Recent Methods for RNA Modeling Using Stochastic Context-Free Grammars

Yasubumi Sakakibara[1] [⋆], Michael Brown[1], Richard Hughey[2], I. Saira Mian[3], Kimmen Sjölander[1], Rebecca C. Underwood[1], David Haussler[1]

[1] Computer and Information Sciences
[2] Computer Engineering
[3] Sinsheimer Laboratories
University of California, Santa Cruz, CA 95064, USA
Email: haussler@cse.ucsc.edu

Abstract. Stochastic context-free grammars (SCFGs) can be applied to the problems of folding, aligning and modeling families of homologous RNA sequences. SCFGs capture the sequences' common primary and secondary structure and generalize the hidden Markov models (HMMs) used in related work on protein and DNA. This paper discusses our new algorithm, Tree-Grammar EM, for deducing SCFG parameters automatically from unaligned, unfolded training sequences. Tree-Grammar EM, a generalization of the HMM forward-backward algorithm, is based on tree grammars and is faster than the previously proposed inside-outside SCFG training algorithm. Independently, Sean Eddy and Richard Durbin have introduced a trainable "covariance model" (CM) to perform similar tasks. We compare and contrast our methods with theirs.

Tools for analyzing RNA will become increasingly important as *in vitro* evolution and selection techniques produce greater numbers of synthesized RNA families to supplement those related by phylogeny. Recent efforts have applied *stochastic context-free grammars* (SCFGs) to the problems of statistical modeling, multiple alignment, discrimination and prediction of the secondary structure of RNA families. Our approach in applying SCFGs to modeling RNA is highly related to our work on modeling protein families and domains with HMMs [HKMS93, KBM$^+$94].

In RNA, the nucleotides adenine (`A`), cytosine (`C`), guanine (`G`) and uracil (`U`) interact to form characteristic secondary-structure motifs such as helices, loops and bulges [Sae84, WPT89]. Intramolecular `A`-`U` and `G`-`C` Watson-Crick pairs as well as `G`-`U` and, more rarely, `G`-`A` base pairs constitute the so-called *biological palindromes* in the genome. When RNA sequences are aligned, both primary

⋆ Y. Sakakibara's current address is ISIS, Fujitsu Labs Ltd., 140, Miyamoto, Numazu, Shizuoka 410-03, Japan. We thank Anders Krogh, Harry Noller and Bryn Weiser for discussions and assistance, and Michael Waterman and David Searls for discussions. This work was supported by NSF grants CDA-9115268 and IRI-9123692 and NIH grant number GM17129. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

and secondary structure need to be considered since generation of a multiple alignment and analysis of folding are mutually dependent. (A multiple alignment of RNA sequences is a list of the sequences with the letters representing nucleotides spaced such that nucleotides considered functionally equivalent have their letters appearing in the same column in the list. To enhance the alignment of some sequences with respect to others, spaces may need to be inserted in them.) Elucidation of common folding patterns among multiple sequences may indicate the pertinent regions to be aligned and vice versa [San85].

Two principal methods have been established for predicting RNA secondary structure (i.e., which nucleotides are base-paired). The first technique, phylogenetic analysis of homologous RNA molecules [FW75, WGGN83], ascertains structural features that are conserved during evolution. The second technique employs thermodynamics to compare the free energy changes predicted for formation of possible secondary structure and relies on finding the structure with the lowest free energy [TUL71, TSF88, Gou87]. Though in principle HMMs could also be used to model RNA, the standard HMM approach treats all positions as having independent distributions and is unable to model the interactions between positions. However, if two positions in an alignment are base-paired, then the bases at these positions will be highly correlated. Since base-pairing interactions play such a dominant role in determining RNA structure and function, any statistical method for modeling RNA that does not consider these interactions will encounter insurmountable problems.

In this work, we use formal language theory to describe a means to generalize HMMs to model most RNA interactions. We compare our method to Eddy and Durbin's use of "covariance models" (CMs) to model RNA [ED94]. CMs are equivalent to SCFGs, but Eddy and Durbin employ different algorithms for training and producing multiple alignments.

