An Overview of Probabilistic Methods for RNA Secondary Structure Analysis

David W Richardson
Department of Computer Science & Engineering
University Of Washington
daverich@cs.washington.edu

Abstract

Biologists typically try to solve two important classes of RNA analysis problems: (1) predicting the secondary structure of a given RNA sequence, and (2) analyzing multiple alignments of families of related RNAs to create a structure profile useful for searching databases for homologous RNAs and for aligning RNA sequences to the profile.

In this paper, we provide a literature review of some of the most important papers on standard probabilistic methods for solving these two classes of problems.

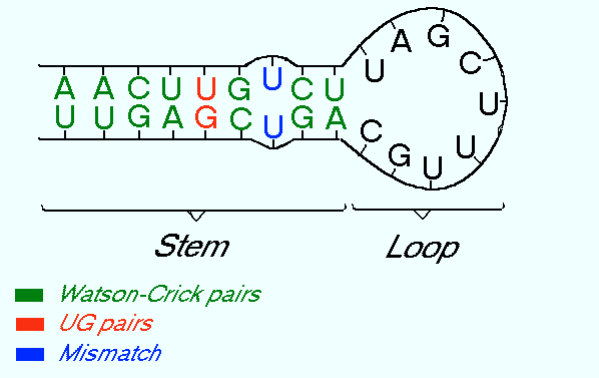

Figure 1: An example secondary structure of an RNA molecule [1]. Stem and loop regions are labeled, along with different types of base pairs.

1 Introduction

RNA is a polymer composed of four necleotides: adenine, cytosine, guanine, and uracil, abbreviated as A, C, G, and U. As in DNA, these nucleotides form complementary, hydrogen-bonded Watson-Crick base pairs between A-U and G-C, where uracil replaces thymine (T) from DNA. The G-C base pair is more stable than the A-U base pair because of an additional hydrogen bond. While there are 16 possible base-pairings, only 6 (A-U, G-U, G-C, U-A, U-G, C-G) are stable enough to form pairs. The rest of the unstable pairs are known as "mismatches."

Because RNA is a single-stranded molecule, it tends to fold back on itself by binding together stable base pairs. This resulting folded structure is called the *secondary structure* of the RNA. An example RNA sequence and its secondary structure is shown in figure 1. Standard terminology defines a *stem* as a region of contiguously stacked base pairs, and a *loop* as a region of unpaired bases bounded on both ends by paired bases.

It turns out that the secondary structure of RNA tends to be more important to RNA functionality than its primary sequence of nucleotides. In fact, base substitutions in the primary sequence can oftentimes be made as long as the secondary structure remains intact. As a result, analysis of RNA tends to be more complicated than DNA sequence analysis. There are two important classes of RNA analysis problems that biologists typically try to solve:

1. Predicting the secondary structure of a given RNA sequence.
2. Analyzing multiple alignments of families of related RNAs to create a structure profile useful for searching databases for homologous RNAs and for aligning RNA sequences to the profile.

In this paper, we provide a literature review of some of the most important papers on standard probabilistic methods for solving these two classes of problems. In particular, we consider the use of Covariance Models (CMs) [4] and Stochastic Context-Free Grammars (SCFGs) [12]. After reviewing the original papers in these areas, we look at an extension to SCFG-based methods that incorporates evolutionary history to increase the accuracy of the analysis [8]. This methodology has led to an important RNA analysis tool, Pfold [9]. We then discuss a number of papers that consider the analysis of noncoding RNAs [10] through what are called Pair-Stochastic Context-Free Grammars (Pair-

SCFGs) [11], [7]. Finally, we look at a paper that addresses how to design appropriate SCFGs [2], and we conclude by providing a very brief overview of common non-probabilistic methods for the interested reader.

Throughout the paper, we occasionally refer to the excellent explanations found in the standard book on the subject of biological sequence analysis [3]. This book is highly recommended for anyone interested in RNA sequence analysis.

2 RNA Analysis Using Stochastic Context-Free Grammars

In this section, we examine the original paper of Sakakibara et al. [12] that applied SCFGs to the problems of folding, aligning, and discriminating families of tRNA sequences. This paper does not address how to build profiles necessary for searching DNA-level databases for homologous RNAs. This is certainly possible with SCFG-based models, but we'll see in the next section a related formalism that better deals with this modeling issuse.

2.1 SCFGs

The paper begins by first providing definitions for SCFGs and related terminology. A *grammar*, or more accurately, a *context-free grammar* is composed of 3 distinct parts: (1) A finite alphabet Σ of *terminal* symbols. The terminal symbols for RNA sequences are simply those in the alphabet A, C, G, and U. (2) A set N of nonterminal symbols $S_1 ... S_N$ with a special start symbol S_0. (3) A set of *productions* P that define how to replace nonterminal symbols with a new subsequence. The language represented by the grammar is the set of all sequences of terminals that can be derived from the grammar by repeatedly applying production rules to expand nonterminals starting from the nonterminal S_0.

A *derivation* is a sequence of grammar production rules used to derive a given sequence of terminals from S_0. A *parse tree* visualizes a derivation as a syntactic tree-like structure, where the root node of the tree is S_0, and internal nodes correspond to intermediary sequences encountered in the derivation. Leaf nodes correspond to sequences of terminals found as subsequences in the derived sequence. A given sequence may have multiple valid parse trees. If some sequence has more than one parse tree, the grammar is considered to be *ambiguous*. The reader is referred to any standard introductory text on the theory of computation for more details.

A SCFG defines probability values for each production rule in the grammar. Therefore, a SCFG defines a probability distribution over all possible sequences that can be generated by the grammar.

SCFGs can represent RNA sequences having a common secondary structure by defining productions that can be classified into 4 types: $S \rightarrow aSa$ describes base pairs (for example, $S \rightarrow ASU$ defines an A-U base pair), $S \rightarrow aS$ and $S \rightarrow Sa$ describe unpaired bases, $S \rightarrow S$ describes deletions in the RNA sequence, and $S \rightarrow SS$ describes branched secondary structures. So for example if we have a SCFG that generates the sequence CAGCU according to the derivation $S \Rightarrow CS \Rightarrow CASU \Rightarrow CAGCU$, this derivation would correspond to a secondary structure whereby the A-U and G-C are paired to form a stem, and the first C is unpaired into an unstructured single strand. This derivation, and therefore the secondary structure of CAGCU are easily identified by looking at the sequence's parse tree.

As we've seen, a parse tree for a given RNA sequence defines a particular derivation and therefore corresponds to that sequence's predicted RNA secondary structure. This correspondence between parse trees and RNA secondary structure is key to SCFG-based analysis methods.

