predicted base pairs not in the trusted structure are false positives (FP);

base pairs in the trusted structure but not predicted by the algorithm are false negatives (FN). Sensitivity is the fraction of base pairs in the trusted structures that are predicted correctly: $TP / (TP+FN)$. Positive predictive value (PPV; sometimes called specificity) is the number of predicted base pairs that are in the trusted structure: $TP / (TP+FP)$.

Implementations

For benchmarking experiments, we used *mfold* v3.1.2, Pfold (Oct 2003), PKNOTS v 1.01, RNAstructure v4.0, and the Vienna RNA package v1.4. The *mfold* software was obtained from Michael Zuker at <http://www.bioinfo.rpi.edu/~zukerm/rna/mfold-3.0.html>. An early release of RNAstructure v4.0 [115] was graciously provided by Dave Mathews. Both *mfold* and RNAstructure were run with MAX=750 P=20 W=0 and *efn2* rearrangement as recommended by [118]. The Vienna RNA package was obtained at <http://www.tbi.univie.ac.at/~ivo/RNA/> and was run with default parameters. Bjarne Knudsen provided Pfold executables and guidance on how to run the package (personal communication). The PKNOTS program was run with default parameters, which does not attempt to predict pseudoknots. All benchmarks were conducted on Intel-based servers running a GNU/Linux operating system, with the exception of RNAstructure which was run in batch mode under Windows XP Pro 2002.

3.4.1 Grammar Comparison

In order to verify the CYK parsing code for each grammar, a control experiment was done with a non-stochastic scoring system that gives a score of +1 to each base pair. Under the plus-one scoring system, each grammar is forced to seek the structure with the maximal number of base pairs, and this optimal score must be identical for each grammar; this was verified. Under plus-one scoring, base pair sensitivity varied between 19 and 24% and PPV varied between 13% and 15% on the benchmark set of 403 RNase P's, SRP RNAs, and tmRNAs. The reason for the variation is that multiple different structures have the same optimal score, and an arbitrary parse is returned as the optimal traceback.

The secondary structure prediction accuracy of each of the nine parameterized grammars is measured, by calculating sensitivity and PPV on the benchmark set of 403 trusted secondary structures derived by comparative analysis.

To minimize performance differences due to differences in free parameter number, as opposed to grammar structure, we tied emission parameters together where possible. That is, each grammar uses only one emission distribution for 4 unpaired bases, 16 base pairs and, in the case of the stacking grammars, 256 stacking parameters. Thus the effective (tied) number of parameters only differs because of the number of transitions in each grammar (Tables 3.1 and 3.2). To minimize differences caused by the grammars having slightly different languages, we also forced each grammar to have the same minimum hairpin loop size (two) by implementing the appropriate steps in dynamic programming parsers to ignore hairpin loops of length one or zero, even if the grammar permits them. Maximum a posteriori parameters were then determined for each grammar using a set of annotated ribosomal RNA secondary structures as described.

For comparison, we also evaluated the performance of several implementations of energy minimization algorithms for predicting secondary structure on the same benchmark set: *mfold* v3.1.2, PKNOTS v1.01, RNAstructure v4.0, and the Vienna RNA package v1.4. Vienna RNA and *mfold* use the thermodynamic parameters described in [118] but use slightly different implementations of the rules for exterior and multibranch loops. RNAstructure uses a recently revised set of parameters [115], and PKNOTS uses an earlier set of thermodynamic parameters [154].

We benchmarked the Knudsen and Hein Pfold program, which independently implements the *G6* grammar. The Pfold program was parameterized on a different training set composed of rRNAs and tRNAs, and calculates the optimal accuracy folding, which increases PPV at the expense of sensitivity. Pfold is intended for consensus structure prediction, using an RNA evolutionary model to fold a given input RNA multiple alignment, but we have used it here in single-sequence mode to compare to our implementation and parameterization of *G6*.

Table 3.3: **Performance of single sequence grammars.** The first section contains simple grammars, the second contains stacking grammars, and the last section contains some other available software packages that predict secondary structure. The nine grammars were trained on rRNA as described in the text. The second column gives the performance on the full benchmarking dataset, which is subdivided into each family in the subsequent columns. The metrics are calculated over base pairs as described in the text, evaluating the metrics over individual sequences shows similar results (data not shown).

Benchmarking Set				
Sensitivity % (PPV %)				
Grammar	Full	RNase P	SRP	tmRNA
<i>G1</i>	16.48 (12.13)	13.52 (10.71)	37.37 (31.74)	10.00 (5.60)
<i>G3</i>	34.03 (30.61)	36.56 (34.74)	28.00 (28.18)	31.07 (22.46)
<i>G4</i>	9.75 (7.87)	9.52 (8.20)	19.00 (17.44)	3.49 (2.20)
<i>G5</i>	3.19 (3.51)	3.25 (3.74)	2.45 (3.04)	3.52 (3.17)
<i>G6</i>	47.43 (44.80)	48.55 (49.08)	47.31 (48.91)	44.43 (33.12)
<i>G2</i>	35.61 (24.91)	30.90 (23.28)	58.67 (47.74)	33.06 (17.45)
<i>G7</i>	45.33 (42.61)	45.82 (46.09)	50.28 (52.26)	40.33 (29.79)
<i>G8</i>	46.32 (43.46)	45.93 (46.04)	51.69 (53.42)	43.73 (32.41)
<i>G6^S</i>	48.61 (45.43)	49.94 (49.82)	49.76 (51.32)	43.94 (32.47)
<i>mfold</i> v3.1.2	56.02 (47.90)	56.13 (51.41)	69.50 (66.20)	45.69 (30.49)
Vienna RNA	54.90 (47.05)	55.22 (50.80)	67.20 (63.94)	44.81 (29.87)
PKNOTS	50.28 (41.20)	52.52 (46.01)	58.21 (54.65)	37.68 (23.87)
RNAstructure	56.06 (47.35)	57.87 (52.45)	61.97 (57.88)	46.17 (30.45)
Pfold	38.99 (68.93)	41.96 (75.64)	35.31 (64.12)	32.91 (53.66)

The results of the comparison are shown in Table 3.3. Among the four simple (nonstacking) unambiguous grammars, the simplest grammar, *G5*, with only three types of production rules, has an abysmal prediction performance. The grammar with the most rules, *G3*, did not give the best performance. The best simple SCFG design we tested is *G6*, the grammar used in Knudsen and Hein’s Pfold. Though *G6* does not perform quite as well as energy minimization methods, it does surprisingly well considering its simplicity. *G6*’s performance is nearly comparable to the performance of PKNOTS, which uses the older (1995) thermodynamic energy rules. *G6* has only 21 free probability parameters, compared to the several hundred thermodynamic parameters in energy minimization programs.

Performance for two of the grammars, $G3$ and $G4$, increased significantly when these grammars were remodeled to include stacking correlations, $G7$ and $G8$, respectively. Interestingly, $G6$ did not show any real performance gain when the parameters were extended to include stacking correlations ($G6^S$). With stacking correlations included, four of the grammars ($G6$, $G7$, $G8$, and $G6^S$) have comparably good performance, at the cost of increasing parameter number to include the 16x16 parameters set for the three with stacking correlations.

3.4.2 Stacking Variations

Grammars which handle stacking must decide if lone pairs (a base pair without a stacking partner) are permissible. When a closing base pair is explicitly modeled, $P \rightarrow aPa' \mid aNa'$, lone pairs are not permitted. The alternative is to permit lone pairs which is typically accomplished by $P \rightarrow aPa' \mid N$. As written thus far, grammars do not permit lone pairs. In some cases, it is also possible to completely eliminate the N nonterminal in a structurally unambiguous fashion. We utilize the $G8$ grammar to compare these alternative formulations. We refer to the version which permit lone pairs as $G8^{LP}$ and the version in which the N nonterminal is eliminated entirely, leaving simply $P \rightarrow aPa' \mid aS \mid Ta \mid TaP^{aa'}a'$, as $G8^{noN}$.

Because the $G6$ grammars showed limited improvement with the addition of stacking parameters, we wondered if the improvements observed in $G7$ and $G8$ relative to $G3$ and $G4$ are a result of the utilization of stacking parameters or because of the remodeling of the grammar. As such, both the $G7$ and $G8$ grammars were re-parameterized without the use of the 16x16 stacking matrix. Effectively this takes the P production from $P^{bb'} \rightarrow aP^{aa'}a' \mid aNa'$ to $P \rightarrow aPa' \mid aNa'$. We refer to the reduced alternative parameterizations as $G7^{nostack}$ and $G8^{nostack}$ respectively.

The performance of these stacking variations are shown in Table 3.4. The differences observed between grammars $G8$, $G8^{LP}$, $G8^{noN}$, who differ only in the method of closing a stem, were negligible. For RNase P and SRP in our benchmarking set we observe the increasing general order of $G8^{noN}$, $G8^{LP}$, $G8$. This ordering is maintained for the vast majority of RNA families in Rfam

Table 3.4: **Stacking Variations Performance.** The first section contains alternative stacking methods which modify the backbone of the grammar. The second section shows pairs of grammars, the first in each pair does not utilize the 16x16 stacking parameters compared to the second, but otherwise the pair maintain the same grammar backbone. In all cases the grammars were trained on rRNA as described in the text.

Benchmarking Set				
Sensitivity % (PPV %)				
Grammar	Full	RNase P	SRP	tmRNA
<i>G8</i>	46.32 (43.46)	45.93 (46.04)	51.69 (53.42)	43.73 (32.41)
<i>G8^{LP}</i>	45.65 (43.08)	45.47 (46.01)	50.88 (52.79)	42.30 (31.32)
<i>G8^{noN}</i>	45.79 (43.81)	45.00 (46.17)	50.76 (53.04)	44.48 (33.50)
<i>G6</i>	47.43 (44.80)	48.55 (49.08)	47.31 (48.91)	44.43 (33.12)
<i>G6^S</i>	48.61 (45.43)	49.94 (49.82)	49.76 (51.32)	43.94 (32.47)
<i>G7^{nostack}</i>	44.00 (41.43)	44.01 (44.38)	50.18 (51.33)	39.35 (29.34)
<i>G7</i>	45.33 (42.61)	45.82 (46.09)	50.28 (52.26)	40.33 (29.79)
<i>G8^{nostack}</i>	44.85 (42.23)	44.15 (44.54)	50.70 (51.65)	42.60 (31.81)
<i>G8</i>	46.32 (43.46)	45.93 (46.04)	51.69 (53.42)	43.73 (32.41)

(results not shown) despite tmRNA (and consequently our full benchmarking set) reversing the order of *G8^{noN}* and *G8^{LP}*. In all cases, the *G8* version shows slightly better performance than the two alternatives despite roughly one quarter of the sequences in our testset having annotated lone pairs.

More intriguing, the addition of stacking parameters appears to improve performance by only roughly one percent. This was unexpected. The fundamental paradigm of thermodynamic methods is the nearest-neighbor model [42, 117, 118, 168, 170, 185] in which the free-energy for forming a base pair depends on the adjacent pair. Consequently, modeling stacking was expected to be an important component to improving probabilistic models. Yet our results imply only minimal improvement with stacking parameters. This is consistently observed for three separate grammar backbones, *G6^S* relative to *G6*, *G7* relative to *G7^{nostack}*, and *G8* relative to *G8^{nostack}*.

The reason for the strong differences between different designs is probably related to how naturally the production rules of the grammar correspond to the biological constraints of RNA structures. The compact *G5* grammar, for instance, must invoke the same rule $S \rightarrow aSa'S$ for every base pair and for every

structural bifurcation, which are quite different structural features that occur with very different frequencies. The productions of $G5$ are thus “semantically overloaded”: they collapse too many different types of information into the same parameters. Looking at Figure 3.1, $G6$ and $G3$ arguably have the most natural parse trees, in the sense that they invoke one dedicated production per base pair (as opposed to two, as in $G4$ ’s stutter-step for each base pair, or the overloaded base pair/bifurcation production in $G5$).

3.4.3 Comparison to Mathews et al. benchmark

The sensitivity of *mfold* and RNAstructure was reported to be 73% on a different benchmark dataset [118], which is quite a bit higher than the 56% as obtained here. This concerned us, so before drawing any conclusions from our results, we did additional work to verify our benchmarking of the thermodynamic methods, using *mfold* as a representative method.

Dave Mathews kindly provided us the benchmark set used in [118]. Because some of the sources of his structure data were privileged, Mathews could not grant us permission to freely distribute this benchmark set (which is why we report performance results on a new benchmark set we are comfortable distributing). We measure *mfold* at 67% sensitivity on Mathews’ dataset instead of the 73% sensitivity reported in Mathews *et al.* [118]. Our procedure differs from theirs in two steps. First, we count only exactly correct base pair predictions, whereas Mathews *et al.* count a base pair as correct even if there is a slight (1 nt) slip in the prediction. Second, we report the sensitivity over all base pairs in the dataset (total correct base pairs / total base pairs predicted in the whole benchmark), whereas they calculate an average of the individual sequence base pair sensitivities. We found that each of these differences contributes about a 3% difference in the measured sensitivities, with our procedures systematically producing more conservative measures than Mathews et al.

The difference between the 56% we report and the 67% obtained by applying our method to Mathews’ dataset lies in the choice of secondary structures. Most importantly, prediction accuracy varies significantly from RNA family to RNA family. We have excluded tRNAs from our benchmark set

because they are small and reasonably predicted by both methods; the $G6^S$ grammar achieves roughly 80% sensitivity on the Mathews' tRNAs, and Mathews reports 83% for energy minimization [118]. Instead, we included tmRNAs, which tend to be difficult to predict ($mfold$ and $G6^S$ achieve around 45% sensitivity) because of the presence of pseudoknots which account for 21% of base pairs in the family. These differences account for some of the performance difference, since Mathews *et al.* included tRNAs but did not use tmRNAs.