As in the elegant work of Searls [Sea92], we view the character strings representing DNA, RNA and protein as sentences derived from a formal grammar. In the simplest kind of grammar, a *regular* grammar, strings are derived from productions (rewriting rules) of the forms $S \rightarrow aS$ and $S \rightarrow a$, where S is a *nonterminal symbol*, which does not appear in the final string, and a is a *terminal symbol*, which appears as a letter in the final string. Searls has shown that base pairing in RNA can be described by a *context-free grammar* (CFG). CFGs are often used to define the syntax of programming languages. A CFG is more powerful than a regular grammar in that it permits a greater variety of productions, such as those of the forms $S \rightarrow SS$ and $S \rightarrow aSa$. As described by Searls, these additional types of productions are crucial to model the base-pairing structure in RNA. (CFGs can not describe all RNA structure, but we believe they can account for enough to make useful models.) Productions of the forms $S \rightarrow \texttt{A}\ S\ \texttt{U}$, $S \rightarrow \texttt{U}\ S\ \texttt{A}$, $S \rightarrow \texttt{G}\ S\ \texttt{C}$ and $S \rightarrow \texttt{C}\ S\ \texttt{G}$ describe the structure in RNA due to Watson-Crick base pairing. Using productions of this type, a CFG can specify the language of biological palindromes.

Searls' original work [Sea92] argues the benefits of using CFGs as models for RNA folding, but does not discuss stochastic grammars or methods for creating the grammar from training sequences. Recent work provides an effective method for building a stochastic context-free grammar (SCFG) to model a family of RNA sequences. Some analogs of stochastic grammars and training methods do appear in Searls' most recent work in the form of costs and other trainable parameters used during parsing [Sea93a, Sea93b, SD93], but we believe that our integrated probabilistic framework may prove to be a simpler and more effective approach.

We have designed an algorithm that deduces grammar parameters automatically from a set of unaligned primary sequences. It is a novel generalization of the *forward-backward algorithm* commonly used to train HMMs. Our algorithm, Tree-Grammar EM, is based on tree grammars [TW68] and is more efficient than the *inside-outside algorithm* [LY90], a computationally expensive generalization of the forward-backward algorithm developed to train SCFGs [Bak79]. Full details are described elsewhere [SBH+93]; here we present a summary.

1 Stochastic Context-Free Grammar Methods

Specifying a probability for each production in a grammar yields a *stochastic* grammar. A stochastic grammar assigns a probability to each string it derives. Stochastic regular grammars are equivalent to HMMs and suggest an interesting generalization from HMMs to SCFGs [Bak79]. In this work, we explore stochastic models for tRNA sequences using a stochastic context-free grammar that is similar to our HMMs [KBM+94] but incorporates base-pairing information.

1.1 Context-free grammars for RNA

A grammar is principally a set of productions (rewrite rules) that is used to generate a set of strings, a *language*. The productions are applied iteratively to generate a string in a process called *derivation*. For example, application of the productions in Figure 1 could generate the RNA sequence **CAUCAGGGAAGAUCUCUUG** by the following derivation:

Beginning with the start symbol S_0, any production with S_0 left of the arrow can be chosen to have its right side replace S_0. If the production $S_0 \rightarrow S_1$ is selected (in this case, this is the only production available), then the symbol S_1 replaces S_0. This derivation step is written $S_0 \Rightarrow S_1$, where the double arrow signifies application of a production. Next, if the production $S_1 \rightarrow \texttt{C}\ S_2\ \texttt{G}$ is selected, the derivation step is $S_1 \Rightarrow \texttt{C}\ S_2\ \texttt{G}$. Continuing with similar derivation steps, each time choosing a nonterminal symbol and replacing it with the right-hand side of an appropriate production, we obtain the following derivation terminating with the desired sequence:

$$\begin{aligned}
S_0 \Rightarrow\ & S_1 \ \Rightarrow\ \mathtt{C}S_2\mathtt{G} \ \Rightarrow\ \mathtt{CA}S_3\mathtt{UG} \ \Rightarrow\ \mathtt{CA}S_4S_9\mathtt{UG} \\
\Rightarrow\ & \mathtt{CAU}S_5\mathtt{A}S_9\mathtt{UG} \ \Rightarrow\ \mathtt{CAUC}S_6\mathtt{GA}S_9\mathtt{UG} \\
\Rightarrow\ & \mathtt{CAUCA}S_7\mathtt{GA}S_9\mathtt{UG} \ \Rightarrow\ \mathtt{CAUCAG}S_8\mathtt{GA}S_9\mathtt{UG} \\
\Rightarrow\ & \mathtt{CAUCAGGGA}S_9\mathtt{UG} \ \Rightarrow\ \mathtt{CAUCAGGGAA}S_{10}\mathtt{UUG} \\
\Rightarrow\ & \mathtt{CAUCAGGGAAG}S_{11}\mathtt{CUUG} \\
\Rightarrow\ & \mathtt{CAUCAGGGAAGA}S_{12}\mathtt{UCUUG} \\
\Rightarrow\ & \mathtt{CAUCAGGGAAGAU}S_{13}\mathtt{UCUUG} \\
\Rightarrow\ & \mathtt{CAUCAGGGAAGAUCUCUUG}.
\end{aligned}$$