For the algorithms presented in this paper, the authors assume that the SCFGs are in a special restricted form, called *Chomsky normal form* (CNF). Grammars in this form only have production rules of the form $W_v \rightarrow W_y W_z$ or $W_v \rightarrow a$. It turns out that any grammar can be rewritten in CNF with appropriate modifications. While all of the algorithms in this paper rely upon the grammars being in CNF, there are variants of the algorithms that work on more general grammar forms. Some of these are discussed in [3].

2.2 Secondary Structure Prediction

The first problem the authors solve is how to use SCFGs to predict secondary structure. More formally, given an RNA primary sequence s and a SCFG G that models the RNA family of s, we want to predict the most likely secondary structure of s. Since parse trees of s represent secondary structures, this problem is reduced to finding the most likely parse tree for s.

Finding the most likely parse tree for s can be solved using the Cocke-Younger-Kasami (CYK) algorithm. Let's say that the sequence s has L symbols, indexed $x_1, ..., x_L$. Suppose the SCFG in CNF has M nonterminals $W = W_1, ..., W_M$ with start nonterminal W_1. We let v, y, and z be indices for nonterminals W_v, W_y, and W_z. The probability parameters of the SCFG are denoted as $t_v(y, z)$ to specify the transition probability of the production rule $W_v \rightarrow W_y W_z$, and as $e_v(a)$ to specify the emission probability of the production rule $W_v \rightarrow a$. The CYK algorithm uses dynamic programming to calculate the variable $\gamma(i, j, v)$ which equals the log probability of the most likely parse subtree rooted at the nonterminal W_v for the subsequence $x_i, ..., x_j$ of s for all i,i, and v. At the end of the calculation, $\gamma(1, L, 1)$

holds the log probability of the most likely parse tree of s. The initialization and recurrences for calculating $\gamma(i, j, v)$ are given below:

Initialize: for i=1 to L, v=1 to M:
$\gamma(i, j, v) = loge_v(x_i)$

Recurrence: for i=1 to L-1, j=i+1 to L, v=1 to M:
$\gamma(i, j, v) = max_{y,z} max_{k=i...j-1}$
$\{\gamma(i, k, y) + \gamma(k + 1, j, z) + logt_v(y, z)\}$

By incorporating a traceback variable into this procedure, a simply traceback algorithm can then be applied to recover the most likely parse tree of s [3].

With the CYK algorithm, once can also produce a multiple, structural alignment of RNA sequences. If we define a "match" nonterminal as a nonterminal symbol that derives either a letter or two base-paired letters in the consensus structure of the RNA family being modeled, then match nonterminals correspond to columns in the multiple alignment where the match nonterminal's letters appear in the consensus structure. For each sequence in the alignment, simply calculate that sequence's most likely parse tree. Then print the sequence in such a way that letters derived from match nonterminals are placed in their appropriate columns in the alignment.

2.3 Likelihood of a Sequence

To calculate the likelihood or probability that a given sequence s is generated by the model SCFG, one must sum up all of the probabilities of all possible parse trees of s that can be generated by the grammar. The algorithm for doing this is known as the Inside algorithm. It is very similar to the CYK algorithm, but replaces the max terms with sum terms in the recurrence to calculate variables $\alpha(i, j, v)$ equal to the probability of a parse subtree rooted at nonterminal W_v for a subsequence $x_i, ...x_j$ of s for all i, j, and v.

The authors show that by taking the -log of this probability, they get the *negative log likelihood* (NLL) score of s being generated by the grammar. This score is normalized to produce a Z-score that is suitable for thresholding to determining whether or not a given sequence belongs to the family of RNA sequences represented by the SCFG model.

The authors fail to mention that the Inside algorithm (or for that matter the CYK algorithm) can be used to scan over large genomes to score all subsequences of length $\leq w$. The dynamic programming algorithm calculates the Z-score for each subsequence (or alternatively the log-odds score) to determine whether or not the sequence is a match to the SCFG model. Therefore, both the Inside algorithm and the CYK algorithm can be used as a basis for searching for homologous RNAs in a genome database. A nice example of such a scanning algorithm is discussed in [10].

2.4 SCFG Parameter Estimation

Finally, the authors provide an expectation-maximization type algorithm known as the Inside-Outside algorithm for estimating the probability parameters of a SCFG from a set of training sequences. This algorithm makes use of an algorithm called the Outside algorithm that calculates variables $\beta(i, j, v)$ equal to the summed probability of all parse trees excluding subtrees rooted at nonterminal W_v that generates subsequence $x_i, ...x_j$. The outside algorithm uses the $\alpha(ij, v)$ values calculated by the Inside algorithm [3].

After calculating $\alpha(i, j, v)$ and $\beta(i, j, v)$ values, one can use EM to re-estimate the parameters of the SCFG. This procedure essentially calculates the expected number of times $c(v)$ that a given nonterminal W_v is used in a derivation, and the expected number of times $c(v \rightarrow yz)$ that the production rule $W_v \rightarrow W_y W_z$ is used from nonterminal W_v in a derivation. The value $\frac{c(v)}{c(v \rightarrow yz)}$ is the re-estimation of the transition probability $t_v(y, z)$. A similar procedure can be applied to re-estimate probabilities for the $W_v \rightarrow a$ production rules.

The authors note that the standard Inside-Outside algorithm takes time cubic in the length of each training sequence. As a result, they develop a faster, modified version of the Inside-Outside algorithm, known as the Trree-Grammar EM training algorithm. This algorithm however requires folded RNA sequences (not just the RNA sequence) as training examples.

2.5 Results

The authors apply their new EM-style training algorithm to a database of 1477 tRNA sequences to produce four grammars. They then used these grammars to perform 3 tasks: discriminate tRNAs from non-tRNAs, produce multiple sequence alignments, and predict secondary structures of new RNA sequences.

The generated multiple alignments for each of the four grammars were compared to a known alignment called EMBLtRNA. The results indicate that for the best trained grammar, upwards of 99% of base pairs specified by the EMBLtRNA alignment were also matched in the alignment made by the grammar. However, one particular class of tRNAs showed only 83% matching accuracy. The authors note that the majority of such non-matching RNAs were parasitic worm and mammalian mitochondrial tRNA that lack a specific domain known as the D-domain. This group of tRNAs was referred to as Part III.

To assess the accuracy of discriminating tRNAs from non-tRNAs, the authors compute Z-scores for all tested

tRNA sequences and non-tRNA sequences using the normalized NLL score mentioned previously. A Z-score threshold of 5 was set to discriminate tRNAs from non-tRNAs. Once again, very good discrimination was achieved except for the Part III sequence group.