We also looked carefully at the overlap between the 16 RNase P's in the Mathews benchmark and the 225 RNase P's in ours: 9 are unique to the Mathews set; 218 are unique to ours; and 7 occur in both sets. $mfold$ predicts the 9 structures unique to the Mathews set with 65% sensitivity and the 218 structures unique to our dataset at 56% sensitivity, so there may be some bias toward more difficult-to-predict structures in our set. In addition, there are differences between the secondary structures in our dataset and in Mathews' set even for exactly the same sequences. We compared the structures for the 7 RNase Ps included in both sets and found that these matched in only about 90% of their base pairs. $mfold$ predicted those structures annotated in the Mathews benchmark at 60% sensitivity, whereas it predicted those in ours (e.g. structures as annotated in Jim Brown's current RNase P database) at 56% sensitivity. The secondary structures of many bacterial and archaeal RNase P's were revised in 2001 in light of additional comparative sequence analysis [61]. This suggests (not unexpectedly) that there is a slight bias toward $mfold$ in benchmarks. RNA secondary structures are derived from a combination of manual comparative analysis and computational prediction, so there is some logical circularity when benchmarking (that is, we are benchmarking $mfold$ on structures that sometimes may have been produced in part by $mfold$). This bias disappears gradually as comparative evidence accumulates and structures are refined. We therefore feel we can adequately account for the absolute differences in prediction accuracy as measured by the two benchmarks.

The most relevant question is whether different prediction methods are ranked the same on either data set. This appears to be the case. For example, on the Mathews benchmark set measured by our procedures, the $G3$ grammar has a sensitivity of 41%, $G6$ has a sensitivity of 55%, and $mfold$ is at 67%, which is

essentially the same relative difference as shown in Table 3.3. We therefore believe that the relative performances reported in Table 3.3 are reasonably fair and largely independent of our choice of test structures.

3.5 Concluding Remarks

Our goal in this chapter was to explore how much SCFG design decisions impact RNA secondary structure prediction accuracy. Structural ambiguity is clearly a concern if structure prediction is the goal, as we showed with the reordering experiment. The results in Table 3.3 indicate that even among different unambiguous SCFG designs, grammar design decisions matter quite a bit.

Our results here probably underestimate the performance that could be wrung from any of these SCFGs if they were more systematically and carefully parameterized. Our parameterization was based just on counts from a large rRNA database. SSU and LSU ribosomal RNAs, though large, are not representative of all RNA structures; the ideal training set would be a large number of evolutionarily *unrelated* RNA secondary structures. Additional performance can probably be achieved from these SCFGs by training on larger, more diverse datasets.

Apparently without considering alternative designs, Knudsen and Hein had already described the simple and effective *G6* grammar, and extended it to analysis of input multiple sequence alignments in the program Pfold [100, 102]. Because our exploration has not been exhaustive, we can not say that *G6* is the best possible simple grammar. However, after exploring various alternative SCFG designs, we confirm that the Knudsen grammar is an excellent, simple framework in which to develop probabilistic RNA analysis methods.

Chapter 4

Pairwise RNA Structural Alignment

One of the first attempts to tackle structural alignment was by Gorodkin and Stormo [49]. Rather than computing the full structural alignment algorithm, they simplify the problem by eliminating bifurcation rules thereby limiting the structure to stems. This reduces the runtime to $O(L^4)$. A simple plus one scoring scheme is used for base pairs and a simple scoring matrix is used for alignment. They then applied this algorithm, FOLDALIGN, to the analysis of UTR regions with some success [50].

Despite being computationally demanding, computer advances and increasing scientific need have made the full pairwise Sankoff algorithm attractive. To implement structural algorithm requires developing a method of combining the score of folding, which is typically thermodynamic, with the score of alignment, which is largely probabilistic. The approach we have taken to reconciling the scoring system is to develop a fully probabilistic model, a pairSCFG. In this chapter we outline the details of the development of our pairSCFG.

4.1 Pairwise Grammar Design

From the single sequence grammars evaluated in Chapter 3, $G6$, $G7$, and $G8$ give reasonable prediction accuracy while staying relatively small. The $G6$ grammar looks particularly attractive because of its solid performance and exceedingly small size, requiring only three nonterminals.

However, $G6$ suffers from a fatal flaw when extending it to pairwise alignment. The $G6$ grammar emits all unpaired regions of the structure through the $L \rightarrow a$ rule. Hence each unpaired residue is emitted independently of its neighbors. Extending this to pairwise by replacement of the $L \rightarrow a$ rule by

$L \rightarrow \begin{matrix} x \\ y \end{matrix}$ would allow only ungapped alignments. On the other hand, using

$L \rightarrow \begin{matrix} x & | & x & | & - \\ y & & - & & y \end{matrix}$ precludes us from enforcing gap ordering, resulting in an

alignment ambiguous pairwise SCFG.

Consequently, we focused on extending the $G8$ grammar, the smaller of the two remaining candidates (4 nonterminals). It is instructive to walk through the derivation of a pairSCFG which we believe to be both alignment and structurally ambiguous. We start with the core of the structurally unambiguous $G8$ grammar, recall:

$$\begin{aligned} \text{G8: } S &\rightarrow aS \mid T \mid \epsilon \\ T &\rightarrow Ta \mid aPa' \mid TaPa' \\ P &\rightarrow aPa' \mid aNa' \\ N &\rightarrow aS \mid Ta \mid TaPa' \end{aligned}$$

which we want to combine with a fairly standard pairHMM, an example shown in Figure 2.2. The combination must preserve the unambiguous nature of both alignment and structure. We first extend base pairing states to the pairwise case,

$\begin{matrix} a & a' \\ P & \\ b & b' \end{matrix}$. We then replace single stranded regions with the full pairHMM,

eliminating extra non-emitting states whenever possible. The resulting grammar:

$$\begin{aligned}
S &\rightarrow \begin{matrix} a \\ b \end{matrix} S \mid \begin{matrix} a \\ - \end{matrix} L_x \mid \begin{matrix} - \\ b \end{matrix} L_y \mid T \mid \epsilon \\
T &\rightarrow T \begin{matrix} a \\ b \end{matrix} \mid R_x \begin{matrix} a \\ - \end{matrix} \mid R_y \begin{matrix} - \\ b \end{matrix} \mid \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \mid T \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \\
L_x &\rightarrow \begin{matrix} a \\ b \end{matrix} S \mid \begin{matrix} a \\ - \end{matrix} L_x \mid T \mid \epsilon \\
L_y &\rightarrow \begin{matrix} a \\ b \end{matrix} S \mid \begin{matrix} - \\ b \end{matrix} L_y \mid T \mid \epsilon \\
R_x &\rightarrow T \begin{matrix} a \\ b \end{matrix} \mid R_x \begin{matrix} a \\ - \end{matrix} \mid \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \mid T \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \\
R_y &\rightarrow T \begin{matrix} a \\ b \end{matrix} \mid R_y \begin{matrix} - \\ b \end{matrix} \mid \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \mid T \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \\
P &\rightarrow \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix} \mid N \\
N &\rightarrow \begin{matrix} a \\ b \end{matrix} S \mid \begin{matrix} a \\ - \end{matrix} L_x \mid \begin{matrix} - \\ b \end{matrix} L_y \mid T \begin{matrix} a \\ b \end{matrix} \mid R_x \begin{matrix} a \\ - \end{matrix} \mid R_y \begin{matrix} - \\ b \end{matrix} \mid T \begin{matrix} a \\ b \end{matrix} P \begin{matrix} a' \\ b' \end{matrix}
\end{aligned}$$

has eight nonterminals. An example parse tree is shown in Figure 4.1.

Lexicalization of the stacking state such that

$$P^{xx',yy'} \rightarrow \begin{matrix} a \\ b \end{matrix} P^{aa',bb'} \begin{matrix} a' \\ b' \end{matrix} \mid N$$

would require a 16x16x16x16 matrix to model all possible base pair combinations, resulting in 65,536 parameters for the stacking rule alone. This was considered prohibitive, particularly with limited training data available. Table 3.4 showed only a small loss in performance accuracy when single sequence grammars are not lexicalized, so we chose not to lexicalize the pairSCFG.

As with the single sequence grammars, there is flexibility in whether to model lone pairs. There is concern that enforcing a closing pair, which gave only a minuscule improvement in performance for single sequence, would be overly restrictive here. Therefore $P \rightarrow N$ is utilized and lone pairs are permitted.

4.2 Parameterization

If training the grammar for a particular application we might have reason to believe the rate for insertion in \mathbf{x} should differ from \mathbf{y} . However, when building a general tool, it seems disconcerting for the grammar to contain arbitrary directionality. It would be preferable if the model demonstrated symmetry, i.e. the comparison of \mathbf{x} to \mathbf{y} is equivalent to comparing \mathbf{y} to \mathbf{x} . Therefore we seek to parameterize the model so as to enforce symmetry.

The production rules of the grammar consist of three distinct types: single nucleotide rules where the right side of the production rule is

$$\begin{array}{c} a \\ - \\ L_x \end{array}, \begin{array}{c} - \\ b \\ L_y \end{array}, R_x \begin{array}{c} a \\ - \\ \end{array} \text{ or } R_y \begin{array}{c} - \\ b \\ \end{array},$$

aligned nucleotide rules where the right side of the production rule is

$$\begin{array}{c} a \\ b \\ S \text{ or } T \end{array} \begin{array}{c} a \\ b \\ \end{array},$$

and base paired nucleotide rules where the right side of the production rule includes

$$\begin{array}{c} a \\ b \\ P \end{array} \begin{array}{c} a' \\ b' \\ \end{array}$$

either alone or as part of a bifurcation rule. A fully parameterized version of this grammar contains 12 single nucleotide rules (4 parameters apiece), 8 aligned nucleotide rules (16 parameters apiece), and 8 aligned base pair nucleotide rules (256 parameters apiece) totaling 2234 emission parameters. We reduce the required emission parameters to 276 by tying together similar emissions so that the pairSCFG is described by 4 single nucleotide parameters, a single 4x4 matrix of alignment parameters, and a single 16x16 matrix of base pair parameters. To enforce scoring symmetry, the two matrices (4x4 alignment and 16x16 base pair) must be symmetric. The result is 150 emission parameters of which 129 are free.

Making the emission matrices symmetric is necessary but not sufficient to enforce scoring symmetry. We must also tie together transition parameters

appropriately. This is accomplished in two steps. The first step makes certain that there is no difference in starting a gap in either \mathbf{x} or \mathbf{y} . This is accomplished by tying together the three sets of “gap open” rules:

$$\begin{aligned}
 S &\rightarrow \begin{array}{c} a \\ - \end{array} L_x \mid \begin{array}{c} - \\ b \end{array} L_y \\
 T &\rightarrow R_x \begin{array}{c} a \\ - \end{array} \mid R_y \begin{array}{c} - \\ b \end{array} \\
 N &\rightarrow \begin{array}{c} a \\ - \end{array} L_x \mid \begin{array}{c} - \\ b \end{array} L_y \mid R_x \begin{array}{c} a \\ - \end{array} \mid R_y \begin{array}{c} - \\ b \end{array}
 \end{aligned}$$

from the same nonterminal. For example $S \rightarrow L_x$ must equal $S \rightarrow L_y$. The result is the S , T and N nonterminals are reduced to four transitions parameters each. The second step is to tie together similar productions from the L and R nonterminals associated with gap extensions and gap closure. The rules of the two L nonterminals:

$$\begin{aligned}
 L_x &\rightarrow \begin{array}{c} a \\ b \end{array} S \mid \begin{array}{c} a \\ - \end{array} L_x \mid T \mid \epsilon \\
 L_y &\rightarrow \begin{array}{c} a \\ b \end{array} S \mid \begin{array}{c} - \\ b \end{array} L_y \mid T \mid \epsilon
 \end{aligned}$$

are tied so that productions identical on the right side are equal. For example, the $L_x \rightarrow T$ rule has the same transition probability as $L_y \rightarrow T$. Similarly the two R nonterminals:

$$\begin{aligned}
 R_x &\rightarrow T \begin{array}{c} a \\ b \end{array} \mid R_x \begin{array}{c} a \\ - \end{array} \mid \begin{array}{c} a \\ b \end{array} P \begin{array}{c} a' \\ b' \end{array} \mid T \begin{array}{c} a \\ b \end{array} P \begin{array}{c} a' \\ b' \end{array} \\
 R_y &\rightarrow T \begin{array}{c} a \\ b \end{array} \mid R_y \begin{array}{c} - \\ b \end{array} \mid \begin{array}{c} a \\ b \end{array} P \begin{array}{c} a' \\ b' \end{array} \mid T \begin{array}{c} a \\ b \end{array} P \begin{array}{c} a' \\ b' \end{array}
 \end{aligned}$$

are described by four transitions rather than eight. The result is a reduction in transition parameters from 35 to 22 total parameters (16 free).

In the simplest parameterization of a pairSCFG, they are estimated from frequencies observed in annotated secondary structures. As utilized in Chapter 2, the training sets are the large and small subunit rRNAs, obtained from the European Ribosomal Database [181, 183]. Sequences containing more than 5%

ambiguous bases or with less than 40% base pairing are discarded. The resulting data set was then filtered to remove sequences with greater than 80% identity. For the pairSCFG we utilize the two resulting multiple sequence alignments which contain 707 sequences, of which 568 are SSU and 139 are LSU.

The parse tree for each pairwise alignment is determined and the number of occurrences of each production type is counted. All pairs, excluding self-to-self comparisons, are counted. Production probabilities are then estimated from the counts using a Laplace (plus-one) prior [30].

4.3 Benchmarks

Given the full pairwise structural alignment algorithm (CYK with our pairSCFG), we can examine performance relative to accuracy of RNA structure prediction, accuracy of alignment, memory requirements, and runtime. Our test set is a reproduction of the set described by Mathews for Dynalign [119]. We generate the multiple sequence alignments from Rfam seed alignments, one with 13 tRNA sequences and the other with 7 5S RNA sequences. This dataset, with all sequences shorter than 120 nucleotides, can be analyzed using the pairwise structural alignment.

As previously described, the parameters of the pairSCFG are tied to enforce symmetry. In other words, the comparison of sequence \mathbf{x} to \mathbf{y} is the same as comparing \mathbf{y} to \mathbf{x} . As a control, this was confirmed. Unless otherwise specified, we do not compare a sequence to itself. We need only compare each pair once, so in a multiple sequence alignment of n sequences we make $n(n - 1)/2$ pairwise comparisons.

As described in Chapter 3, we utilize base pair sensitivity and base pair positive predictive value (PPV) as structure metrics. We compare the structure predicted by the pairSCFG to the pairwise consensus structure given in the trusted alignment. We report sensitivity and PPV as cumulative statistics over all possible base pairs (total correct base pairs / total base pairs in all pairs).