$$\begin{array}{llll}
P = \{ & S_0 \rightarrow S_1, & S_7 \rightarrow \mathtt{G}\ S_8, & \\
& S_1 \rightarrow \mathtt{C}\ S_2\ \mathtt{G}, & S_8 \rightarrow \mathtt{G}, & \\
& S_1 \rightarrow \mathtt{A}\ S_2\ \mathtt{U}, & S_8 \rightarrow \mathtt{U}, & \\
& S_2 \rightarrow \mathtt{A}\ S_3\ \mathtt{U}, & S_9 \rightarrow \mathtt{A}\ S_{10}\ \mathtt{U}, & \\
& S_3 \rightarrow S_4\ \ S_9, & S_{10} \rightarrow \mathtt{C}\ S_{10}\ \mathtt{G}, & \\
& S_4 \rightarrow \mathtt{U}\ S_5\ \mathtt{A}, & S_{10} \rightarrow \mathtt{G}\ S_{11}\ \mathtt{C}, & \\
& S_5 \rightarrow \mathtt{C}\ S_6\ \mathtt{G}, & S_{11} \rightarrow \mathtt{A}\ S_{12}\ \mathtt{U}, & \\
& S_6 \rightarrow \mathtt{A}\ S_7, & S_{12} \rightarrow \mathtt{U}\ S_{13}, & \\
& S_7 \rightarrow \mathtt{U}\ S_7, & S_{13} \rightarrow \mathtt{C} & \}
\end{array}$$

Fig. 1. This set of productions P generates RNA sequences with a certain restricted structure. $S_0, S_1, \ldots, S_{13}$ are nonterminals; `A`, `U`, `G` and `C` are terminals representing the four nucleotides.

A derivation can be arranged in a tree structure called a *parse tree* (Figure 2). A parse tree represents the syntactic structure of a sequence produced by a grammar. For an RNA sequence, this syntactic structure corresponds to the physical secondary structure (Figure 3).

Formally, a context-free grammar G consists of a set of nonterminal symbols N, an alphabet of terminal symbols Σ, a set of productions P, and the start symbol S_0. For a nonempty set of symbols X, let X^* denote the set of all finite strings of symbols in X. Every CFG production has the form $S \rightarrow \alpha$ where $S \in N$ and $\alpha \in (N \cup \Sigma)^*$, thus the left-hand side consists of a single nonterminal while there is no restriction on the number or placement of nonterminals and terminals on the right-hand side. The production $S \rightarrow \alpha$ means that the nonterminal S can be replaced by the string α. If $S \rightarrow \alpha$ is a production in P, then for any strings γ and δ in $(N \cup \Sigma)^*$, we define $\gamma S \delta \Rightarrow \gamma\alpha\delta$ and we say that $\gamma S \delta$ *directly derives* $\gamma\alpha\delta$ in G. We say the string β can be *derived* from α, denoted $\alpha \overset{*}{\Rightarrow} \beta$, if there exists a sequence of direct derivations $\alpha_0 \Rightarrow \alpha_1$, $\alpha_1 \Rightarrow \alpha_2, \ldots, \alpha_{n-1} \Rightarrow \alpha_n$ such that $\alpha_0 = \alpha$, $\alpha_n = \beta$, $\alpha_i \in (N \cup \Sigma)^*$, and $n \geq 0$. Such a sequence is called a *derivation*. Thus, a derivation is an order of productions applied to generate a string. The grammar generates the language $\{w \in \Sigma^* \mid S_0 \overset{*}{\Rightarrow} w\}$, the set of all terminal strings w that can be derived from the grammar.

Our work in modeling RNA uses productions of the following forms: $S \rightarrow SS$, $S \rightarrow aSa$, $S \rightarrow aS$, $S \rightarrow S$ and $S \rightarrow a$, where S is a nonterminal and a is a terminal. $S \rightarrow aSa$ productions describe the base pairings in RNA; $S \rightarrow aS$ and $S \rightarrow a$ describe unpaired bases; $S \rightarrow SS$ describe branched secondary structures and $S \rightarrow S$ are used in the context of multiple alignments.