2.6 Discussion

This paper showed that SCFG-based techniques for the analysis of RNA can be effective. The authors note though that their method is unable to create a good enough model necessary for searching databases at the DNA-level to find homologous RNAs. We mentioned that such a scanning algorithm could be constructed using either the Inside algorithm or the CYK algorithm as a basis for scoring subsequences. However, the grammars should ideally be modified to allow for introns and insertions and deletions of bases. Allowing for insertions and deletions turns out to be very important for modeling any real RNA families.

The next paper we consider uses a related model known as a Covariance Model (CM) that is a closely related formalism to SCFGs. However, CMs allow for insertions and deletions of bases, thus making them better suited for use in scanning algorithms.

3 RNA Analysis Using Covariance Models

In this section, we look at a paper by Sean Eddy and Richard Durbin [4] that develops probabilistic models known as *covariance models* (CM) for performing RNA consensus secondary structure prediction, multiple sequence alignment, and database similarity searching.

This paper surfaced at nearly the same time as the paper on SCFGs we previously discussed. The model and techniques are essentially a different formalism for SCFGs-based analysis. However, unlike the SCFGs we saw in [12], the covariance models in [4] can account for insertions and deletions of bases necessary for providing a more accurate model of an RNA family suitable for performing databases searches for homologous RNAs.

3.1 Covariance Models

The basic idea behind a covariance model is straightforward. A CM is based on a binary tree that describes both the primary sequence and secondary structure of a single RNA. Suppose we are given an RNA sequence and its secondary structure. We can model this sequence and its structure by using a tree-like state machine. There are nodes that correspond to emitting unparied nucleotides, and nodes for emitting paired nucleotides. Special "bifurcation" nodes represent junctions between multiple helices in the secondary structure. Given such a tree, you simply walk the tree from root to leaves and left to right to get the RNA's primary sequence. Paired nucleotides

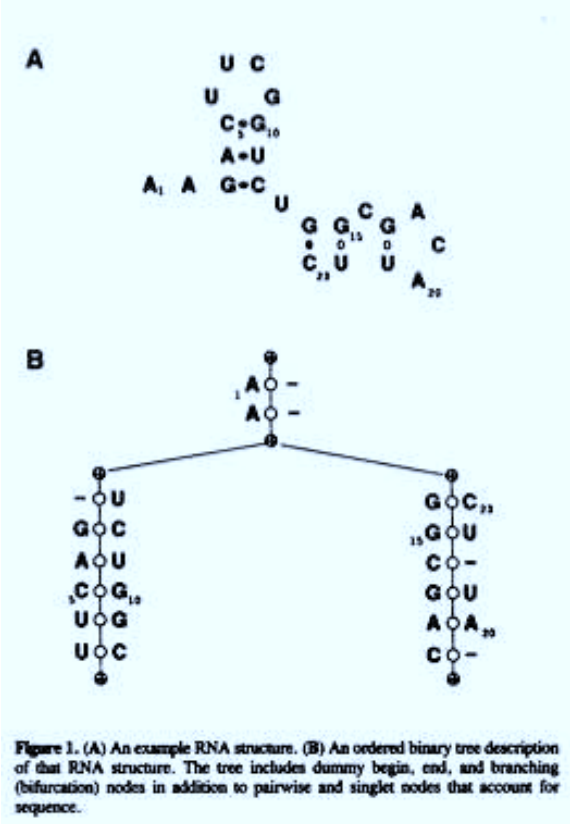

Figure 1. (A) An example RNA structure. (B) An ordered binary tree description of that RNA structure. The tree includes dummy begin, end, and branching (bifurcation) nodes in addition to pairwise and singlet nodes that account for sequence.

Figure 2: An example RNA structure along with its ordered binary tree description from the original paper.

in the secondary structure correspond to pairs emitted by the corresponding node in the tree. This idea is analogous to emitting nested pairs in a context-free grammar. In fact, these trees are nothing more than parse trees for the derivation of an RNA sequence from a SCFG. An example of such a binary tree is shown in figure 2.

However, as was the case with parse trees for SCFGs, such binary trees can only describe a fixed RNA structure. In general we want to model a family of RNA sequences and their structure; that is, we want a profile of an RNA family. A given family of RNAs will exhibit a consensus secondary structure. However, individual homologous RNAs will likely differ from the consensus profile in different ways. To model these variations, we need to allow for the possibility of insertions, deletions, and mismatched base-pairs with respect to the consensus. If we view a binary tree as modeling the consensus structure of the RNA family, then these trees are like "guide trees" where nodes correspond to columns in a trusted multiple alignment.

To model variations in emitted bases, nodes in the guide tree can be modified to emit bases and base-pairs

probabilistically. To accommodate variations in structure relative to the consensus, nodes are replaced with a number of *states* that correspond to different structures that can be emitted by the node. For example, a node might be replaced by a "match" state, an "insert" state, and a "delete" state. A match state corresponds to the probabilistic emission of a matched base-pair, where an insert state corresponds to the probabilistic emission of an unmatched (inserted) base.

States are connected to each other via *transition probabilities*. Therefore, each node is essentially expanded into a small state machine which can make transitions within itself or to a child node, emitting bases and base-pairs probabilistically. These expanded-node state machines allow for changes in secondary structure relative to the consensus structure specified by the initial guide tree. The resulting probabilistic model is known as a *covariance model*.

We note that there are many node types possible in a CM, each of which can expand into many different states with varying transitions. Some of these node types along with their accompanying states are listed below:

1. *left singlet node* - this node corresponds to the top 2 "A o -" nodes in the binary tree of figure 2. This node has 3 possible states with emission probabilities on letters: MATL (match left), INSL (insert left), and DEL (delete). These states correspond to the various ways that a single letter could be emitted with no "right" match from this node. This is why it is called a left singlet node.

2. *right singlet node* - this is the dual of a left singlet node.

3. *bifurcation node* - this node corresponds to a junction between multiple helices in the secondary structure. The node has 2 children nodes, each of which are transitioned to with probability 1.

4. *pairwise node* - this node has a special "match" state which emits 16 possible base-pairs, each with some probability. This corresponds to a base pairing in the RNA secondary structure. There are also MATL and MATR states corresponding to when part of the base-pair match is missing. Other states such as INSL and DEL are also possible in this node.

5. *begin* and *end* nodes - these are special nodes that mark the beginning and end of a helix.

An example of a CM showing these nodes and states is shown in figure 3. This figure is also directly lifted from the original paper.

To use these models, we have to set up a specific model structure for an RNA family and then assign probabilities to the model's parameters. Doing so defines a probability distribution on primary sequences and secondary structures modeled by the CM.