In addition we calculate the alignment identity between the trusted pairwise structural alignment and our predicted alignment. The identity of an alignment is defined as the number of alignment columns correctly determined relative to the trusted (given) alignment. As with structure comparisons, we compute alignment identity cumulatively over all columns in all alignment pairs.

Compared to Single Sequence

We first want to show that the pairSCFG improves upon structure prediction relative to its single sequence analog. Both grammars are trained on the same ribosomal dataset described earlier. Using the single sequence SCFG upon which our pairSCFG is based (referred to in Chapter 3 as $G8^{nostack}$) we calculate the single sequence accuracy on each sequence in our dataset. Then we utilize the pairSCFG but compare each sequence to itself (therefore we do not gain any sequence alignment information). Lastly, we calculate the performance of all possible pairs, excluding self comparisons.

Table 4.1: **Comparing Performance of Single Sequence to Pairwise SCFG**
A comparison of the single sequence $G8^{nostack}$ grammar to the pairSCFG built from its backbone. Using the pairSCFG to compare a sequence to itself shows that the pairwise grammar performs roughly equivalent to its single sequence counterpart in the absence of alignment information.

Method	Full Set		5S		tRNA	
	Sens	PPV	Sens	PPV	Sens	PPV
$G8^{nostack}$	47.08	48.08	30.36	33.50	60.81	58.66
self vs self	48.49	42.65	33.48	31.51	60.81	50.76
pairwise	71.47	68.50	50.20	49.87	80.42	75.96

The results are shown in Table 4.1. In the absence of alignment information (self versus self), the pairSCFG performs similarly to the single sequence grammar on which it was developed. With the addition of alignment information, performance improves. For 5S, the worst case single sequence structure prediction (AE017179) improves from 7.14% sensitivity to 28.57 - 57.14 % depending on the pairwise comparison considered. For tRNA the worse single sequence case (RF6320) has 0% sensitivity, but with any pairwise comparison

improves to 33.33 - 100.00 %. In addition, of the 78 pairwise tRNA comparisons 11/78 result in 100% sensitivity and 9/78 result in 100% PPV (two are both).

Dynalign and *Stemloc* Comparisons

Dynalign is essentially a pairwise extension to the thermodynamic single sequence program RNAstructure. Dynalign computes a constrained pairwise structural alignment, restricting the alignment positions to a maximum distance of \mathcal{M} . Mathews reports single sequence performances on the tRNA and 5S subsets as 59.7% and 47.8% and pairwise performance of 86.1% and 86.4% respectively [119].

Using my reproduction of the Mathews dataset and our benchmarking method, we attempted to reproduce the Mathews benchmark as described in [119] using both Dynalign and *stemloc*. Neither of these implementations are scoring symmetric, giving different results depending on whether sequence \mathbf{x} is compared to \mathbf{y} or sequence \mathbf{y} is compared to \mathbf{x} . Therefore, we examine all pairs, excluding self-to-self comparisons.

In our hands, *mfold* obtains 65.2% and 30.8% and Dynalign obtains 84.8% and 58.1% sensitivity on tRNA and 5S respectively. As with the single sequence benchmark differences (see Section 3.4.3), the discrepancy between our benchmark and the one reported in [119] is attributable to slightly different metrics, different measurement techniques, and data set changes. These differences are most pronounced in the 5S dataset.

In the case of tRNA we both show approximately a 20% improvement in sensitivity when using pairwise methods over single sequence folding (thermodynamics improves from 65.2% to 84.8% and we improve from 60.8% to 80.4%). Our 5S performance is also comparable (thermodynamics improves from 30.8% to 58.1% and we improve from 30.4% to 50.2%). Interestingly, other grammars from Chapter 3 showed better single sequence performance on the 5S dataset (data not shown).

For our pairSCFG, a single set of parameters is utilized for both families. But for Dynalign, a pseudoenergy gap penalty ΔG_{gap}^0 was first optimized for each

family. The 5S penalty of $0.6 \text{ kcal/mol}^{-1}$ is quite different from the $2.0 \text{ kcal/mol}^{-1}$ utilized for tRNA. In addition, Dynalign permits single base pair inserts when predicting 5S but does not when predicting tRNA. Therefore it can be argued that the Dynalign performance is heavily optimized for this dataset.

An implementation of the Holmes envelope algorithm has only recently become available (<http://dart.sourceforge.net/stemloc/>). Holmes' implementation, *Stemloc*, is technically impressive. In addition to his previously described fold envelopes, the implementation utilizes alignment envelopes. Both sequences are pre-folded by a single sequence SCFG and pre-aligned by a pairwise hidden Markov model (pairHMM). The n best folds (per sequence) and n best alignments (per sequence pair) are sampled to generate the fold and alignment envelopes. The pairSCFG then only considers solutions which are consistent with the precomputed fold and alignment envelopes [77]. The Holmes pairSCFG is completely general and therefore both alignment and structurally ambiguous [80].

When used with default parameters, *stemloc* obtains 67.2% and 35.5% sensitivity on our tRNA and 5S datasets, respectively. (There is no single sequence equivalent for *stemloc*.) The default implementation of *stemloc* is constrained to minimize runtime. When these constraints are removed (-pleasekillme parameter) the tRNA performance improves to 84.7% sensitivity. However, in this mode there is insufficient memory on the machine to compute the 5S alignments, i.e. 5S alignments require $> 2.5 \text{ GB}$ memory.

Compute Resources

The full pairSCFG algorithm is computationally intensive. Using a dual 2.8 GHz P4 machine with 2.5 GB of memory we determine the empirical resource requirements. As shown in Figure 4.2, the observed runtime and memory requirements adhere to expected $O(L^4)$ and $O(L^6)$ in memory and time, respectively. For two representative tRNAs, lengths 76 and 77, the algorithm requires 290 MB of memory and 910 seconds. For two representative 5S sequences, lengths 116 and 117, the algorithm has already climbed to 1556 MB of memory and 22,902 seconds. While the resource requirements are daunting, they represents an unoptimized baseline implementation. Additional speedups and

memory reductions should be possible using software engineering techniques such as code profiling and vectorization. These methods have not yet been explored.

4.4 Concluding Remarks

The pairSCFG is postulated to be structurally and alignment unambiguous. To empirically confirm that the grammar is unambiguous would require the implementation of the full Inside and the development of a conditional Inside. A conditional Inside algorithm for a pairSCFG would calculate

$$P(\mathbf{x}, \mathbf{y}, \varpi | \mathcal{G}, \Theta) = \sum_{\pi \in \mathcal{C}(\varpi)} P(\mathbf{x}, \mathbf{y}, \pi | \mathcal{G}, \Theta)$$

where the Inside algorithm is limited to $\mathcal{C}(\varpi)$, those parse trees π consistent with the given structural alignment ϖ . If the structural alignment ϖ is a labeling that captures both base pairing (structure ν) and gap labeling (alignment ν), then structural and alignment ambiguity are simultaneously tested.

No amount of software engineering will solve the reality of $O(L^4)$ in memory and $O(L^6)$ in time. Consequently, if we are to utilize the structural alignment algorithm for larger datasets, it must first be constrained to run in less memory and with quicker runtimes. A constrained algorithm is the subject of the next chapter.

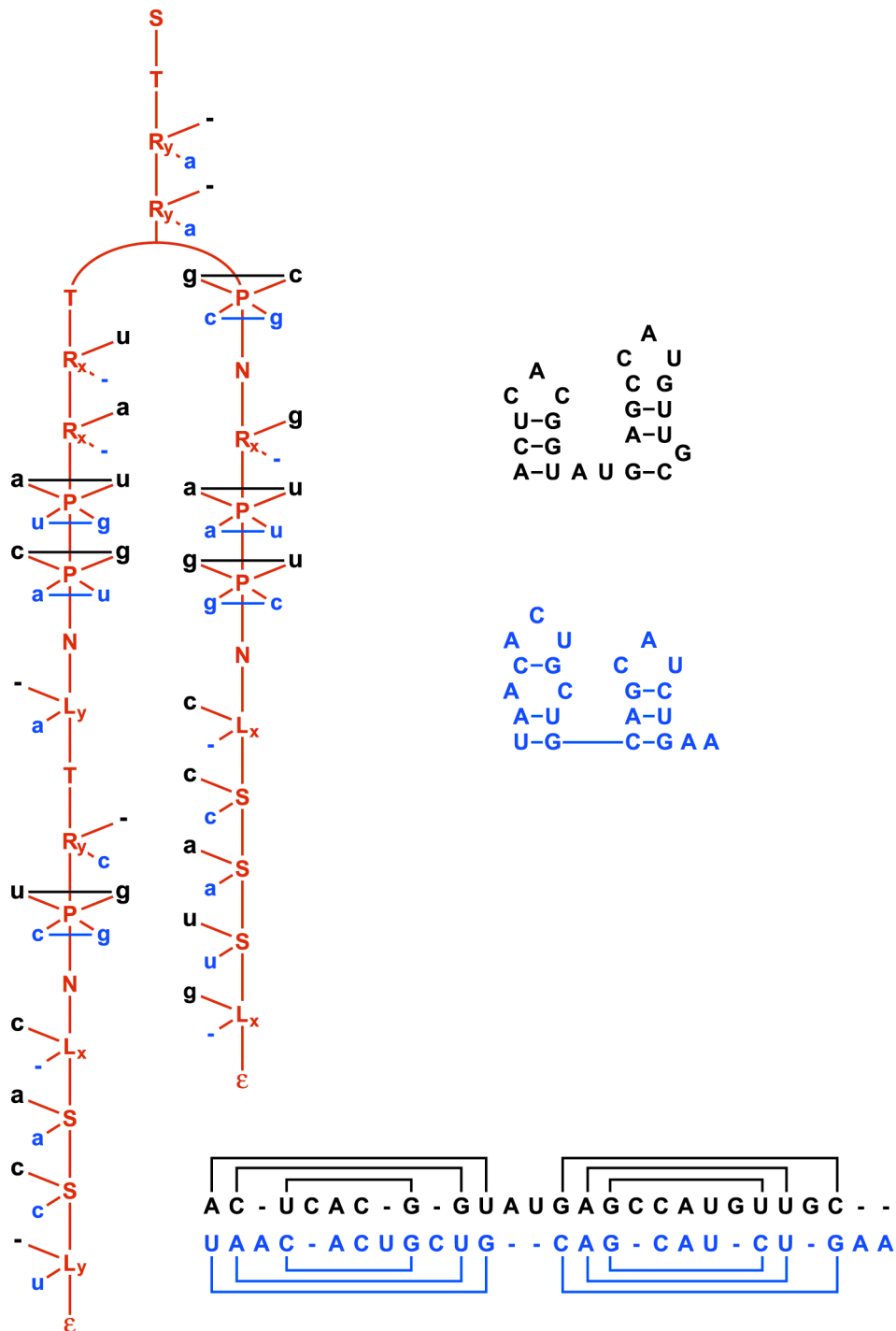


Figure 4.1: **Example parse tree for pairSCFG.** Left: An example parse tree for the pairSCFG described in the text. The grammar emits two correlated sequences, x in black (above) and y in blue (below). The resulting structural alignment is shown at the bottom, with lines connecting base pairs. Note that the structure for x (in black, above) has what is most likely an unconserved base pair C-G in second stem of the structure. The individual structures are shown on the right.

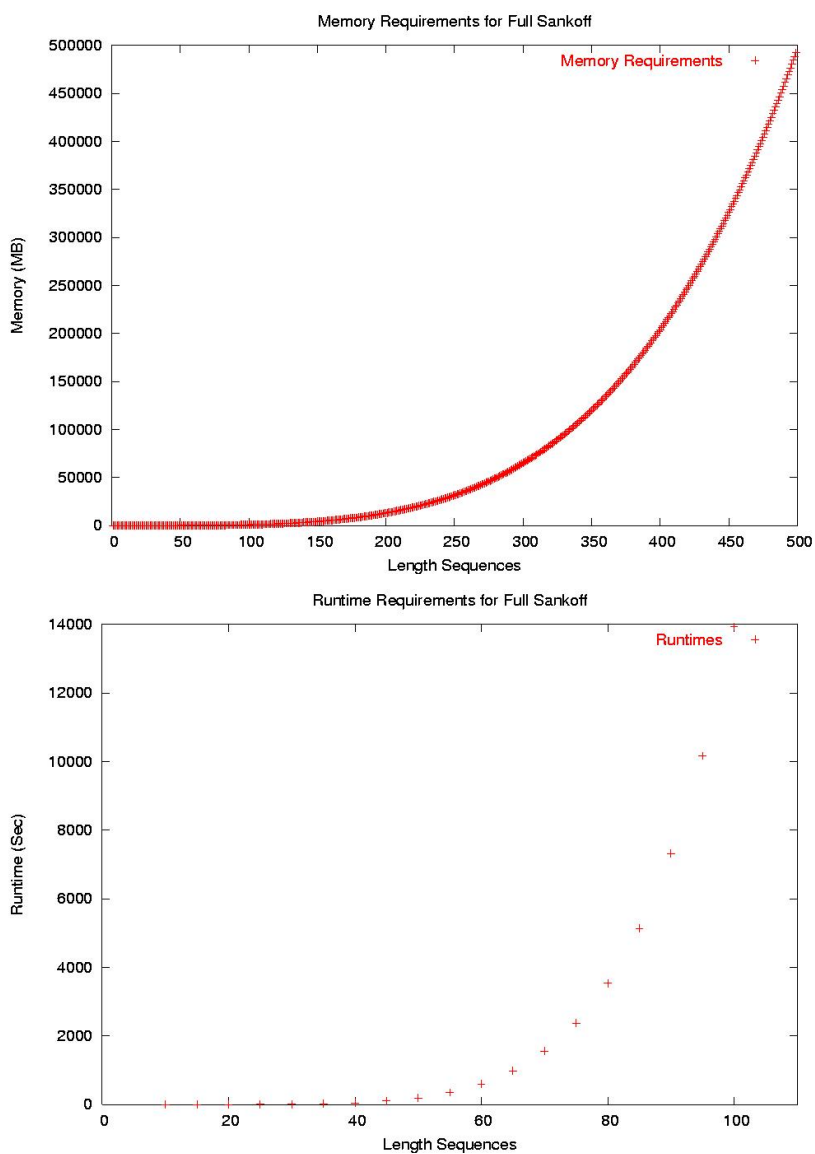


Figure 4.2: **Full pairwise structural alignment resource requirements.** The top graph shows the empirically determined memory requirements (Y axis in MB) for the pairwise alignment of two sequences of identical length (X axis in nucleotides). The bottom graph is a similar plot showing the empirically determined runtime (Y axis in seconds) for two sequences of identical length (X axis).