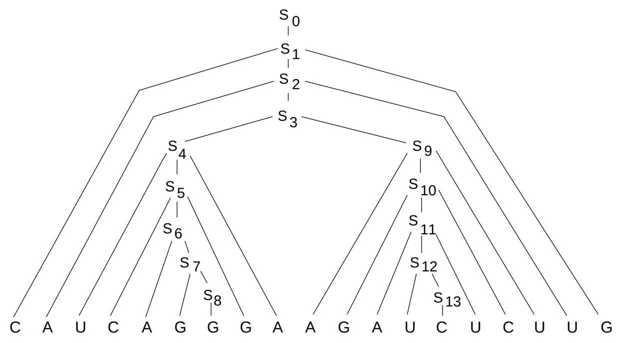

Fig. 2. For the RNA sequence CAUCAGGGAAGAUCUCUUG, the grammar whose productions are given in Figure 1 yields this parse tree which reflects a specific secondary structure (see Figure 3).

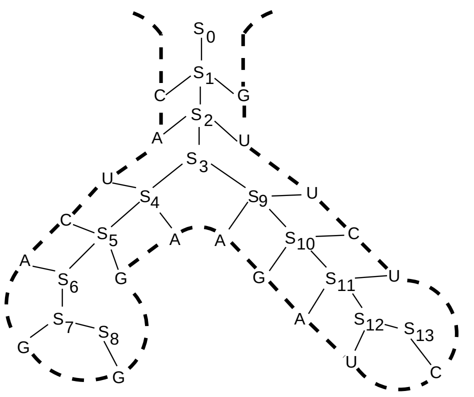

Fig. 3. For the RNA sequence CAUCAGGGAAGAUCUCUUG, the grammar whose productions are given in Figure 1 yields a parse tree (see Figure 2) which reflects this specific secondary structure.

SCFGs are a generalization of HMMs. The three main types of nonterminals in an SCFG correspond to each of the primary states in an HMM: *match, insert* and *skip* [HKMS93, KBM+94]. The match nonterminals in a grammar correspond to important structural positions in an RNA molecule. Insert nonterminals also generate nucleotides but with different distributions. These are used to model loops by inserting nucleotides between important (match) positions. Productions of the form $S \rightarrow S$ are called *skip productions* because they are used to skip a match nonterminal, so that no nucleotide appears at that position in a multiple alignment. In an SCFG, use of a skip production in parsing a sequence is equivalent to choice of a delete state in aligning a sequence to an HMM.

1.2 Stochastic context-free grammars

In an SCFG, every production for a nonterminal S has an associated probability value such that a probability distribution exists over the set of productions for S. (Any production with the nonterminal S on the left side is called "a production for S.") We denote the associated probability for a production $S \rightarrow \alpha$ by $\mathcal{P}(S \rightarrow \alpha)$.

A stochastic context-free grammar G generates sequences and assigns a probability to each generated sequence, thereby defining a probability distribution on the set of sequences. The probability of a derivation (parse tree) can be calculated as the product of the probabilities of the production instances applied to produce the derivation. The probability of a sequence s is the sum of probabilities over all possible derivations that G could use to generate s, written as follows:

$$\begin{aligned}\text{Prob}(s \mid G) &= \sum_{\substack{\text{all derivations} \\ \text{(or parse trees) } d}} \text{Prob}(S_0 \overset{d}{\Rightarrow} s \mid G) \\ &= \sum_{\alpha_1, \ \ldots \ , \alpha_n} \Big[\text{Prob}(S_0 \Rightarrow \alpha_1 \mid G) \cdot \ \text{Prob}(\alpha_1 \Rightarrow \alpha_2 \mid G) \cdot \ \cdots \\ &\qquad \cdot \text{Prob}(\alpha_n \Rightarrow s \mid G) \Big]\end{aligned}$$

where terms $\alpha_i \in (N \cup \Sigma)^*$.