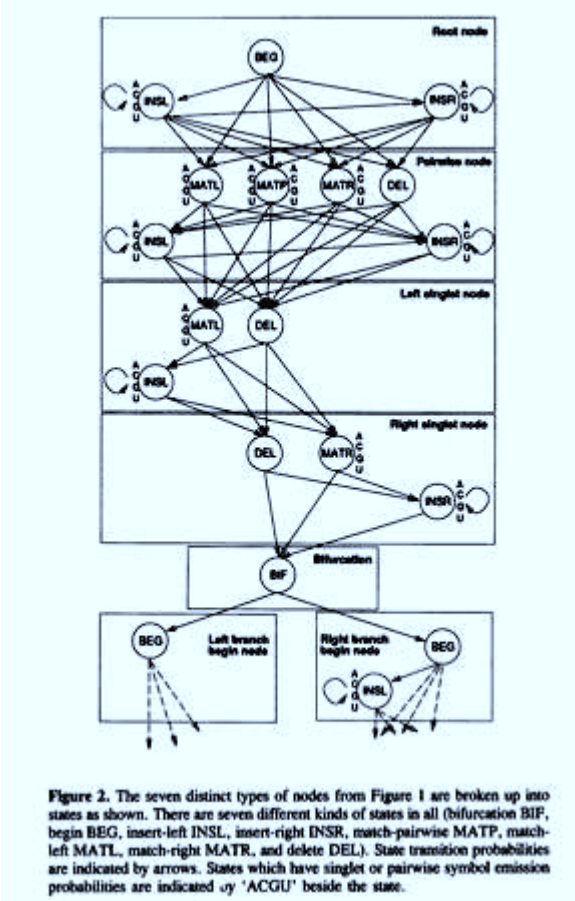

Figure 2. The seven distinct types of nodes from Figure 1 are broken up into states as shown. There are seven different kinds of states in all (bifurcation BIF, begin BEG, insert-left INSL, insert-right INSR, match-pairwise MATP, match-left MATL, match-right MATR, and delete DEL). State transition probabilities are indicated by arrows. States which have singlet or pairwise symbol emission probabilities are indicated by 'ACGU' beside the state.

Figure 3: An example of a covariance model showing the different node types and state types.

3.2 Alignment

All of the analysis for CMs rely on the basic algorithm for aligning a given RNA sequence to a CM and calculating this alignment's probability score. Multiple sequence alignments are then performed by aligning sequences one-by-one to the CM. Model training essentially optimizes model parameters and structure so that training sequences are assigned high alignment scores by the resulting model. Secondary structure prediction is easily derived from looking at which bases are assigned to which match states in an alignment. Finally, database searching for homologous RNAs is accomplished by looking for database subsequences with the highest alignment scores.

The CM alignment algorithm is oftentimes referred to as the Inside algorithm, and is a variant of the Inside algorithm for SCFGs. Let's suppose that the RNA

sequence x to be aligned has L symbols denoted as $x_1, ..., x_L$, and that the CM has M states. This algorithm uses a 3-dimensional dynamic programming algorithm to calculate the values $S_{i,j,y}$ which are the log likelihoods of alignments of subsequences of x from $1 \leq x_i ... x_j \leq L$ to subtrees of the CM beginning in state y where $1 \leq y \leq M$. The states of the CM are numbered from the root such that children states of a given state always have higher indices than their parent state via preorder traversal.

The initialization and recurrence for calculating the $S_{i,j,y}$ are broken down into cases depending on what type of state y happens to be. Each $S_{i,j,y}$ is the sum of three numbers: (1) the symbol emission log probability for emitting a given symbol(s) at state y in either location i and/or j, (2) the state transition log probability of transitioning to a given child state y_{next} of y, and (3) the score $S_{i',j',y_{next}}$ for subsequence $i'...j'$ for child state y_{next} which is found via recursion. The actual recurrences are messy and are given on page 2081 in the paper [4].

At the end of the algorithm, $S_{1,L,1}$ contains the score of the entire sequence's alignment. The alignment can be reconstructed by tracing back through the dynamic programming matrix and at each-state, following the max-score path. This Inside algorithm takes time cubic in the size of the sequence L and time linear in the size of the CM.

The key difference between this alignment algorithm and the Inside algorithm used for SCFGs-based analysis is that the CM alignment algorithm uses what is called the "Viterbi assumption." This assumes that the probability that the CM emits a given sequence is equal to the probability of the best alignment of the sequence to the model, ie the Viterbi alignment. The accuracy of this assumption hinges on whether or not the single Viterbi alignment dominates the sum of all possible alignments permitted by the model.

3.3 CM Training

The authors then present an algorithm for training a CM to a set of training sequenes. Given a set of training sequences, a training algorithm finds the CM which maximizes the likelihood of generating those sequences. Training actually involves two processes: estimating parameters of a given CM and determining an optimal structure of a CM.

Given a model structure, the training algorithm estimates the parameters for the CM via EM using the alignment algorithm on each training sequence to construct an optimal alignment to the current model. Re-estimates of transition and emission probabilities are calculated based on frequencies of observed state transitions and symbol emissions. This alignment of sequences to calculate new model parameters is repeated until the parameters converge to a local optimum.

But how do you define the original structure of the model? While this problem remains a bit of a black art (as it is with HMMs), the authors propose the following solution that uses the notion of *mutual information*. Given an alignment of the training sequences, mutual information calculates the amount of information we gain about the letter in column i of the alignment if we know the identify of the letter in column j. If f_{xi} is the frequency of nucleotide x in column i of the alignment, then we define the mutual information of column i and j as:

$$M_{ij} = \sum_{xi,xj} f_{xi,xj} * log_2 \frac{f_{xi,xj}}{f_{xi} * f_{xj}} \qquad (1)$$

where $f_{xi,xj}$ is the joint (pairwise) frequency of one of the sixteen possible base pairs observed in columns i and j. Basically, M_{ij} measures how much the joint frequency distribution deviates from the distribution expected if the two columns i and j varied independently. For RNA, M_{ij} varies between 0 and 2 bits. M_{ij} is maximal if columns i and j appear completely random when observed individually, but are perfectly correlated.

A dynamic programming algorithm using thes $M_{i,j}$ values can be used to calculate a tree that contains the consensus secondary structure of the training sequences and captures the maximum amount of correlation information. A traceback routine can recover this tree, and this tree is then as a guide to create the structure of the CM.

The full training process therefore consists of first creating a CM structure via mutual information from some initial sequence alignment. EM iterations then estimate the optimal parameters of the current CM. Once the parameters converge, the current alignment to the model is used to create a new CM structure using mutual information, and the EM iteration process begins again. This process can be repeated continually to produce a fully trained CM.