Chapter 5

A Constrained Structural Alignment Algorithm

Recall that the full pairwise structural alignment algorithm is $O(L^4)$ in memory and $O(L^6)$ in time. With our implementation on a machine with 2 GB of memory, this limits the structural alignment determination to sequences less than 120 nucleotides in length. Therefore, we can not employ the algorithm on the sequences of the dataset outlined in Chapter 2, which contains RNase P (average length 336 nucleotides), SRP (average length 202 nucleotides), and tmRNA (average length 353 nucleotides). In Rfam v6.1, 228 of 350 families (4930 of 7482 total sequences) are 120 nucleotides or less. It should be noted, however, that Rfam is biased toward small RNAs [51], subdividing larger RNAs into subdomains. To utilize pairwise structural alignment more broadly requires reducing the memory and runtime requirements.

5.1 General Methods of Constraint

Consider the problem of single sequence folding. When chemical modification, enzyme accessibility, or other experimental information [34] is available the folding algorithms can exclude structures which contradict observations [84, 117]. Numerous studies have shown that the utilization of

constraint information improves thermodynamic secondary structure predictions [115, 118, 191].

In alignment, a common constraint is the utilization of an exact short ungapped region, a word. The word anchors the sequences together, allowing for the construction of the alignment around these positions. More commonly, words are not known with absolute certainty. The expectation that words exist is the underlying principle to heuristic search algorithms such as BLAST and FASTA [4, 30, 136].

Two recent approaches have tackled the full structural alignment by utilizing constraints. Mathews and Turner [119] produced an implementation, Dynalign, which takes advantage of the global nature of the alignment to fix a window \mathcal{M} around the alignment diagonal. In other words, a position in sequence \mathbf{x} is restricted to align within $\pm\mathcal{M}$ of the same position in \mathbf{y} . Holmes and Rubin described an alternative approach. As described, their pairSCFG is restricted to a set of precomputed secondary structures, a “fold envelope” [80].

Compared with structure prediction, which is typically $O(L^3)$ in time, sequence alignment algorithms are a rapid $O(L^2)$. In addition, for suitably similar sequences, we anticipate that alignment will be more accurate than structure prediction. Consequently, in absence of experimental information, we favor alignment over structure constraints.

In this chapter we detail a third method of constraining the structural alignment algorithm. Our approach assumes a few positions of the structural alignment are known, which we refer to as “pins”. The structural alignment is then computed around these fixed positions. Because pins are not known *a priori*, we also detail a constraint generating system for this algorithm which utilizes the posterior probability table as generated from a pairHMM.

5.2 Constrain on Alignment Pins Algorithm

First we define notation. By convention, i , a , and j are indices in \mathbf{x} and k , b , and l are indices in \mathbf{y} . Sequence \mathbf{x} is length M and \mathbf{y} is length N . The

potential base pairing partner of i is j . The potential base pairing partner of k is l . The indices a and b locate the split points in a bifurcation rule. The subsequence $i\dots j$ aligns to $k\dots l$. For each nonterminal in the grammar we keep a four dimensional matrix, i, j, k, l . With bifurcation rules, we must consider all possible split points, therefore $1 \leq i < a < j \leq M$ and $1 \leq k < b < l \leq N$.

We define a pin q_z as a coordinate pair $(q_z(x), q_z(y))$ where the nucleotide $q_z(x)$ aligns to the nucleotide $q_z(y)$. The ordered set of alignment pins \mathcal{Q} contains Z pins, numbered from the 5' end of sequence \mathbf{x} , i.e. $z = 1..Z$. We always define two pins ($Z \geq 2$) as boundary conditions: one which comes before the 5' nucleotide of each sequence $q_1 = (0, 0)$ and one which follows the 3' nucleotide of each sequence $q_Z = (M + 1, N + 1)$.

We seek to calculate

$$\hat{\pi} = \operatorname{argmax}_{\pi} P(\pi, \mathbf{x}, \mathbf{y} \mid \mathcal{Q}, \mathcal{G}, \Theta)$$

the highest probability parse tree $\hat{\pi}$ for the sequences \mathbf{x} and \mathbf{y} given the set of alignment pins \mathcal{Q} , the pairwise grammar \mathcal{G} , and the parameters of the model Θ . If the pins are “correct”, i.e. $\forall z \in \mathcal{Q} : \{(q_z(x), q_z(y))\}$ are aligned nucleotides in the alignment from the unconstrained algorithm, then the constrained algorithm is guaranteed to find the same optimal structural alignment $\hat{\pi}$ as generated by the unconstrained algorithm.

Given \mathcal{Q} , we define a segment $\mathcal{S}(i)$ as the range of index k in \mathbf{y} which must be considered. A position i between pins q_z and q_{z+1} has a segment $\mathcal{S}(i)$ which implies $q_z(y) < k < q_{z+1}(y)$. We refer to edges of the range, in this case $q_z(y)$ and $q_{z+1}(y)$ as $\mathcal{S}_L(i)$ and $\mathcal{S}_R(i)$ respectively. Refer to Figure 5.1 *Panel A* for a labeled example. For the unconstrained algorithm, \mathcal{Q} contains two pins (q_1 and q_Z), $\mathcal{S}(i)$ is the full N nucleotides of \mathbf{y} , and the algorithm computes the full structural alignment algorithm.

Each constraint provided is an additional pin in \mathcal{Q} and will reduce the range of indices which must be considered. Figure 5.1 *Panel B* outlines in pseudocode the constrained structural alignment algorithm. The existence of alignment pins generally does not restrict the structure prediction (indices i, a , and j) of the first sequence, \mathbf{x} . However, for a particular instance of each of these

indices, the corresponding range of their possible alignment partners (k , b , and l respectively) is reduced. If we were using an ungapped alignment approach, a single midpoint pin would reduce the memory by 1/4 and the runtime by 1/8.

The effect of gaps deserves some discussion. When a position of interest i is a pin, the precise alignment partner is known. It seems naively that the segment $\mathcal{S}(i)$ should be one nucleotide. This is only true in ungapped alignments. In gapped alignments, we must consider that the pin may end or begin a gap. An indel aligns a nucleotide to *nothing*. In alignment dynamic programming the effect is that one index k progresses while the other i is fixed. Consequently, to consider all possible gap states requires $\mathcal{S}(i = q_z(x))$ to imply $q_{z-1}(y) < k < q_{z+1}(y)$. In this case the segment $\mathcal{S}(i)$ overlaps the segments $\mathcal{S}(i - 1)$ and $\mathcal{S}(i + 1)$, shown in Figure 5.1 *Panel C*. For this reason, a single midpoint pin takes more memory and longer time than the ungapped case.

Performance Given Correct Pin

In general, the runtime and memory requirements of the constrained algorithm are dominated by the largest segment. Consequently, the speedup and memory reduction implied by one or more pins depends on their distribution along the two sequences. Optimal placement results in evenly sized segments. All runtime and memory measurements were conducted on a dual 2.8 GHz P4 machine with 2.5 GB of memory.

Figure 5.2 shows a heatmap representation of the memory required when every possible single pin is utilized to constrain two random sequences of length 90 nucleotides. Runtime produces a similar heatmap. Clearly, pins which join the 5' end of one sequence to the 3' end of the other severely restricts the possible structural alignment and thus dramatically reduce the resources needed for the full pairSCFG algorithm. These are not, however, likely to be correct pins.

Because the alignment is global, most correct pins are expected to be near the diagonal. Figure 5.3 shows the memory reduction and time speedup for pins along the diagonal. Any pin eliminates a large number of possible alignments and therefore reduces the search space. The full unconstrained algorithm requires 560 MB memory and 3438 seconds to compute (for two 90mers). The pin which

provides the least improvement in resources connects the two sequences via their 5' (or 3') end and requires 536 MB and 1068 seconds. The midpoint pin (best case) requires 210 MB and 255 seconds.

Table 5.1: **Performance of pairSCFG using alignment pins.** A comparison of the performance of the full unconstrained structural alignment to the performance with pins. The first row recalls the performance of the full unconstrained structural alignment algorithm using the parameters trained on LSU and SSU as described in Chapter 4. The constrained algorithm utilizes well-spaced correct pins, taken from the known structural alignment.

Method	Full		5S		tRNA	
	Align	Sens (PPV)	Align	Sens (PPV)	Align	Sens (PPV)
base	83.79	71.47 (68.50)	81.36	50.20 (49.87)	84.50	80.42 (75.96)
Given Pins, Equally Spaced						
one	90.72	78.02 (74.02)	82.62	53.54 (53.80)	91.04	86.84 (82.66)
two	88.62	77.47 (75.39)	78.61	51.18 (52.76)	91.31	87.67 (83.49)
three	90.72	78.02 (74.02)	92.62	53.54 (53.80)	92.91	87.52 (81.26)

Assuming for the moment that the pins are correct, we can assess the performance of the constrained algorithm. We use the parameters as described in Chapter 4, estimated from the LSU and SSU training set using a Laplace (plus-one) prior. We then select correct pins from known structural alignments using the 5S and tRNA dataset described in Chapter 4.

Table 5.1 shows the performance of the constrained algorithm given a few correct pins. In general, performance is roughly equivalent or better than the unconstrained pairSCFG. When the performance improves, the correct pin actually eliminates high scoring incorrect alignments thereby improving the resulting predictions.

Table 5.2 shows the resource requirements of our constrained alignment algorithm, *stemloc*, and Dynalign. Dynalign runs in astonishingly little memory (only 37 MB for a pairwise comparison of 5S). This is the big benefit to a fixed alignment window parameter \mathcal{M} . Our implementation on the other hand requires significant memory for the same comparison (297 MB - 1.6 GB) even with pins. On the other hand, our method is reasonably fast, particularly when pins are utilized. *Stemloc* is resourceful in default mode, but is outperformed by both

Table 5.2: **Resource Utilization Comparison** In the Dynalign paper [119], Mathews gives the resource requirements for a single pair of tRNAs and 5S sequences. In section one, we reproduce those benchmarks on a dual 2.8 GHz P4 machine with 2.5 GB memory. In the second section we use the same benchmark to assess Holmes’ *stemloc*, using both default and -pleasekillme parameterization. (The machine lacked sufficient memory for 5S in -pleasekillme mode.) In the final section we compare these to the resource requirements of both the full and constrained implementation of our pairSCFG.

Sequence Pair	Parameters	CPU (sec)	RAM (MB)
Dynalign			
RD0260 vs RE6781 tRNAs	$\mathcal{M} = 15$	601	19
X02128 vs M16173 5S	$\mathcal{M} = 15$	1798	37
<i>Stemloc</i>			
RD0260 vs RE6781 tRNAs	default	1	17
X02128 vs M16173 5S	default	2	22
RD0260 vs RE6781 tRNAs	pleasekillme	1978	860
pairSCFG			
RD0260 vs RE6781 tRNAs	no pins	910	290
	one correct pin	88	112
	two correct pins	34	59
X02128 vs M16173 5S	no pins	22902	1556
	one correct pin	1708	579
	two correct pins	584	297

Dynalign and our pairSCFG in prediction accuracy (see Section 4.3). When constraints are removed from *stemloc* (-pleasekillme), resource demands soar.

5.3 A Pin Generating System

Unfortunately, we do not know a few alignment pins *a priori*. A method of generating high quality pins is need. We utilize a pairHMM to calculate a reliability measure, or posterior probability, for each position in a pairwise sequence alignment. The posterior probability is simply how often a position in one sequence aligns to a position in the other sequence relative to all possible alignments between the two sequences.

I. Holmes’ *dpswalign* program, a pairHMM implementation from his Dart package [76], is utilized with default parameters. The “-pt” option returns the posterior probability table between any two sequences. The “-oa” option returns the optimal accuracy alignment. An example posterior table is shown in Figure 5.4.

Given a posterior table, we want to select a set of *consistent* pins. A set of pins is consistent if all of the pins can co-exist in at least one sequence alignment. In a consistent set $q_z(x) < q_{z+1}(x)$ implies $q_z(y) < q_{z+1}(y)$. The optimal accuracy alignment [30, 76] maximizes the sum of the individual posterior probabilities implied by the positions of the alignment. We utilize the optimal accuracy alignment as a basis for selecting a set of high probability consistent pins.

The first concern was the accuracy of the posterior probabilities generated by *dpswalign*. Using the 5S and tRNA benchmarking dataset, we collect the posterior probability for all alignment columns of all optimal accuracy alignments. We then assess their correctness relative to the known structural alignment. A 90% posterior probability position should be correct 90% of the time. The result is shown in Table 5.3. As the data shows, the posterior probabilities are often optimistic (the percent correct is less than the posterior value). But at high posterior probabilities they are accurate: 96% of posterior values > 0.95 are correct and 97% of posterior values > 0.98 are correct.

We anticipate that the availability of pins whose posterior value is > 0.95 will depend on the percent identity between the sequences, so we determine how many quality pins (posterior > 0.95) are available for the benchmark dataset. Using the percent identity of the optimal accuracy alignment, Figure 5.5 shows the number of pins which meet our quality criteria (posterior > 0.95) for all pairwise comparisons in the benchmark dataset. For alignments of reasonable percent identity ($> 55\%$) there are plenty of high quality pins. No quality pins (posterior > 0.95) are observed for 19 of the 99 pairwise comparisons (all $< 52\%$ alignment identity). If we accept pins with posteriors as low as 0.90, only 8 of the 99 pairwise comparisons have no quality pins.

For the 80 pairwise comparisons with quality pins, we assess the impact of predicted pins on the structural alignment algorithm. We select a subset of

Table 5.3: **Accuracy of Posterior Probabilities.** The posterior probabilities from the optimal accuracy alignment are binned on 0.05 intervals and assessed for accuracy. For each bin (labeled by the smallest value of the interval) the number of pins (#Pins) observed in the interval, how many of these pins are incorrect (FP), and the resulting percent correct (% correct) are given. For example, the second row shows that 768 potential pins (columns in the optimal accuracy alignments) had posterior probabilities between 0.90 and 0.95 of which 101 were not correct (87% correct).