Efficiently computing $\text{Prob}(s \mid G)$ presents a problem because the number of possible parse trees for s is exponential in the length of the sequence. However, a dynamic programming technique analogous to the Cocke-Younger-Kasami (CYK) or Early parsing methods [AU72] for non-stochastic CFGs can complete this task in polynomial time (proportional to the cube of the length of sequence s). We define the negative logarithm of the probability of a sequence given by the grammar G, $-\log(\text{Prob}(s \mid G))$, as the *negative log likelihood (NLL)* *score* of the sequence. The NLL score quantifies how well the sequence s fits the grammar—the likelihood that the grammar with its production probabilities would produce s.

CFGs have a drawback in that a sequence can sometimes be derived by a CFG in multiple ways. Since alternative parse trees reflect alternative secondary structures (foldings), a grammar may give several possible secondary structures for a single RNA sequence. An SCFG has the advantage that it can provide the most likely parse tree from this set of possibilities. If the productions are chosen carefully and the probabilities are estimated accurately, a parsing algorithm, when given grammar G and an RNA sequence s, will produce a most likely parse tree for s that corresponds to the correct secondary structure for s. Indeed, for most of the tRNA sequences we test, the most likely parse trees given by the tRNA-trained grammar match precisely the accepted secondary structures.

We can compute the most likely parse tree efficiently using a variant of the above procedure for calculating $\mathrm{Prob}(s \mid G)$. To obtain the most likely parse tree for the sequence s, we calculate

$$\max_{\text{parse trees } d} \mathrm{Prob}(S_0 \overset{d}{\Rightarrow} s \mid G).$$

The dynamic-programming procedure to do this resembles the Viterbi algorithm for HMMs [Rab89]. We also use this procedure to obtain multiple alignments: the parser aligns each sequence by finding the most likely parse tree given by the grammar, yielding an alignment of all nucleotides that correspond to the match nonterminals for each sequence, after which the mutual alignment of the sequences among themselves is determined. (Insertions of varying lengths can exist between match nonterminals, but by inserting enough spaces in each sequence to accommodate the longest insertion, an alignment of all the sequences is obtained.) This is equivalent to multiple alignment in an HMM, where the single most likely path for each sequence is computed.

1.3 Estimating SCFGs from sequences

Both an SCFG's production probabilities and the productions themselves can in principle be chosen using an existing alignment of RNA sequences. Results using this approach were reported in our previous work [SBU+93]. Also, Eddy and Durbin report recent results in which nearly all aspects of the grammar are determined solely from the training sequences [ED94]. In contrast, we make more use of prior information about the structure of tRNA to design an appropriate initial grammar, and then use training sequences only to refine our estimates of the probabilities of the productions used in this grammar.

1.4 The Tree-Grammar EM training algorithm

To estimate the SCFG parameters from unaligned training tRNA sequences, we introduce Tree-Grammar EM (Figure 4), a new method for training SCFGs that uses a generalization of the forward-backward algorithm commonly used to train

HMMs. This generalization, called TG Reestimator, is more efficient than the inside-outside algorithm, which was previously proposed to train SCFGs.

```
1. Start with an initial grammar $G_0$.
2. Use grammar $G_0$ and the CYK-like parsing algorithm to parse the raw input sequences, producing a tree representation for each sequence indicating which nucleotides are base-paired. This set of initial trees is denoted $T_0$. Set $T_{old} = \emptyset$ and $T_{new} = T_0$.
3. While $T_{new} \neq T_{old}$ do the following: {
    3a. Set $T_{old} = T_{new}$.
    3b. Use $T_{old}$ trees as input to our TG Reestimator algorithm, which iteratively re-estimates the grammar parameters until they stabilize. The grammar with the final stabilized probability values is called new grammar $G_{new}$.
    3c. Use grammar $G_{new}$ and the CYK-like parsing algorithm to parse the input sequences, producing a new set of trees $T_{new}$.
}
```

Fig. 4. Pseudocode for the Tree-Grammar EM training algorithm.

The inside-outside algorithm [LY90, Bak79] is an *expectation maximization* (EM) algorithm that calculates maximum likelihood estimates of an SCFG's parameters based on training data. However, it requires the grammar to be in Chomsky normal form, which is possible but inconvenient for modeling RNA (and requires more nonterminals). Further, it takes time at least proportional to $|s|^3$ per training sequence, whereas the forward-backward procedure for HMMs takes time proportional to $|s|$ per training sequence s, where $|s|$ is the length of s. In addition, the inside-outside algorithm is prone to settling in local minima; this presents a problem when the initial grammar is not highly constrained.