It turns out that this model estimation procedure works well even for unaligned training sequences when a random initial alignment must be guessed.

3.4 Database Searching

The algorithm for searching for homologous RNAs in a database is nearly identical to the alignment algorithm described earlier. In scanning the database, subsequences are aligned against the CM and their alignment scores are calculated. These scores can be used to calculate a log-likelihood ratio compared to a null model that this sequence is a homologous RNA. Appropriate thresholding can then be applied to distinguish which sequences most closely match the model.

3.5 Results

The paper presents results demonstrating the performance and accuracy of RNA covariance models. The authors make use of a database of 1415 aligned tRNA DNA sequences to select two randomly selected training sets each of size 100, and a random test set also of size 100. Two CMs were constructed from trusted alignments of the two training sets, and one CM was constructed from the trusted alignment of all 1415 tRNA sequences. In addition, two CMs were constructed from randomly selected sequences from an unaligned training set with no information about secondary structure.

Each model was used to produce multiple sequence alignments of the set of 100 test sequences. The alignments were then compared to the trusted alignment to determine accuracy. The models constructed from trusted alignments produced scores ranging from 93% to 95% correct. Surprisingly, the models constructed from unaligned sequences still produced scores ranging from 90% to 92% correct.

Finally, the authors compare their best model against the then state-of-the-art tRNA detection program TRNASCAN to see how well the model can detect homologous RNAs in a database search. The best-trained CM built from a trusted alignment of 1415 tRNA sequences outperformed TRNASCAN with respect to both sensitivity and specificity. In fact, the best-trained CM yielded better than 99.8% true positives with less than .2 false positives per Megabase searched.

The paper also presents a very interesting result showing the amount of pairwise correlation information (in bits) captured by the secondary structure of a CM. The results suggest that a CM captures about twice as much information about an RNA family as does an HMM which can only consider primary sequence information alone. Surprisingly, the results also show that tertiary structure information such as pseudoknots only contributes at most 2 to 3 additional bits of information. This suggests that capturing tertiary structure information in an RNA analysis model is not crucial for the task of database searching.

3.6 Discussion

This paper showed how covariance models can be used to create accurate models of RNA families suitable for performing multiple alignments, secondary structure prediction, and database searching for homologous RNAs.

It turns out that covariance models are really just an alternate formalism for the SCFGs described in [12] with a slight extension. The SCFG-based model does not allow for insertions and deletions in the RNA sequence. Thus, it is an ungapped model. CMs on the other hand, extend SCFG-based models by allowing for insertions and deletions (ie gaps), thereby creating a more accurate profile model of an RNA family. Such a model is better suited for searching databases at the DNA level. This idea is similar to how profile HMMs generalize the match-state-only HMMs (ie ungapped weight matrix models) typically used for DNA sequence analysis. Simply put, CMs are to SCFGs as profile HMMs are to match-state-only HMMs.

4 Taking Phylogeny Into Account

The profile SCFGs and covariance model methods we've considered so far use information from many RNA sequences to perform RNA analysis. In fact, these methods require a very large number of RNA sequences to create accurate prediction models. However, none of these methods takes into account the evolutionary history, or *phylogeny* of the sequences. Presumably, this extra information may make RNA analysis models even more accurate.

In this section, we consider two papers [8], [9] that take the phylogenetic tree of the RNA sequences into account to create a SCFG-based model suitable for RNA secondary structure prediction.

4.1 SCFGs and Phylogenetic Trees

We first consider the original paper by Knudsen and Hein [8]. Given an initial structural alignment of RNA sequences and a phylogenetic tree relating these sequences, the paper presents a two-part model composed of a SCFG and an evolutionary model that can be used to predict a single, common secondary structure describing the RNA sequences. In addition, if the phylogenetic tree is unavailable, the maximum likelihood (ML) estimate of the tree can be calculated from the model.

While the ideas presented in this paper are very interesting, the paper itself is unfortunately rather poorly written and organized. The authors are not clear in many of their definitions, and it is oftentimes difficult to understand how and where the phylogenetic information is incorporated. Nonetheless, the description below hopefully captures most of the paper's intuition.

The model consists of two parts: a SCFG and a mutational model, which is oftentimes confusingly referred to as an "evolutionary model" throughout the paper. The SCFG used in the paper is quite simple:

$$\begin{aligned} S &\rightarrow LS \mid L \\ F &\rightarrow sF\hat{s} \mid LS \\ L &\rightarrow s \mid sF\hat{s} \end{aligned}$$

where s symbolizes an unpaired base and $s\hat{s}$ symbolizes paired bases in a stem. That is, the $sF\hat{s}$ production results in the pairing of the surrounding bases s and $\hat{s}$.

This SCFG's parameters are estimated using the Inside-Outside algorithm described previously on a training set of folded RNAs.

The mutational model consists of two matrices that specify mutation rates between all possible bases X and Y and between all possible base-pairs XY and X'Y'. These mutational rates are also estimated from an initial training set of folded RNA sequences. This mutational model presumable captures some sort of evolutionary information about the set of RNA sequences in the training set.

The paper refers throughout to "the model" as the combination of the trained SCFG and the estimated mutation rates of the mutational model.

Suppose then that we are given a test set of RNA sequences and their structural alignment and phylogenetic tree relating the sequences. The paper shows how to use a trained model to predict a common secondary structure for this test set of sequences.

The first task involves calculating the column probabilities for both paired and unpaired columns in the test set's alignment. This can be done using the model's mutation rates and the test set's phylogenetic tree using a well-known tree post-order traversal procedure developed in the early 1980s.

Having calculated the column probabilities in the alignment using the model's mutation rates, it is possible to calculate the probability of the entire alignment $P(Align|T, M)$ given the phylogenetic tree T and the model M. This is done by summing over all possible secondary structures describing the alignment. Luckily enough, manipulation of this derivation results in a term equal to the product of the column probabilities in the alignment.

It turns out that this sum, and therefore the probability of the alignment given the model and the tree can be calculated using a dynamic programming algorithm that extends the view of the model's grammar to generate productions over columns of the alignment. When an s is used in a production rule, it instead refers to a column in the alignment. Since such a column has a probability that we've already calculated, then we can multiply this probability to the production probability each time an s is produced by the grammar. A similar trick is performed for production rules producing base pairs. By viewing the grammar in this way, we get a SCFG that is based on the trained model but generates columns in alignments.

Finding the most likely, common secondary structure that represents the test set's alignment is then easy to do. Simply apply the CYK algorithm for SCFGs that we saw previously to this "extended" grammar over columns in alignments.