Bin	#Pins	FP	%correct
0.95	2909	114	96
0.90	768	101	87
0.85	445	74	83
0.80	380	124	67
0.75	400	148	63
0.70	353	141	60
0.65	346	155	55
0.60	316	160	49
0.55	394	195	51
0.50	374	184	51
0.45	420	247	41
0.40	316	206	35
0.35	307	187	39
0.30	259	179	31
0.25	184	125	32
0.20	89	63	29
0.15	27	25	7
< 0.15	3	1	67

quality pins (posterior > 0.95) in order of their posterior probabilities. If two proposed pins have the same posterior probability, we bias our selection toward the pin which produces the smallest segments. Table 5.4 shows that quality predicted pins do not reduce the performance relative to the full unconstrained algorithm.

Selecting pins purely on the posterior probability ignores the spatial distribution of the pins selected. Observations indicate that high posterior probabilities tend to cluster, but a neighboring pin is of minimal additional resource improvement (memory and runtime). For maximal resource

Table 5.4: **Performance of pairSCFG using predicted pins.** A comparison of the performance of the full unconstrained structural alignment to the performance with predicted pins. The benchmarking dataset is reduced from 99 pairwise comparisons to the 80 pairwise comparisons which have quality pins (posterior > 0.95). 5S reduces from 21 to 18 pairs and tRNA from 78 to 62 pairs. The performance of the unconstrained base model is given first (on this smaller dataset). Then performance is compared to using the best one or two predicted quality pins.

Method	Full		5S		tRNA	
	Align	Sens (PPV)	Align	Sens (PPV)	Align	Sens (PPV)
base	83.11	72.87 (70.87)	80.52	52.96 (53.15)	83.87	81.14 (77.91)
Predicted Pins, High Posteriors						
one	82.72	72.87 (70.31)	80.52	52.96 (53.15)	83.36	81.14 (77.06)
two	82.72	72.87 (70.31)	80.52	52.96 (53.15)	83.36	81.14 (77.06)

improvements, the pins must be well spaced. A method of pin selection which considers pin distribution is clearly needed.

5.4 Concluding Remarks

The structural alignment can be constrained on alignment, folding, or both. We describe an approach which utilizes alignment “pins”. Mathews used an alignment window method and Holmes described a fold envelope approach. These constraint types are shown in Figure 5.6. There is a natural hierarchy to constraint information, as an appropriately described envelope can be created to represent a window or pin. Note that a window need not be nearby, but instead can be augmented by an offset parameter, which may be particularly useful for fold windows. Alignment windows are conceptually similar to the banded approach to dynamic programming [15, 16]. Folding windows have been utilized in scanning algorithms [146] where the objective is to search through a large (genome sized) sequence looking for structural signals.

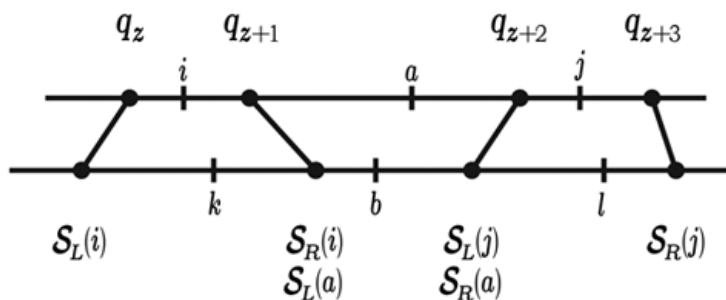
We should note that not all constraint information is useful for reducing the resource requirements of the algorithm. For example, knowing that a position is “paired” can improve the accuracy of the prediction but it is not immediately

clear how to utilize this information to reduce the resource requirements of structural alignment algorithm.

If the full alignment is known *a priori* the problem becomes one of identifying the shared structure. In this case the algorithm reduces to $O(L^2)$ in memory and $O(L^3)$ in time. Pfold [101,103] and QRNA [147] are two approaches to this problem. When the structure of one sequence is known *a priori*, the problem becomes one of finding a structural homolog. In general the algorithm then reduces to $O(L^3)$ in memory and $O(L^4)$ in time. Fundamentally this is the underlying approach of covariance models [33] and Klein's RSEARCH [97].

For sufficiently closely related sequences ($> 55\%$ identical in the optimal accuracy alignments) a few high quality (posterior > 0.95) pins are easily determined. At further distances, the pin method breaks down. Examination of the posterior probability table for those pairwise comparisons for which no quality pins could be found indicates that in these cases windows may be a better approach (data not shown). A reasonably small window would capture 0.95% of the alignment posterior probability mass. Rather than utilizing a fixed window, as Dynalign does, the window could be designed to vary as necessary to capture a majority of the alignment posterior probability mass. It is easy to imagine a *tour de force* implementation capable of utilizing all available constraint information in an optimal fashion.

A.



B.

```

for  $0 \leq i < LenX$ 
  for  $i \leq j < LenX$ 
    for  $S_L(i) \leq k \leq S_R(i)$ 
      for  $\max(k, S_L(j)) \leq l \leq S_R(j)$ 
        for  $i + 1 \leq a < j$ 
          for  $\max(k + 1, S_L(a)) \leq b \leq \min(l - 1, S_R(a))$ 

```

C.

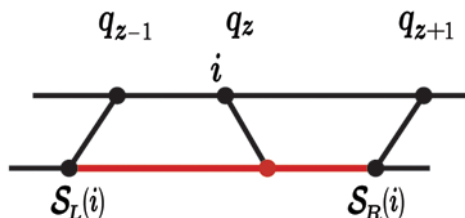


Figure 5.1: **Constrained Sankoff Algorithm.**

Panel A: A cartoon of a proposed problem with four pins, labeled q_z to q_{z+3} and shown as dumbbells which connect the two sequences. The indices $i, a,$ and j are positions in the first sequence \mathbf{x} ; $k, b,$ and l are positions in the second sequence \mathbf{y} . The corresponding segment edges \mathcal{S}_L and \mathcal{S}_R for each position in \mathbf{x} are labeled. In this notation, j is the potential base pairing partner of i , l is the potential base pairing partner of k , the subsequence $i\dots j$ aligns with $k\dots l$. The indices a and b are the required for identifying bifurcation points. *Panel B:* The constrained structural alignment algorithm in pseudocode, where the notation $\mathcal{S}_L(i)$ is defined as the left edge of the segment containing the position i and $\mathcal{S}_R(i)$ is the right edge of the segment containing i . The \max in the range for l is required to handle the case where i and j share the same segment. The b range must be handled similarly. *Panel C:* The special case where the position under consideration is equivalent to a pin. In this case, while we know the location of its alignment partner but must also consider the possibility of insertions in \mathbf{y} which may occur before or after this pin.

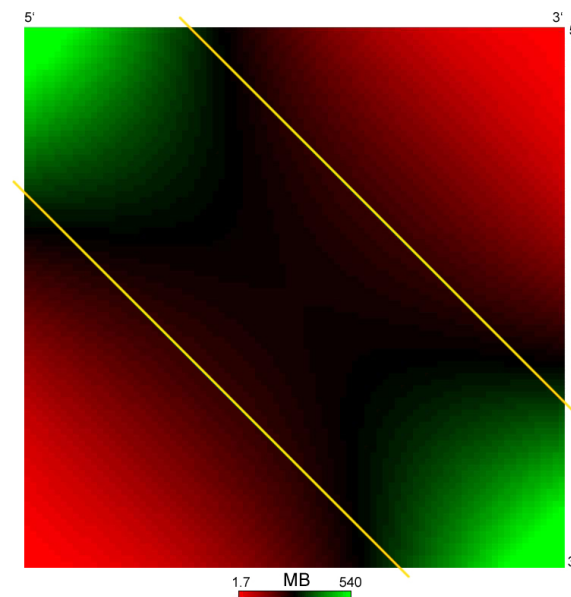


Figure 5.2: **Memory requirements for all possible pins between two 90 nucleotide sequences.** The heatmap shows that the lowest memory requirements (red regions) are in typically non-sensible alignments which join the 5' end of one sequence to the 3' end of the other. The band indicates the region which typically contains meaningful alignments. In this band the lowest memory requirements (black region) occur when the pins join the two sequences near their respective middles. When pins are near the edges (green region), the reduction in memory is significantly less.

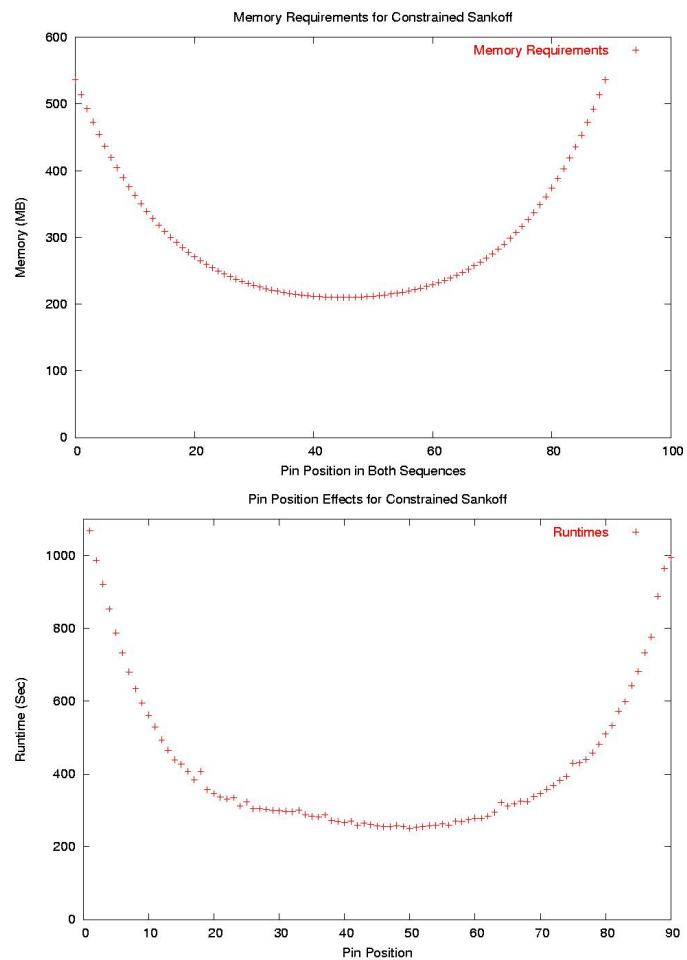


Figure 5.3: **Constrained pairwise structural alignment resource requirements.** The first graph shows the memory requirements (Y axis in MB) for the pairwise alignment of two sequences (length 90 nucleotides) for a single pin along the diagonal (X axis). The second graph shows the runtime of the same diagonal pins.

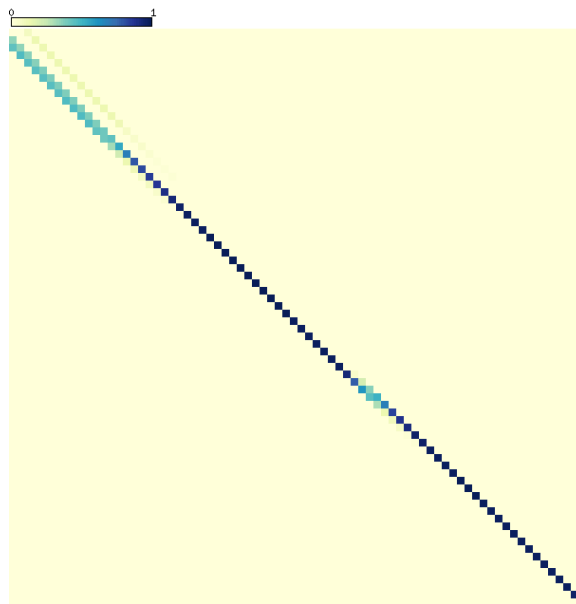


Figure 5.4: **Example posterior probability table.** The posterior probability table of tRNA RD0260 against RE6781 represented as a heatmap. High posteriors are shown in dark blue and low posteriors in yellow. Notice how the quality posteriors are around the diagonal, typically the region of meaningful alignments.

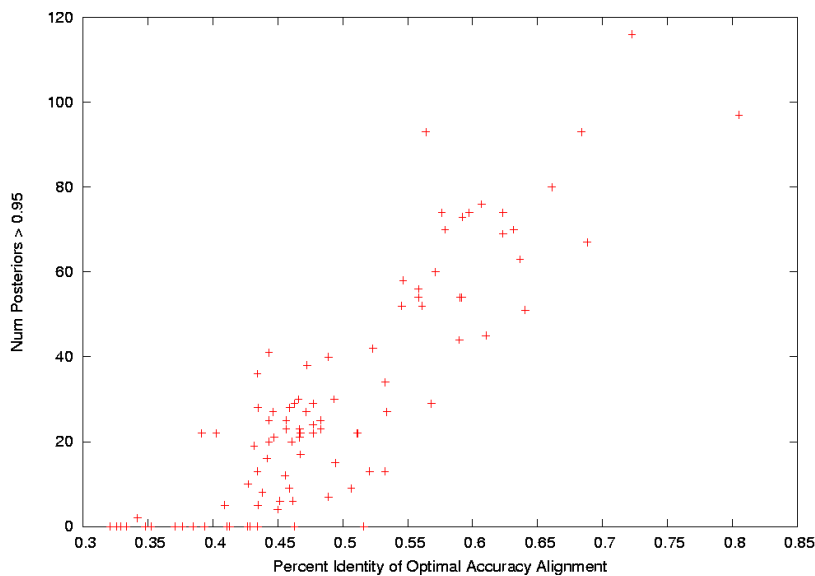
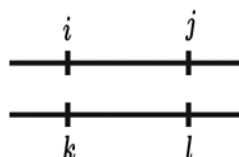


Figure 5.5: **Number of posteriors meeting our quality criteria.** The number of quality pins (posterior > 0.95) at various percent identities. The percent identities are calculated from the optimal accuracy alignment. Each point represents a single pairwise comparison.