To avoid these problems, we developed an iterative estimator algorithm called *TG Reestimator* (Step 3b in Figure 4). While the running times of both Tree-Grammar EM and the inside-outside algorithm are asymptotically equivalent due to the use of the CYK-like algorithm (Step 3c of Figure 4), in practice Tree-Grammar EM is much more efficient. In Tree-Grammar EM, the inner loop (Step 3b) takes time proportional to $|s|$ per sequence per iteration (the grammar size is constant); in the inside-outside algorithm, the analogous step takes time proportional to $|s|^3$ per sequence per iteration. Since the number of iterations in Step 3b is typically on the order of 100, while the number of iterations of Step 3 is typically two or three, Tree-Grammar EM is more practical for longer RNA sequences.

TG Reestimator requires folded RNA sequences as training examples, rather than unfolded ones. Thus, some tentative "base pairs" in each training sequence have to be identified before TG Reestimator can begin. To do this, we design a rough initial grammar (see Section 1.6) that may represent only a portion of the base-pairing interactions (Step 1) and parse the unfolded RNA training sequences to obtain a set of partially folded RNA sequences (Step 2). Then we estimate a new SCFG using the partially folded sequences and TG Reestimator (Step 3b). Further productions might be added to the grammar at this step, though we have not yet experimented with this. The parameter re-estimation is then repeated. In this way, TG Reestimator can be used even when precise biological knowledge of the base pairing is not available. TG Reestimator constitutes one part of the entire training procedure, Tree-Grammar EM.

The Tree-Grammar EM procedure is based on the theory of stochastic tree grammars [TW68, Fu82]. Tree grammars are used to derive labeled trees instead of strings. Labeled trees can be used to represent the secondary structure of RNA easily [SZ90] (see Figure 2). A tree grammar for RNA denotes both the primary sequence and the secondary structure of each molecule. Since these are given explicitly in each training molecule, the TG Reestimator algorithm does not have to (implicitly) sum over *all* possible interpretations of the secondary structure of the training examples when re-estimating the grammar parameters, as the inside-outside method must do. The TG Reestimator algorithm iteratively finds the best parse for each molecule in the training set and then readjusts the production probabilities to maximize the probability of these parses. The new algorithm also tends to converge faster because each training example is much more informative [SBU$^+$93].

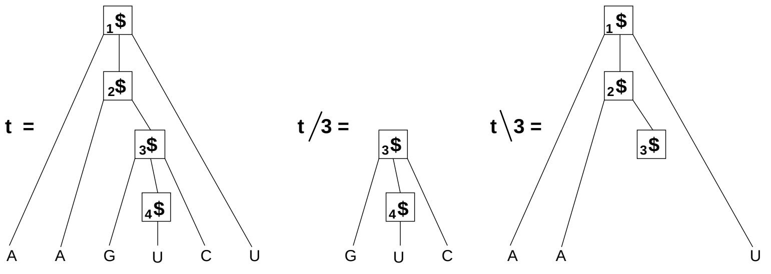

Fig. 5. The folded RNA sequence (AA(GUC)U) can be represented as a tree t (left), which can be broken into two parts such as $t/3$ (middle) and $t\backslash 3$ (right). The root, $_1\$$, and the internal node, $_3\$$, represent A-U and G-C base pairs, respectively.

To avoid unnecessary complexity, we describe this new algorithm in terms of CFGs instead of tree grammars [TW68, Sak92]. A tree is a rooted, directed, connected acyclic finite graph in which the direct successors of any node are linearly ordered from left to right. The predecessor of a node is called the *parent*; the successor, a *child*; and a child of the parent, a *sibling*. A folded RNA sequence can be represented by a labeled tree t as follows. Each leaf node is labeled by one of four nucleotides $\{A, U, G, C\}$ and all internal nodes are labeled by one special symbol, say \$. The sequence of nucleotides labeled at leaf nodes traced from left to right exactly constitutes the RNA sequence, and the structure of the tree represents its folding structure. See Figure 5 for an example of a tree representation of the folded RNA sequence `(AA(GUC)U)`. We assume all internal nodes in t are numbered from 1 to T (the number of internal nodes) in some order. For an internal node n ($1 \leq n \leq T$), let t/n denote the subtree of t with root n (Figure 5, center) and let $t \backslash n$ denote the tree obtained by removing a subtree t/n from t (Figure 5, right).