Finally, the paper also points out that the ML estimate of the phylogenetic tree can be calculated from the model if the tree is not given. Given a tree topology, this calculation essentially optimizes the branch lengths to find the tree T with this topology that maximizes the alignment probability $P(Align|T, M)$. The resulting tree is the ML estimate of the phylogenetic tree. However, the authors brush over how to actually estimate a good tree topology saying that exhaustive search, branch and bound, or heuristic methods can be used.

4.1.1 Results

The authors build a model (both a SCFG and the mutational or evolutionary model) using a training database of tRNAs and large subunit ribosomal RNAS (LSU rRNAs) with well-known and well-established secondary structures. The database was reduced slightly by removing sequences with unknown bases.

Single base frequencies and base pair frequencies were estimated from counts of corresponding positions in the training set of sequences. To estimate mutation rates, all possible ordered pairs of sequences containing at least 85% identical base sequences were considered. The single base positions in the pairs were examined for any differences in observed bases. Counters kept track of the number of times base X and base Y were observed at the same position in different sequences. These counts were then used to estimate the rates of mutation between all possible bases X and Y. A similar technique was used to estimate base-pair mutation rates.

Finally, the parameters of the SCFG part of the model were estimated using the standard Inside-Outside algorithm applied to the same database of training sequences.

Having constructed a representative model of RNA sequences involving evolutionary history, the authors apply their model to predicting the secondary structure of a set of 4 representative bacterial RNase P RNA sequences with known structures, phylogenetic tree, and alignment. The 4 sequences were grouped into all possible pairs and triples to see how well secondary structure prediction accuracy improved as more sequences were added to the alignment. The authors found that there were very significant improvements in prediction accuracy when sequences were added.

The authors also found that if they used Clustal W to perform the initial alignment, the prediction results still showed a high level of accuracy. The results were compared against prediction results when no phylogenetic information was taken into account. This was done by greatly extending branch lengths in the phylogenetic tree so as to essentially make the 4 sequences independent. Results indicated that adding phylogenetic information improves the accuracy of the results by roughly 5%.

Further comparisons were made between the covariance model methods that we previously examined in [4]. The results showed that phylogenetic information can

make up for only having a small number of sequences in the test set's alignment.

4.2 Pfold: Improved Incorporation of Phylogenetic Information

The second paper [9] we consider is by the same authors and improves on the algorithm we just saw for incorporating evolutionary information into probabilistic models of RNA structure.

The paper refers to the methods in [8] for predicting RNA secondary structure as the KH-99 algorithm. This paper improves upon the KH-99 algorithm by making it faster and more robust to alignment errors. The new method is called Pfold, and its implementation is still used as a standard RNA secondary structure prediction tool.

The Pfold algorithm makes the following changes to the KH-99 algorithm:

1. gaps in an alignment are treated as unknown nucleotides with probability one for any nucleotide.

2. any nucleotide has a 1% probability of being any other nucleotide. This allows the algorithm to be more robust because it helps keep single sequences with slightly different structures from significantly changing the predicted consensus structure.

3. estimation of the phylogenetic tree is no longer performed through the time-intensive ML estimation procedure. Instead, standard estimation procedures are employed.

4. instead of using the CYK algorithm to find the most likely secondary structure, a different algorithm is used that finds the structure with the highest expected number of correctly predicted positions in the alignment.

4.2.1 Results

Pfold was tested on the same test set used in the original paper [8]. The results were essentially identical except that Pfold showed better performance. Most importantly, the improvement in computation time allows Pfold to consider many more sequences than the KH-99 algorithm. This makes the algorithm more practical for RNA secondary structure prediction.

The authors find that given a good initial structural alignment, prediction of a consensus secondary structure improves as the evolutionary distance of the aligned sequences increases. This is because more covariation information is yielded by larger evolutionary distances. However, larger distances necessarily make the sequences more difficult to align, and a single consensus structure may not accurately represent many diverged sequences. When using pairs of sequences i and j for an alignment, the authors show that accuracy seems to increase until an evolutionary distance of roughly 0.60 as defined by the "Jukes-Cantor" distance d_{ij}:

$$d_{ij} = -\frac{3}{4} log(1 - 4f/3) \tag{2}$$

where f is the fraction of sites u where the base at position u in sequence i and j differ in the pairwise alignment [3].

The authors point out that their improved Pfold algorithm still lacks a grammar that more closely describes real RNA structures. They claim that if such a grammar was incorporated, the accuracy of the method would approach state-of-the-art energy minimization methods such as Mfold [14] for a single sequence.

5 Probabilistic Models for Detecting Noncoding RNAs

In this section we consider two papers that look at whether or not probabilistic models of RNA structure can be used to effectively detect novel noncoding RNAs (ncRNAs) in a large genome sequence [10], [11]. Because ncRNA genes contain a much smaller amount of statistical information than protein-coding genes, it is difficult to find a statistically significant signal for ncRNA detection. How then do probabilistic models of RNA structure perform with such a weak signal? We also briefly discuss a third paper that proposes practical speed-ups for some of the developed scanning algorithms.

5.1 Detecting ncRNAs using Secondary Structure

The first paper we consider [10] considers whether or not RNA secondary structure is a strong enough statistical signal for detecting new ncRNAs, and whether or not a standard SCFG-based model can detect ncRNA genes in a genome.

The paper constructs two models of RNA folding. Both of these models are based on a straightforward SCFG that models RNA secondary structure in a manner similar to what we've already discussed. The first model, referred to as the "probabilistic model," uses training RNA sequences to derive probability parameters for the grammar via the standard Inside-Outside algorithm. The second model, referred to as the "thermodynamic model," instead derives these parameters from previous experimentally determined thermodynamic information.

A scanning algorithm for finding new RNAs in a genome sequence is then developed for each model. For the probabilistic model, the scanning algorithm essentially uses the Inside algorithm to calculate log-odds

scores for every subsequence of length $\leq w$ in the genome sequence. This log-odds score is the log ratio of the likelihood that this subsequence was generated by the model to the likelihood that it was generated by a null model. The null model emits nucleotides according to an estimated ncRNA base composition. Appropriate thresholding can then be applied to the scores to determine whether or not a subsequence is a match to the model.

The thermodynamic model's scanning algorithm is only slightly different, scoring only subsequences of a fixed length w and using a CYK algorithm to calculate Z-scores instead of log-odds scores. A Z-score calculates how many standard deviations the folding energy of the subsequence differs from the average energy of all permutations of the subsequence.