Constraints	Align	Fold
Pins	(i,k)	(i,j)
Windows	\mathcal{M}	\mathcal{W}
Envelopes	$(i,\{k\})$	$(i,\{j\})$

Figure 5.6: **Methods of Constraining the Structural Alignment** Because structural alignment simultaneously considers both folding and alignment, there are many ways to constrain the algorithm. Top: In general we are looking for the alignment (or folding) partner for a position i in sequence \mathbf{x} . Given a generic coordinate system for structural alignment where i and j are positions in \mathbf{x} and k and l are positions in \mathbf{y} . Bottom: A “pin” designates the exact alignment (i, k) or folding (i, j) partner. A “window” looks for the partner within a specified radius (\mathcal{W} in folding or \mathcal{M} in alignment). An “envelope” extends the concept of a pin to a set of possible alignment k or folding j partners for a position, rather than a single exact partner.

Chapter 6

Future Work

In this work, probabilistic models are explored as a unifying framework for the scoring of alignment and folding. Small, simple, structurally unambiguous, single sequence stochastic context-free grammars were shown to be reasonable models for predicting RNA secondary structure. One of the solid single sequence grammars was then extended to the pairwise case of structural alignment. The resulting structural alignment algorithm shows good performance in predicting consensus secondary structure but is computationally expensive. Consequently, a constrained version of the algorithm was developed which utilizes known alignment positions (pins). The constrained algorithm calculates the structural alignment around these fixed positions, reducing both memory and runtime requirements.

Grammar Designs

The stochastic context-free grammar approach to single sequence folding deserves more attention. The performance of $G6$, $G7$, and $G8$ is encouraging, especially considering how few parameters are used by these grammars. The grammars in this work were tied so as to minimize parameters but preliminary work with $G3$ (not shown) indicates that alternative tying schemes may improve performance. Ultimately it may be possible to develop a SCFG design capable of outperforming the existing energy minimization methods.

The primary parameterization method utilized here was based on counts from SSU and LSU ribosomal RNA. Better performance can probably be achieved from these SCFGs by training on a larger more diverse dataset. One might also explore more sophisticated parameterization strategies. We have done some exploratory work using conditional maximum likelihood training [104] to find parameters that directly optimize the accuracy of structure predictions in the rRNA training data, rather than using maximum likelihood parameters, but did not obtain a significant performance increase. Another potentially promising strategy would be to “cross-train” the parameters, incorporating the thermodynamic parameters in some way as prior information to smooth some of the more poorly determined probabilistic parameters, analogous to how probabilistic data is increasingly being used to augment the energy rules [118].

Because the focus of this work was on the development of a pairSCFG, the grammars explored in Chapter 3 focused entirely on simple geometric loop length grammars. The thermodynamic methods, however, use more sophisticated loop length distributions. Grammars of this sort can be modeled by appropriately designed SCFGs. One implementation of an SCFG mirror of the Zuker algorithm has been described [146], but it used a structurally ambiguous grammar (it was not intended for secondary structure prediction per se; it was only used in summed Inside algorithm calculations where ambiguity doesn’t matter, not in a CYK algorithm where ambiguity does matter).

Preliminary work on loop length grammars was instructive. Three separate unambiguous loop length grammars (now shown) showed surprisingly disappointing performance (best was 35.7% sensitivity and 37.1% PPV on the single sequence benchmarking set of Chapter 3). Looking at the loop lengths distributions of known RNAs, a wide discrepancy was observed between the training set (SSU and LSU) and the testset (RNase P, SRP, and tmRNA). When the loop length distribution was artificially set to mimic the testset, performance approached the performance of the best grammars of Table 3.3. The implication is that careful parameterization of the loop length parameters is critical to the performance of these more complicated grammars.

Code Generator

The grammars compared in Chapter 3 are based on designs produced by myself and a number of colleagues (Graeme Mitchison, Ivo Hofacker, and Bjarne Knudsen) to whom I am grateful. These grammars are by no means exhaustive. It is not immediately obvious how to systematically search the space of possible simple unambiguous grammars.

There is, however, a way to speed the exploration of proposed grammars. The idea is based on the general idea of programming language development. A new programming language is described by both a specification and a parser/compiler that converts the language into machine executable code [1]. Subsequent software is written simply by adhering to the specification and relying on the compiler.

Ewan Birney's Dynamite [7] is a similar system. It is a code generating language for producing efficient source code for regular grammar dynamic programming. Initial work on extending Dynamite to include more extensive likelihood calculus, machine learning techniques, and more complicated grammar designs was described by Ian Holmes at the Bioinformatics Open Source Conference (BOSC) 2000. The result, Telegraph, is an ambitious and (to my knowledge) unimplemented specification.

The idea here is simpler than Telegraph. A specification could be written to describe a SCFG in a very systematic fashion. A parser would then read this language specification and produce training, CYK, Inside-Outside, and Posterior source code for the SCFG specified. Future SCFG designs would then need only be described to the code generator (compiler) to obtain a usable implementation.

Ideally, the code generator could handle pairSCFGs as well as the simpler single sequence SCFGs. In this work we assumed that we could utilize the relative performance of single sequence SCFGs as a proxy to their relative performance when extended to pairwise. A preliminary implementation of a pairwise $G7$ performed slightly under the pairwise $G8$ grammar of Chapter 4. This ordering confirms our assumption for at least these two grammars, but is only a single data point. Alternative grammars may show different results. Having a code generator would hasten the search of alternatives.

Evolutionary Models

The base model for our pairSCFG assumes a single degree of sequence divergence which may not adequately approximate the sequence divergence of a given pair. One alternative is to utilize a time dependent model of evolution. The simplest way to incorporate an evolutionary model into our pairSCFG is to replace the base pair emissions portion of the model parameters with a standard general reversible model of evolution [38, 100, 132]. Preliminary work is briefly described here.

Smith, Liu, and Tillier [157] develop a number of RNA rate matrices. They utilize the European Ribosomal Database [181, 183] to obtain sequence alignments. For each alignment they impose the structure of a single reference member which they obtain from the comparative RNA website database [13]. They produce four rate matrices based on different underlying datasets, Bacterial, Mitochondrial, Eukaryote SSU, and Eukaryote LSU. These matrices were obtained from <http://www.uhnres.utoronto.ca/tillier/rRNA/rna.html>.

Knudsen and Hein [101] develop a RNA rate matrix based on tRNAs from the Sprinzl [159] database and LSU rRNA from the De Rijk [144] database. The resulting rate matrix was obtained from <http://www.daimi.au.dk/~compbio/rnafold/>. This rate matrix is utilized in Pfold [103]. Subsequent work showed this matrix to be robust and reasonable even when applied to rapidly evolving 5'-region of HIV-1 [99].

We can utilize these published RNA base pair rate matrices as an alternative to our parameterization of conserved base pairs. The remainder of the parameters of the model are unchanged. This hybrid model produces complete parameter sets.

Each RNA rate matrix is composed of two components a 16x16 rate matrix \mathbf{R} and the corresponding stationary probabilities τ . Each rate matrix is scaled to an average rate of substitutions to one per unit time. Using the method described in Chapter 2, we then convert the rate matrices \mathbf{R} and stationary probabilities τ into the desired joint probabilities. We combine the joint probabilities (emissions) from the rate matrices with the parameters from the base model.

Table 6.1: **Performance of pairSCFG using different evolutionary models.** The first row (base) contains the performance results for parameters trained using LSU and SSU as described in Chapter 4. The second and third sections replace the base pair emission parameters of the model with rate matrices at $t = 1$. Smith [157] provided the Bacterial (bact), Eukaryote SSU (euk16), Eukaryote LSU (euk23), and Mitochondrial (mito) matrices. The 85-60 matrix is provided by E. Rivas (unpublished). Knudsen and Hein provide the Pfold matrix [101].

Method	Both		5S		tRNA	
	Align	Sens (PPV)	Align	Sens (PPV)	Align	Sens (PPV)
base	83.79	71.47 (68.50)	81.36	50.20 (49.87)	84.50	80.42 (75.96)
Evolutionary Models						
bact	86.63	77.67 (79.10)	79.88	53.94 (57.66)	88.45	86.87 (86.87)
euk16	83.16	74.04 (69.81)	78.73	54.25 (54.94)	84.35	81.72 (75.03)
euk23	84.10	75.41 (71.96)	79.53	53.54 (53.29)	85.34	83.88 (78.78)
mito	72.79	54.73 (51.96)	72.11	35.83 (34.11)	72.79	60.44 (62.06)
85-60	86.86	78.79 (80.24)	80.23	54.49 (58.54)	88.65	88.22 (88.06)
Pfold	85.89	76.86 (80.51)	77.65	52.28 (57.74)	88.11	86.39 (88.71)

Using the full pairSCFG we compare the performance of the base model, the four Smith matrices, the Pfold matrix, and a rate matrix version of RIBOSUM85-60 [96] obtained from Elena Rivas (unpublished). Table 6.1 shows the performance of the full pairSCFG using $t = 1$. The Smith Eukaryote SSU matrix shows comparable performance to our base model. As our training set is dominated by SSU sequences, this is perhaps not surprising. The Smith Mitochondrial matrix shows poor performance, but is based on less data than the other Smith matrices [157]. The remaining rate matrices show improved performance relative to our base model.

The way evolutionary models have been used here to parameterize the pairSCFG is primitive at best. As described, the evolutionary model is not leveraged, but rather used at a fixed t . Ultimately, the goal would be to correlate different values of t with the percent identity of the optimal accuracy alignment between two sequences. Parameters could then be reasonably adjusted according to the problem at hand.

As described, only a base pair evolutionary model is utilized. In addition, rate matrices could be used for aligned (4x4) nucleotides. An additional

extension would be to apply a time dependent model to gaps. At close evolutionary distances, indels are expected to be rarer and shorter than at further distances. An explicit model of indels, such as used in statistical alignment methods [64, 165] is complicated [78]. Simpler approaches include treating gaps as a special fifth character in the alphabet (Rivas, unpublished) or treating them as unknown nucleotides [103, 164].

Appendix A

Availability

The papers, source code, software documentation, and datasets utilized in this work are available at <http://selab.wustl.edu/people/robin/dissertation/>.

Training (parameter estimation), CYK parsing (structure prediction), and conditional Inside (ambiguity testing) code was written for each of the nine grammars and the stacking variants of Chapter 3. Inside with stochastic traceback (suboptimal sampling of parse trees) was implemented for the two ambiguous grammars. For the pairSCFG of Chapter 4, training (parameter estimation) and CYK parsing, both full and constrained, was also implemented. The software developed relies upon the SQUID library, developed by Sean Eddy. The ANSI C source code for these implementations are freely available under the GNU General Public License (GPL).

When possible, the external software packages utilized are also made available from the above URL. When redistribution is not permitted, a link is provided. Externally utilized packages include *mfold* v3.1.2, Pfold (Oct 2003), PKNOTS v 1.01, RNAstructure v4.0, the Vienna RNA package v1.4, Dynalign (Oct 2003), and the Dart package v1.0 (which includes *dpswalign* and *stemloc*). Datasets include rRNA training sets, the Rfam v5.0 ambiguity testing set, the single sequence benchmarking set, and the pairwise benchmarking set. Usage notes detailing how everything was utilized in this work are also provided.

References

- [1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers*. Addison Wesley, 1986.
- [2] V. R. Akmaev, S. T. Kelley, and G. D. Stormo. Phylogenetically enhanced statistical tools for RNA structure prediction. *Bioinformatics*, 16:501–512, 2000.
- [3] S. Altman and L. Kirsebom. Ribonuclease P. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 351–380. Cold Spring Harbor Laboratory Press, New York, 1999.
- [4] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.
- [5] J.F. Atkins, A. Böck, S. Matsufuji, and R.F. Gesteland. Dynamics of the genetic code. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World, Second Edition*, pages 637–673. Cold Spring Harbor Laboratory Press, Plainview, New York, 1999.
- [6] N. Ban, P. Nissen, J. Hansen, P.B. Moore, and T.A. Steitz. The complete atomic structure of the large ribosomal subunit at 2.4Å resolution. *Science*, 289:905–20, 2000.
- [7] E. Birney and R. Durbin. Dynamite: A flexible code generating language for dynamic programming methods used in sequence comparison. In *Proc. Int. Conf. on Intelligent Systems in Molecular Biology*, volume 5, pages 56–64. AAAI Press, 1997.
- [8] P. N. Borer, B. Dengler, , and O. C. Uhlenbeck. Stability of ribonucleic acid double-stranded helices. *J. Mol. Biol.*, 86:843–853, 1974.

- [9] D. Bouthinon and H. Soldano. A new method to predict the consensus secondary structure of a set of unaligned RNA sequences. *Bioinformatics*, 15:785–98, 1999.
- [10] C. M. Bowman, J. Sidikaro, and M. Nomura. Specific inactivation of ribosomes by colicin E3 in vitro and mechanism of immunity in colicinogenic cells. *Nat New Biol.*, 234:133–137, 1971.
- [11] J. W. Brown. The ribonuclease P database. *Nucleic Acids Res.*, 27:314, 1999.
- [12] P. Burgstaller, T. Hermann, C. Huber, E. Westhof, and M. Famulok. Isoalloxazine derivatives promote photocleavage of natural RNAs at G·U base pairs embedded within helices. *Nucleic Acids Res.*, 25(20):4018–4027, 1997.
- [13] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D’Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Muller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics.*, 3:2, 2002.
- [14] R.B. Cary and G.D. Stormo. Graph-theoretic approach to RNA modeling using comparative data. In C. Rawlings et al., editors, *Proc. Int. Conf. on Intelligent Systems in Molecular Biology*, pages 75–80, Menlo Park, CA, 1995. AAAI Press.
- [15] K.M. Chao, R.C. Hardison, and W. Miller. Constrained sequence alignment. *Bulletin of Mathematical Biology*, 55:503–524, 1993.
- [16] K.M. Chao, W.R. Pearson, and W. Miller. Aligning two sequences within a specified diagonal band. *Comput. Applic. Biosci.*, 8:481–487, 1992.
- [17] J.H. Chen, S.Y. Le, and J.V. Maizel. Prediction of common secondary structures of RNAs: a genetic algorithm approach. *Nucleic Acids Res.*, 28:991–9, 2000.
- [18] D.K. Chiu and T. Kolodziejczak. Inferring consensus structure from nucleic acid sequences. *Comput. Applic. Biosci.*, 7:347–52, 1991.

- [19] F. H. Crick. On protein synthesis. *Symp. Soc. Exp. Biol.*, 12:138–163, 1958.
- [20] F.H. Crick. Codon–anticodon pairing: the wobble hypothesis. *J. Mol. Biol.*, 19:548–55, 1966.
- [21] S.H. Damberger and R.R. Gutell. A comparative database of group I intron structures. *Nucleic Acids Res.*, 22(17):3608–3510, 1994.
- [22] R.W. Davies, R.B. Waring, J.A. Ray, T.A. Brown, and C. Scazzocchio. Making ends meet: A model for RNA splicing in fungal mitochondria. *Nature*, 300:719–724, 1982.
- [23] Y. Ding and C. E. Lawrence. A bayesian statistical algorithm for RNA secondary structure prediction. *Comput. Chem.*, 23:387–400, 1999.
- [24] Y. Ding and C. E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res.*, 31:7280–7301, 2003.
- [25] Y. Ding and C.E. Lawrence. Statistical prediction of single-stranded regions in RNA secondary structure and application to predicting effective antisense target sites and beyond. *Nucleic Acids Res.*, 29:1034–46, 2001.
- [26] R.M. Dirks and N.A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput Chem*, 24:1664–1677, 2003.
- [27] K. J. Doshi, J. J. Cannone, C. W. Cobough, and R. R. Gutell. Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics*, 5:105, 2004.
- [28] P. Doty, H. Boedtker, J. R. Fresco, R. Haselkorn, and M. Litt. Secondary structure in ribonucleic acids. *Proc. Natl. Acad. Sci. USA*, 45:482–99, 1959.
- [29] J. A. Doudna, C. Grosshans, A. Gooding, and C. E. Kundrot. Crystallization of ribozymes and small RNA motifs by a sparse matrix approach. *Proc. Natl. Acad. Sci. USA*, 90:7829–7833, 1993.

- [30] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK, 1998.
- [31] S.R. Eddy. Hidden Markov models. *Curr. Opin. Struct. Biol.*, 6:361–365, 1996.
- [32] S.R. Eddy. Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.*, 2:919–29, 2001.
- [33] S.R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Res.*, 22:2079–2088, 1994.
- [34] C. Ehresmann, F. Baudin, M. Mougel, P. Romby, J. P. Ebel, and B. Ehresmann. Probing the structure of RNAs in solution. *Nucleic Acids Res.*, 15:9109–9128, 1987.
- [35] V.A. Erdmann, M.Z. Barciszewska, A. Hochberg, Nathan de Groot, and J. Barciszewski. Regulatory RNAs. *Cell Mol. Life Sci.*, 58:960–77, 2001.
- [36] M. Fekete, I.L. Hofacker, and P.F. Stadler. Prediction of RNA base pairing probabilities on massively parallel computers. *J. Comput. Biol.*, 7:171–82, 2000.
- [37] B. Felden, H. Himeno, A. Muto, J.P. McCutcheon, J.F. Atkins, and R.F. Gesteland. Probing the structure of the *Escherichia coli* 10Sa RNA (tmRNA). *RNA*, 3:89–103, 1997.
- [38] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., 2004.
- [39] D.S. Fields and R.R. Gutell. An analysis of large rRNA sequences folded by a thermodynamic method. *Fold Des.*, 1:419–430, 1996.
- [40] C. Flamm, I.L. Hofacker, and P.F. Stadler. RNA in silico: The computational biology of RNA secondary structures. *Adv. Complex Syst.*, 2:65–90, 1999.
- [41] G.E. Fox and C.R. Woese. 5S RNA secondary structure. *Nature*, 256:505–507, 1975.

- [42] S.M. Freier, R. Kierzek, J.A. Jaeger, N. Sugimoto, M.H. Caruthers, T. Neilson, and D.H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *Proc. Natl. Acad. Sci. USA*, 83:9373–9377, 1986.
- [43] J. R. Fresco, B. M. Alberts, and P. Doty. Some molecular details of the secondary structure of ribonucleic acid. *Nature*, 188:98–101, 1960.
- [44] J.R. Fresco. From DNA melting profiles to tRNA crystals and RNA chaperones: how the secondary and tertiary structures of ribonucleic acids were discovered – a personal recollection. In Robert W. Simons and Marianne Grunberg-Manago, editors, *RNA Structure and Function*, pages 1–35. Cold Spring Harbor Laboratory Press, New York, 1998.
- [45] K. Gardiner and N.R. Pace. RNase P of *Bacillus subtilis* has a RNA component. *J. Biol. Chem.*, 255:7507–9, 1980.
- [46] R. Giegerich. A systematic approach to dynamic programming in bioinformatics. *Bioinformatics*, 16:665–77, 2000.
- [47] R. Giegerich, B. VoB, and M.Rehmsmeier. Abstract shapes of RNA. *Nucleic Acids Res.*, 32(15):1–9, 2004.
- [48] T. C. Gluick and D. E. Draper. Thermodynamics of folding a pseudoknotted mRNA fragment. *J. Mol. Biol.*, 241:246–262, 1994.
- [49] J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res.*, 25:3724–3732, 1997.
- [50] J. Gorodkin, S.L. Stricklin, and Gary D. Stormo. Discovering common stem-loop motifs in unaligned RNA sequences. *Nucleic Acids Res.*, 29(10):2135–2144, 2001.
- [51] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam: an RNA family database. *Nucleic Acids Res.*, 31:439–41, 2003.
- [52] C. Guerrier-Takada, K. Gardiner, T. Marsh, N. Pace, and S. Altman. The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, 35:849–857, 1983.

- [53] B. Gulko and D. Haussler. Using multiple alignments and phylogenetic trees to detect RNA secondary structure. In *Pac. Symp. Biocomput.*, pages 350–367, 1996.
- [54] A.P. Gulyaev, F.H.D. Van Batenburg, and C.W.A. Pleij. An approximation of loop free energy values of RNA H-pseudoknots. *RNA*, 5:609–617, 1999.
- [55] A.P. Gulyaev, F.H. van Batenburg, and C.W. Pleij. The computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol.*, 250:37–51, 1995.
- [56] R. R. Gutell, J. C. Lee, and J. J. Cannone. The accuracy of ribosomal RNA comparative structure models. *Curr. Opin. Struct. Biol.*, 12:301–310, 2002.
- [57] R.R. Gutell, N. Larsen, and C.R. Woese. Lessons from an evolving rRNA: 16S and 23S rRNA structures from a comparative perspective. *Microbiol. Rev.*, 58:10–26, 1994.
- [58] R.R. Gutell, A. Power, G.Z. Hertz, E.J. Putz, and G.D. Stormo. Identifying constraints on the higher-order structure of RNA: Continued development and application of comparative sequence analysis methods. *Nucleic Acids Res.*, 20:5785–5795, 1992.
- [59] R.R. Gutell, M.N. Schnare, and M.W. Gray. A compilation of large subunit (23S- and 23S-like) ribosomal RNA structures. *Nucleic Acids Res.*, 20:2095–109, 1992.
- [60] K. Han and H. Jin Kim. Prediction of common folding structures of homologous RNAs. *Nucleic Acids Res.*, 21:1251–1257, 1993.
- [61] J.K. Harris, E.S. Haas, D. Williams, D.N. Frank, and J.W. Brown. New insight into RNase P RNA structure from comparative analysis of the archaeal RNA. *RNA*, 7:220–32, 2001.
- [62] M.A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

- [63] M. Hasegawa, H. Kishino, and T. Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, 22:160–174, 1985.
- [64] J. Hein, C. Wiuf, B. Knudsen, M.B. Moller, and G. Wibling. Statistical alignment: computational properties, homology testing and goodness-of-fit. *J. Mol. Biol.*, 302:265–79, 2000.
- [65] H.A. Heus and A. Pardi. Structural features that give rise to the unusual stability of RNA hairpins containing GNRA loops. *Science*, 253:191–194, 1991.
- [66] H. Himeno, N. Nameki, T. Tadaki, M. Sato, K. Hanawa, M. Fukushima, M. Ishii, C. Ushida, and A. Muto. *Escherichia coli* tmRNA (10Sa RNA) in trans-translation. *Nucleic Acids Symp. Ser.*, 37:185–186, 1997.
- [67] M. B. Hoagland, M. L. Stephenson, J. F. Scott, L I. Hecht, and P. C. Zamecnik. A soluble ribonucleic acid intermediate in protein synthesis. *J. Biol. Chem.*, 231:241–57, 1958.
- [68] P.G. Hoel, S.C. Port, and C.J. Stone. *Introduction to Stochastic Processes*. Waveland Press Inc., 1972.
- [69] I.L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Res.*, 31(13):3429–3431, 2003.
- [70] I.L. Hofacker, S.H.F. Bernhart, and P.F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–2227, 2004.
- [71] I.L. Hofacker, M. Fekete, C. Flamm, M.A. Huynen, S. Rauscher, P.E. Stolorz, and P.F. Stadler. Automatic detection of conserved RNA structure elements in complete RNA virus genomes. *Nucleic Acids Res.*, 26:3825–36, 1998.
- [72] I.L. Hofacker, M. Fekete, and P.F. Stadler. Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.*, 319:1059–66., 2002.
- [73] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Chemical Monthly*, 125:167–88, 1994.

- [74] S.R. Holbrook. Crystallographic analysis of RNA structure. In Robert W. Simons and Marianne Grunberg-Manago, editors, *RNA Structure and Function*, pages 147–174. Cold Spring Harbor Laboratory Press, New York, 1998.
- [75] R. W. Holley, J. Apgar, G. A. Everett, J. T. Madison, M. Marquisse, S. H. Merrill, J. R. Penswick, and A. Zamir. Structure of a ribonucleic acid. *Science*, 147:1462–1465, 1965.
- [76] I. Holmes. *Studies in Probabilistic Sequence Alignment and Evolution*. PhD thesis, University of Cambridge, 1998.
- [77] I. Holmes. *Stemloc* documentation. 2004.
- [78] I. Holmes and W. J. Bruno. Evolutionary HMMs: a bayesian approach to multiple alignment. *Bioinformatics*, 17:803–820, 2001.
- [79] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *J. Comput. Biol.*, 5:493–504, 1998.
- [80] I. Holmes and G. M. Rubin. Pairwise RNA structure comparison with stochastic context-free grammars. In *Pac. Symp. Biocomput.*, pages 163–174, 2002.
- [81] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.
- [82] Y. Hu. Prediction of consensus structural motifs in a family of coregulated RNA sequences. *Nucleic Acids Res.*, 30(17):3886–3893, 2002.
- [83] J. A. Jaeger, D. H. Turner, and M. Zuker. Improved predictions of secondary structures for RNA. *Proc. Natl. Acad. Sci. USA*, 86:7706–7710, 1989.
- [84] J.A. Jaeger, J. SantaLucia, and I. Tinoco. Determination of RNA structure and thermodynamics. *Ann. Rev. Biochem.*, 62:255–287, 1993.

- [85] B.D. James, G.J. Olsen, J.S. Liu, and N.R. Pace. The secondary structure of ribonuclease P RNA, the catalytic element of a ribonucleoprotein enzyme. *Cell*, 52:19–26, 1988.
- [86] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.
- [87] Y. Ji, X. Xu, and G. D. Stormo. A graph theoretical approach to predict common RNA secondary structure motifs including pseudoknots in unaligned sequences. *Bioinformatics*, 20(10):1591–602, July 2004.
- [88] J. SantaLucia Jr. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA*, 95:1460–5, 1998.
- [89] V. Juan and C. Wilson. RNA secondary structure prediction based on free energy and phylogenetic analysis. *J. Mol. Biol.*, 289:935–947, 1999.
- [90] F. M. Jucker and A. Pardi. Solution structure of the CUUG hairpin loop: a novel RNA tetraloop motif. *Biochemistry.*, 34:14416–14427, 1995.
- [91] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, NY, 1969.
- [92] R.J. Keenan, D.M. Freymann, R.M. Stroud, and P. Walter. The signal recognition particle. *Annu. Rev. Biochem.*, 7:755–75, 2001.
- [93] J. Kim, J.R. Cole, and S. Pramanik. Alignment of possible secondary structures in multiple RNA sequences using simulated annealing. *Comput. Appl. Biosci.*, 12:259–67, 1996.
- [94] S.H. Kim, G.J. Quigley, F.L. Suddath, A. McPherson, D. Sneden, J.J. Kim, J. Weinzierl, and A. Rich. Three-dimensional structure of yeast phenylalanine transfer RNA: folding of the polynucleotide chain. *Science*, 179:285–8, 1973.
- [95] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120, 1980.

- [96] R.J. Klein. *Finding Noncoding RNA Genes in Genomic Sequences*. PhD thesis, Washington University School of Medicine, 2003.
- [97] R.J. Klein and S.R. Eddy. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44, 2003.
- [98] R. Knight, A. Birmingham, and M. Yarus. BayesFold: Rational 2° folds that combine thermodynamic, covariation, and chemical data for aligned RNA sequences. *RNA*, 10:1323–1336, 2004.
- [99] B. Knudsen, E.S. Andersen, C. Damgaard, J. Kjems, and J. Gorodkin. Evolutionary rate variation and RNA secondary structure prediction. *Comput. Biol. Chem.*, 28:219–26., 2004.
- [100] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15:446–454, 1999.
- [101] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15:446–54, 1999.
- [102] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res.*, 31:3423–3428, 2003.
- [103] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res.*, 31:3423–8., 2003.
- [104] A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. *Proc. Int. Conf. on Intelligent Systems in Molecular Biology*, 5:179–186, 1997.
- [105] R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Engineering.*, 7:1059–1068, 1994.
- [106] W. Li. *Molecular Evolution*. Sinauer Associates, Inc., 1996.
- [107] L.P. Lim, M.E. Glasner, S. Yekta, C.B. Burge, and D.P. Bartel. Vertebrate microRNA genes. *Science*, 299:1540, 2003.