5.1.1 Results

Initial results were promising for the probabilistic model's scanning algorithm. Using a log-odds ratio larger than 9.1, the authors determine that scanning with a maximum window length of $w = 100$ nucleotides will yield no more than 10 false positives per megabase. The hits found when testing a number of sequences with known RNA genes were clearly above this threshold. However, they also observed a clear correlation between the relative CG bias in the RNA genes and the strength of the detected signal.

To understand what role the effects of this CG bias plays in their scanning algorithm, they composed a new scanning algorithm that simply looks for CG bias in a genome to find RNA genes. The results of using this scanning algorithm were remarkably similar to the scanning performed with structural information. Even after randomly shuffling the RNA gene sequences, the structural scanning algorithm was still largely unaffected. In addition, embedding an RNA gene into a random sequence of identical base composition resulted in an inability of the structural scanning algorithm to find the RNA gene.

These tests unfortunately suggested that the detector was actually latching onto the increased amount of CG base composition of RNA genes as opposed to any secondary structure signal. These experiments lead the authors to conclude that probabilistic models can detect ncRNAs in a genome simply because of a CG base composition bias and not as the result of secondary structure information.

On the flip-side, similar results were also discovered for the thermodynamic model's scanning algorithm. In fact, these results were even more discouraging since Z-scores assume that the background composition is random. This is certainly not the case in most genome sequences.

The authors reluctantly conclude that then state-of-the-art probabilistic models and algorithms for RNA secondary structure prediction are unable to distinguish ncRNAs from random sequences. This negative result holds true even if the model's parameters are based on thermodynamic information.

5.2 Detecting ncRNAs using Comparative Sequence Analysis

In this section we consider an extension of the work of [10] for detecting ncRNAs [11]. This paper attempts to use comparative sequence analysis between aligned pairs of homologous sequences (for example, a pairwise alignment of two related genomes) to find RNA genes.

The basic idea is that we can find conserved RNAs by looking for locations in the alignment where the pattern of mutation between bases and base-pairs suggests a region of conserved secondary structure. Since the secondary structure of RNAs tends to be more important than the primary sequence, then it makes sense to use comparative sequence analysis to find regions where base and base-pair mutations exist that still conserve a common secondary structure.

The authors note though that this technique will only work for conserved structural ncRNAs.

The authors construct three probabilistic models of structurally conserved RNA pairwise alignments that can be used to scan for structurally conserved RNAs in a given genome pairwise alignment. The basic idea is very similar to the scanning algorithm proposed in [10]. The main difference is that the models generate aligned pairs of RNA sequences instead of a single RNA sequence. That way, the scanning algorithm can scan a pair of aligned genomes to find structural conserved RNA regions that match the model.

The model that is particularly relevant to our analysis is based on a *pair stochastic context-free grammar* (Pair-SCFG). Such a grammar is identical in nature to the SCFGs we've already seen. However, Pair-SCFGs generate a pair of sequences simultaneously. That is, each production generates symbols in two different sequences at the same time. The authors extend the SCFG of [10] to a pair-SCFG that generates two aligned RNA sequences.

After training the pair-SCFG model, one can then perform a scanning algorithm on a pairwise aligned set of genome sequences to find any matches to the model. This scan makes use of an extension of the Inside algorithm for SCFGs in order to calculate log-odds scores of all pairs of aligned subsequences of length $\leq w$ in the genome alignment. Once again, appropriate thresholding can be applied to determine which pairs of subsequences most closely match the model.

However, it is usually desirable to forget about the input pairwise alignment and allow the pair-SCFG to opti-

mally realign the input sequences to the model. This is typically desirable since input alignments oftentimes will not have considered secondary structure. Thus, the alignment around structurally conserved RNA regions may be suboptimal. However, the authors point out that realigning a pair of sequences of length m and length n respectively to a pair-SCFG takes time $O(n^3m^3)$, which is far too prohibitive.

5.2.1 Results

The authors first test their trained models, whose implementation was called QRNA, on simulated aligned sequences to see how base composition affects detection. Wary of the tendency for these SCFG-based models to latch onto CG bias in the sequences, the authors generated a range of CG composition sequences for testing the scanning algorithm on. They found again that the specificity of the algorithm degrades as the CG bias fluctuates outside of a content range of 45% to 60%. To deal with this, they tuned their model by setting the parameters based on the CG composition of the input alignment. However crude this method, it does seem to offset the bias created by CG-rich input sequences.

They then took single RNA genes from a given family and used these sequences as a BLASTN query to find all genes in the same family. They retained those found BLASTN alignments greater than 50 nucleotides in length, with an E-value ≤ 0.01, and with an overall similarity of $\geq 65\%$. These resulting alignments were then run as input to QRNA to score whether or not they were RNAs. The authors found that there is a tradeoff between needing a BLASTN alignment that is correct enough but also dissimilar enough to highlight mutations in base-pairs of the RNA secondary structure. The authors choose to analyze only BLASTN alignments in a "sweet spot" of between 65% and 85% nucleotide identity.

Testing against an entire genome was done by aligning the E. Coli and Salmonella typhi genomes using known BLASTN alignments. The genomes were broken into 3 components. The scanning algorithm used a threshold log-odds score of 5 bits to classify RNAs in windows of 200 nucleotides advanced at 50 nucleotide intervals. Results were not overly promising, showing that only 29% of known RNAs were detected using this scanning algorithm.

The authors suggest that the failure was largely due to the lack of BLASTN alignments that fell in this "sweet spot" identify range. Therefore, the authors conclude that non-structurally aware initial alignments are the limiting factor in such a probabilistic-based scanning algorithm's detection accuracy. Clearly, performing an initial, optimal realignment of the sequences to the model would be helpful, although this was not undertaken due to the vast computational overhead needed.

5.3 Speeding-up Pair-SCFG Alignment for ncRNA Detection

A fairly recent paper [7] looks at ways to speed up the alignment of pairs of sequences to a pair-SCFG. In particular, the paper shows how to speed up the standard Inside and CYK algorithms for pair-SCFGs needed by realignment and scanning algorithms like those discussed in [11].

The details of the algorithms are complicated, but the intuition is as follows. The algorithms for pair-SCFGs work by computing likelihoods for every possible pair of substrings of the two sequences. However, if the secondary structure of each RNA sequence is known (or we have a best guess for the structure), then the recursion in the algorithms can be limited to considering only pairs of substrings consistent with the known secondary structure. The term "fold envelope" is used to describe a valid set of substrings consistent with the known structure for any one sequence in the alignment. Computing likelihoods conditioned on secondary structures is then transformed to computing fold envelopes.

These new algorithms can show dramatic increase in speed. In fact, for some extreme cases, the reduced running time is only $O(mn)$ where n and m are the lengths of each of the aligned sequences respectively!