- [108] R. Lück, S. Gräf, and G. Steger. ConStruct: a tool for thermodynamic controlled prediction of conserved secondary structure. *Nucleic Acids Res.*, 27:4208–4217, 1999.
- [109] R.B. Lyngso and C.N. Pedersen. RNA pseudoknot prediction in energy-based models. *J. Comput. Biol.*, 7:409–27, 2000.
- [110] R.B. Lyngso, M. Zuker, and C.N.S. Pedersen. Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, 15:440–5, 1999.
- [111] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [112] J.T. Madison, G.A. Everett, and H. Kung. Nucleotide sequence of a yeast tyrosine transfer RNA. *Science*, 153:531–4, 1966.
- [113] J.T. Madison, G.A. Everett, and H.K. Kung. On the nucleotide sequence of yeast tyrosine transfer RNA. *Cold Spring Harbor Symp. Quant. Biol.*, 3:409–16, 1966.
- [114] D. H. Mathews, A. R. Banerjee, D. D. Luan, T. H. Eickbush, and D. H. Turner. Secondary structure model of the RNA recognized by the reverse transcriptase from the R2 retrotransposable element. *RNA.*, 3:1–16, 1997.
- [115] D. H. Mathews, D. H. Disney, M. D. Childs, J. L. Schroeder, M. Zuker, and D. H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci. USA*, 101(19):7287–7292, 2004.
- [116] D.H. Mathews. Using an RNA secondary structure partition function to determine confidence in base pairs predicted by free energy minimization. *RNA*, 10:1178–1190, 2004.
- [117] D.H. Mathews, J.M. Diamond, and D.H. Turner. The application of thermodynamics to the modeling of RNA secondary structure. In Enrico Di Cera, editor, *Thermodynamics and Biology*, pages 177–201, 2000.

- [118] D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288:911–40, 1999.
- [119] D.H. Mathews and D.H. Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.*, 317:191–203, 2002.
- [120] J.S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–19, 1990.
- [121] F. Michel and M. Costa. Inferring RNA structure by phylogenetic and genetic analysis. In Robert W. Simons and Marianne Grunberg-Manago, editors, *RNA Structure and Function*, pages 175–202. Cold Spring Harbor Laboratory Press, Plainview, New York, 1998.
- [122] F. Michel, A. Jacquier, and B. Dujon. Comparison of fungal mitochondrial introns reveals extensive homologies in RNA secondary structure. *Biochimie*, 64:867–881, 1982.
- [123] F. Michel and E. Westhof. Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis. *J. Mol. Biol.*, 216:585–610, 1990.
- [124] V. Moulton, M. Zuker, M. Steel, R. Pointon, and D. Penny. Metrics on RNA secondary structures. *J. Comput. Biol.*, 7:277–92, 2000.
- [125] S.V. Muse. Evolutionary analyses of DNA sequences subject to constraints of secondary structure. *Genetics*, 139:1429–39, 1995.
- [126] P. Nissen, J. Hansen, N. Ban, P.B. Moore, and T.A. Steitz. The structural basis of ribosome activity in peptide bond synthesis. *Science*, 289:920–30, 2000.
- [127] H.F. Noller, J. Kop, V. Wheaton, J. Brosius, R.R. Gutell, A.M. Kopylov, F. Dohme, W. Herr, D.A. Stahl, R. Gupta, and C.R. Woese. Secondary structure model for 23S ribosomal RNA. *Nucleic Acids Res.*, 9:6167–6189, 1981.

- [128] R. Nussinov and A.B. Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc. Natl. Acad. Sci. USA*, 77:6903–13, 1980.
- [129] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithms for loop matchings. *SIAM J. Appl. Math.*, 35:68–82, 1978.
- [130] N. R. Pace, B. C. Thomas, and C. R. Woese. Probing RNA structure, function, and history by comparative analysis. In R. F. Gesteland, T. R. Cech, and J. F. Atkins, editors, *The RNA World*, pages 113–141. Cold Spring Harbor Laboratory Press, second edition, 1999.
- [131] N.R. Pace, D.K. Smith, G.J. Olsen, and B.D. James. Phylogenetic comparative analysis and the secondary structure of ribonuclease P RNA – a review. *Gene*, 82:65–75, 1989.
- [132] R.D.M. Page and E.C. Holmes. *Molecular Evolution: A Phylogenetic Approach*. Blackwell Publishers, 1998.
- [133] G. E. Palade. A small particulate component of the cytoplasm. *J. Biophys. Biochem. Cytol.*, 1:59–67, 1955.
- [134] J. Parsch, J.M. Braverman, and W. Stephan. Comparative sequence analysis and patterns of covariation in RNA secondary structure. *Genetics*, 154:909–921, 2000.
- [135] G. Pavesi, G. Mauri, M. Stefani, and G. Pesole. RNAProfile: an algorithm for finding conserved secondary structure motifs in unaligned RNA sequences. *Nucleic Acids Res.*, 32:3258–3269, 2004.
- [136] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.
- [137] O. Perriquet, H. Touzet, and M. Dauchet. Finding the common structure shared by two homologous RNAs. *Bioinformatics*, 19:108–16, 2003.
- [138] J.M. Pipas and J.E. McMahon. Method for predicting RNA secondary structure. *Proc. Natl. Acad. Sci. USA*, 72:2017–21, 1975.

- [139] E.V. Puglisi and J.D. Puglisi. Nuclear magnetic resonance spectroscopy of RNA. In Robert W. Simons and Marianne Grunberg-Manago, editors, *RNA Structure and Function*, pages 117–146. Cold Spring Harbor Laboratory Press, New York, 1998.
- [140] G. J. Quigley, L. Gehrke, D. A. Roth, and P. E. Auron. Computer-aided nucleic acid secondary structure modeling incorporating enzymatic digestion data. *Nucleic Acids Res.*, 12:347–366, 1984.
- [141] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.
- [142] U.L. RajBhandary, A. Stuart, R.D. Faulkner, S.H. Chang, and H.G. Khorana. Nucleotide sequence studies on yeast phenylalanine sRNA. *Cold Spring Harbor Symp. Quant. Biol.*, 3:425–34, 1966.
- [143] J. Reeder and R. Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5(104), 2004.
- [144] P. De Rijk, E. Robbrecht, S. de Hoog, A. Caers, Y. Van de Peer, and R. De Wachter. Database on the structure of large subunit ribosomal RNA. *Nucleic Acids Res.*, 27:174–178, 1999.
- [145] E. Rivas and Sean R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–68, 1999.
- [146] E. Rivas and S.R. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16:583–605, 2000.
- [147] E. Rivas and S.R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2:8, 2001.
- [148] D.P. Romero and E.H. Blackburn. A conserved secondary structure for telomerase RNA. *Cell*, 67:343–353, 1991.

- [149] M. A. Rosenblad, J. Gorodkin, B. Knudsen, and C. Zwieb and T. Samuelsson. SRPDB: signal recognition particle database. *Nucleic Acids Res.*, 31:363–364, 2003.
- [150] J. Ruan, G. D. Stormo, and W. Zhang. An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics.*, 20:58–66, 2004.
- [151] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, K. Sölander, R.C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Res.*, 22:5112–5120, 1994.
- [152] D. Sankoff. Simultaneous solution of the RNA folding, alignment, and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.
- [153] B. W. Senior and I. B. Holland. Effect of colicin E3 upon the 30S ribosomal subunit of *escherichia coli*. *Proc. Natl. Acad. Sci. USA*, 68:959–963, 1971.
- [154] M.J. Serra and D.H. Turner. Predicting thermodynamic properties of RNA. *Methods Enzymol.*, 25:242–61., 1995.
- [155] R.G. Shulman, C.W. Hilbers, Y.P. Wong, K.L. Wong, D.R. Lightfoot, B.R. Reid, and D.R. Kearns. Determination of secondary and tertiary structural features of transfer RNA molecules in solution by nuclear magnetic resonance. *Proc. Natl. Acad. Sci. USA*, 70:2042–5, 1973.
- [156] M. Sipser. *Introduction to Theory of Computation*. Brooks Cole Pub. Co., 1996.
- [157] A.D. Smith, T.W.H. Lui, and E.R.M. Tillier. Empirical models for substitution in ribosomal RNA. *Mol. Bio. Evol.*, 21(3):419–427, 2004.
- [158] M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch, and S. Steinberg. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res.*, 26:148–153, 1998.
- [159] M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch, and S. Steinberg. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res.*, 26:148–153, 1998.

- [160] B.C. Stark, R. Kole, E.J. Bowman, and S. Altman. Ribonuclease P: an enzyme with an essential RNA component. *Proc. Natl. Acad. Sci. USA*, 75:3717–21, 1978.
- [161] G. Storz. An expanding universe of noncoding RNAs. *Science*, 296:1260–3, 2002.
- [162] J.E. Tabaska, R.B. Cary, H.N. Gabow, and G.D. Stormo. An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, 14:691–9, 1998.
- [163] S. Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. In *Lectures on Mathematics in the Life Sciences*, volume 17, pages 57–86. American Mathematical Society, 1986.
- [164] J. L. Thorne, N. Goldman, and D. T. Jones. Combining protein evolution and secondary structure. *Mol. Bio. Evol.*, 13:666–673, 1996.
- [165] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *J. Mol. Evol.*, 33:114–124, 1991.
- [166] E.R. Tillier and R.A. Collins. High apparent rate of simultaneous compensatory base-pair substitutions in ribosomal RNA. *Genetics*, 148:1993–2002, 1998.
- [167] I. Tinoco, P. N. Borer, B. Dengler, M. D. Levin, O. C. Uhlenbeck, D. M. Crothers, and J. Bralla. Improved estimation of secondary structure in ribonucleic acids. *Nat. New Biol.*, 246:40–41, 1973.
- [168] I. Tinoco, O.C. Uhlenbeck, and M.D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:362–7, 1971.
- [169] H. Touzet and O. Perriquet. CARNAC: folding families of related RNAs. *Nucleic Acids Res.*, 32:W142–W145, 2004.
- [170] D.H. Turner, N. Sugimoto, J.A. Jaeger, C.E. Longfellow, S.M. Freier, and R. Kierzek. Improved parameters for prediction of RNA structure. *Cold Spring Harbor Symp. Quant. Biol.*, 52:123–133, 1987.

- [171] G. Varani and I. Tinoco Jr. RNA structure and NMR spectroscopy. *Q. Rev. Biophys.*, 24:479–532, 1991.
- [172] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Muller, D. H. Mathews, and M. Zuker. Coaxial stacking of helices enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl. Acad. Sci. USA*, 91:9218–9222, 1994.
- [173] P. Walter and G. Blobel. Signal recognition particle contains a 7S RNA essential for protein translocation across the endoplasmic reticulum. *Nature*, 299:691–8, 1982.
- [174] M. S. Waterman and T. F. Smith. RNA secondary structure: A complete mathematical analysis. *Math. Biosci.*, 42:257–266, 1978.
- [175] M.S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl. Acad. Sci. USA*, 80:3123–3124, 1983.
- [176] M.S. Waterman and T.H. Byers. A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. *Mathematical Biosciences*, 77:179–88, 1985.
- [177] A. L. Williams and I. Tinoco. A dynamic programming algorithm for finding alternative RNA secondary structures. *Nucleic Acids Res.*, 14:299–315, 1986.
- [178] K.P. Williams and D.P. Bartel. Phylogenetic analysis of tmRNA secondary structure. *RNA*, 2:1306–10, 1996.
- [179] C.R. Woese, L.J. Magrum, R. Gupta, R.B. Siegel, D.A. Stahl, J. Kop, N. Crawford, J. Brosius, R. Gutell, J.J. Hogan, and H.F. Noller. Secondary structure model for bacterial 16S ribosomal RNA: phylogenetic, enzymatic and chemical evidence. *Nucleic Acids Res.*, 8:2275–93, 1980.
- [180] S. Wuchty, W. Fontana, I.L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49:145–65, 1999.

- [181] J. Wuyts, Y. Van de Peer, T. Winkelmans, and R. De Wachter. The european database on small subunit ribosomal RNA. *Nucleic Acids Res.*, 30:183–5, 2002.
- [182] J. Wuyts, G. Perriere, and Y. Van De Peer. European ribosomal RNA database. *Nucleic Acids Res.*, 32 Database Issue:D101–3, January 2004.
- [183] J. Wuyts, P. De Rijk, Y. Van de Peer, T. Winkelmans, and R. De Wachter. The European large subunit ribosomal RNA database. *Nucleic Acids Res.*, 29:175–7, 2001.
- [184] J. R. Wyatt and J. D. Puglisi. RNA pseudoknots. stability and loop size requirements. *J. Mol. Biol.*, 214:455–470, 1990.
- [185] T. Xia, M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, and D. H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry.*, 37:14719–14735, 1998.
- [186] H.G. Zachau, D. Dutting, H. Feldmann, F. Melchers, and W. Karau. Serine specific transfer ribonucleic acids. XIV. comparison of nucleotide sequences and secondary structure models. *Cold Spring Harbor Symp. Quant. Biol.*, 3:417–24, 1966.
- [187] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.
- [188] M. Zuker. Suboptimal sequence alignment in molecular biology. Alignment with error analysis. *J. Mol. Biol.*, 221:403–20, 1991.
- [189] M. Zuker. Calculating nucleic acid secondary structure. *Curr Opin Struct Biol*, 10:303–10, 2000.
- [190] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, 31:3406–3415, 2003.
- [191] M. Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.

- [192] C. Zwieb. Structure and function of signal recognition particle RNA. *Prog. Nucl. Acid Res. Mol. Biol.*, 37:207–234, 1989.
- [193] C. Zwieb, J. Gorodkin, B. Knudsen, J. Burks, and J. Wower. tmRDB (tmRNA database). *Nucleic Acids Res.*, 31:446–447, 2003.

Vita

Robin D. Dowell

Date of Birth	June 26, 1974
Place of Birth	Durant, Oklahoma
Degrees	B.S. Computer Engineering, May 1997 B.S. Genetics, May 1997 M.S. Computer Science, May 2001 D.Sc. Biomedical Engineering, December 2004
Professional Societies	Association for Computing Machinery Institute of Electrical and Electronics Engineers International Society for Computational Biology
Publications	Dowell, R. D. S. R. Eddy (2004). Designing Stochastic Context-Free Grammars for RNA Secondary Structure Prediction. <i>BMC Bioinformatics</i> 5 (71). Dowell, R. D. R. M. Jokerst, A. Day, S. R. Eddy, L. Stein (2001). The Distributed Annotation System. <i>BMC Bioinformatics</i> 2 (7). Dowell, R. D. (2001). A Distributed Annotation System. <i>WUCS Technical Report wucs-01-07</i> .

December 2004