6 SCFG Design Considerations

None of the papers we have considered discusses how to choose an underlying grammar for a SCFG-based model. Indeed, more complex grammars lead to more parameters that need to be estimated from training data. However, overly simple grammars may fail to capture important relationships or may bias prediction results towards a very narrow model.

A recent paper [2] attempts to answer the question of what small, simple SCFG designs are best for RNA secondary structure prediction?

This paper defines the notion of "structural ambiguity" of a grammar: a grammar G is structurally unambiguous if for every RNA sequence x generated by G, every unique secondary structure of x has only one parse tree in G that represents it. The authors show that if the grammar is not structurally unambiguous, then the CYK algorithm for determining the optimal secondary structure can yield suboptimal results. This turned out to be a practical concern for most of the grammars tested. As a result, the authors therefore focus on developing a set of small, structurally unambiguous grammars to see how well they perform.

The authors compared their set of grammars against standard energy minimization methods for predicting

RNA secondary structure. They found that the grammar used by Knudsen and Hein in Pfold [9] and discussed previously performs quite well considering its simplicity.

One minor shortcoming of this paper is that the authors provide no formal proofs that their grammars are structurally unambiguous. In a recent paper, we modify some of their formalisms to provide a framework for proving grammars to be structurally unambiguous. We then use this framework to prove that all of the grammars in the paper are in fact structurally unambiguous as needed.

7 Non-Probabilistic Methods for RNA Secondary Structure Analysis: Overview

In this section we briefly mention some of the standard non-probabilistic methods for RNA secondary structure analysis. The goal of this paper has been to look at probabilistic methods. However, it is important to understand that many standard and popular non-probabilistic methods also exist and are used with varying degrees of success.

7.1 Minimum Free Energy Methods

Perhaps the most popular method of RNA secondary structure prediction uses a technique known as the *minimum free energy method* (MFE) for folding a single RNA sequence. This method uses thermodynamic parameters to construct a secondary structure of a single RNA sequence that minimizes its total free energy. Molecules with less free energy are typically more stable structures in nature. Therefore, a minimum energy structure represents an optimal energetic folding. This technique is employed in two very popular implementation packages called Mfold [14] and RNAfold [6]. These two packages have been around for many years and are still commonly used as *the* benchmark for evaluating new probabilistic methods.

This technique is usually augmented in practice by combining a comparative sequence analysis approach. Many methods of comparative sequence analysis using MFE-style heuristics have been applied. While a detailed discussion of such techniques is beyond the scope of this paper, a good overview of these various methods can be found in a recent survey paper by Gardner and Giegerich [5].

7.2 Maximum Weighted Matching Method

Another recent method for RNA secondary structure prediction uses a graph-theoretic approach [13]. This method is particularly clever because of its simplicity. The majority of methods we've discussed are incapable of incorporating tertiary structure information (such as pseudoknots or base-triples) into the predicted RNA structures. This paper proposes using a *maximum weighted matching method* for RNA secondary structure prediction that can model any possible tertiary structure.

The basic idea is simple enough. Given an RNA sequence, construct a graph as follows: for every base in the sequence, add a vertex to the graph. Then, add edges between all possible pairs of vertices. Edges are given weights based on either phylogenetic or thermodynamic scores between possible base-pairs (such as mutual information that we discussed previously), experimental data, or a combination of both. A standard algorithm can then be applied to find the matching in the graph where the sum of the edge weights in the matching is maximal (maximal weighted matching). The algorithm is called "Gabow's algorithm" and works by continually building the optimal weighted matching for k edges by augmenting the optimal weighted matching using $k-1$ edges. The resulting matching corresponds to what bases are paired in the RNA sequence.

The paper of [13] also implements a clever post-processing filter that removes spurious base-pairs from the predicted structure. It does so by removing matchings that resulted because of intermediate matching performed during the augmentation steps of Gabow's algorithm. It turns out that important matchings tend to remain fixed throughout the course of the algorithm.

The great promise of this technique is that by simply playing with the structure of the graph and the edge weights, any type of base-pair relationship, including tertiary relationships, in RNA structures can be predicted. The paper implements an RNA folding algorithm using the maximum weighted matching method with a number of edge weight heuristics. The paper finds that the predicted structures are oftentimes nearly as good as the structures determined manually through lab experiments.

8 Summary

In this paper, we've attempted to provide a literature review of some of the most important papers on probabilistic methods for RNA secondary structure analysis. We examined the original papers on SCFG-based and CM-based analysis. From there we showed how more recent work has attempted to incorporate phylogenetic information into the models. We also discussed newer techniques for detecting ncRNAs using a combination of comparative sequence analysis and SCFG-based models, and how to address issues of algorithm speed and efficiency for the related Pair-SCFG-models. The issue of SCFG design was also addressed. Finally, we provided a brief overview of some of the standard non-probabilistic methods that exist as further reading for anyone interested in RNA secondary structure analysis.

We should note that there are still a great many variations on the basic probabilistic models we discussed. For example, there has been a good deal of work on combining energy calculations with evolutionary information for RNA secondary structure prediction. The curious reader should find ample resources in the reference sections of the papers discussed in this paper.

References

[1] http://bioinf.man.ac.uk/resources/phase/ manual/node72.html.

[2] Robin D. Dowell and Sean R. Eddy. Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. *BMC Bioinformatics*, 5(71), 2004.

[3] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: Probabilistic models of proteins and nucleic acids, 1998.

[4] Sean R. Eddy and Richard Durbin. Rna sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[5] Paul P. Gardner and Robert Giegerich. A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinformatics*, 5(140), 2004.

[6] IL Hofacker, W Fontana, S Bonhoeffer, and PF Stadler. Fast folding and comparison of rna secondary structures. *Monatshefte fur Chemie*, 125:167–188, 1994.

[7] I. Holmes and G.M. Rubin. Pairwise rna structure comparison with stochastic context-free grammars. In *Proceedings of Pacific Symposium on Biocomputing*, pages 163–174, 2002.

[8] B. Knudsen and J. Hein. Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15(6):446–454, 1999.

[9] B. Knudsen and J. Hein. Pfold: Rna secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13):3423–3428, 2003.

[10] Elena Rivas and Sean R. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding rnas. *Bioinformatics*, 16(7):583–605, 2000.

[11] Elena Rivas and Sean R. Eddy. Noncoding rna gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(8), 2001.

[12] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjolander, Rebecca C. Underwood, and David Haussler. Stochastic context-free grammars for trna modeling. *Nucleic Acids Research*, 22(23):5112–5120, 1994.

[13] Jack E. Tabaska, Robert B. Cary, Harold N. Gabow, and Gary D. Stormo. An rna folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, 14(8):691–699, 1998.

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