The probability of any folded sequence t given by an SCFG $G = (N, \Sigma, P, S_0)$ is calculated efficiently using a dynamic programming technique (as is done with the forward algorithm in HMMs). A labeled tree t representing a folded RNA sequence has the shape of a parse tree, so to parse the folded RNA, the grammar G needs only to assign nonterminals to each internal node according to the productions. Let $in_n(S)$ be the probability of the subtree t/n given that the nonterminal S is assigned to node n and given grammar G, for all nonterminals S and all nodes n such that $1 \leq n \leq T$. We can calculate $in_n(S)$ inductively as follows:

1. Initialization: $in_n(a) = 1$, for all leaf nodes n and all terminals a (all nucleotides). This extension of $in_n(S)$ is for the convenience of the inductive calculation of $in_n(S)$.

2. Induction:

$$in_m(S) = \sum_{\substack{Y_1,\ldots,Y_k \\ \in (N \cup \Sigma)}} in_{n_1}(Y_1) \ \cdots \ in_{n_k}(Y_k) \cdot \ \mathcal{P}(S \to Y_1 \ \cdots \ Y_k),$$

for all nonterminals S, all internal nodes m and all m's children nodes $n_1, \ldots, n_k$.

3. Termination: For the root node n and the start symbol S_0,

$$\mathrm{Prob}(t \mid G) \ = \ in_n(S_0). \tag{1}$$

We need one more quantity, $out_n(S)$, which defines the probability of $t \backslash n$ given that the nonterminal S is assigned to node n and given grammar G, which we obtain similarly.

1. Initialization: For the root node n,

$$out_n(S) \ = \ \begin{cases} 1 \text{ for } S = S_0 \text{ (start symbol)}, \\ \\ 0 \text{ otherwise.} \end{cases}$$

2. Induction:

$$out_m(S) = \sum_{\substack{Y_1,\ldots,Y_k \\ \in (N \cup \Sigma), \\ S' \in N}} in_{n_1}(Y_1) \ \cdots \ in_{n_k}(Y_k) \cdot \ \mathcal{P}(S' \to Y_1 \ \cdots \ S \ \cdots \ Y_k) \cdot \ out_l(S'),$$

for all nonterminals S, all internal nodes l and m such that l is m's parent and all nodes $n_1, \ldots, n_k$ are m's siblings. (There is no termination step given in this case because the calculation of $\mathrm{Prob}(t \mid G)$ is given in the termination step for $in_n(S)$.)

Given a set of folded training sequences $t(1), \ldots, t(n)$, we can determine how well a grammar fits the sequences by calculating the probability that the grammar generates them. This probability is simply a product of terms of the form given by (1), i.e.,

$$\text{Prob}(\text{sequences} \mid G) = \prod_{j=1}^{n} \text{Prob}(t(j) \mid G), \tag{2}$$

where each term Prob($t(j)$ | G) is calculated as in Equation (1). The goal is to obtain a high value for this probability, called the *likelihood* of the grammar. The *maximum likelihood* (ML) method of model estimation finds the model that maximizes the likelihood (2). There is no known way to directly and efficiently calculate the best model (the one that maximizes the likelihood) and avoid getting caught in suboptimal solutions during the search. However, the general EM method, given an arbitrary starting point, finds a local maximum by iteratively re-estimating the model such that the likelihood increases in each iteration, and often produces a solution that is acceptable, if perhaps not optimal. This method is often used in statistics.

Thus in Step 1 of Tree-Grammar EM, an initial grammar is created by assigning values to the production probabilities $\mathcal{P}(S \rightarrow Y_1 \ \cdots \ Y_k)$ for all S and all $Y_1, \ \ldots \ , Y_k$, where S is a nonterminal and Y_i $(1 \leq i \leq k)$ is a nonterminal or terminal. If some constraints or features present in the sequences' actual (trusted) multiple alignment are known, these are encoded in the initial grammar. The current grammar is set to this initial grammar.

In Step 3b of Tree-Grammar EM, using the current grammar, the values $in_n(S)$ and $out_n(S)$ for each nonterminal S and each node n for each folded training sequence are calculated in order to get a new estimate of each production probability, $\hat{\mathcal{P}}(S \rightarrow Y_1 \ \cdots \ Y_k)$=

$$\frac{\sum_{\text{all } t} \left(\sum_{\text{all } m} out_m(S) \cdot \mathcal{P}(S \rightarrow Y_1 \cdots Y_k) \cdot in_{n_1}(Y_1) \cdots in_{n_k}(Y_k) / \text{Prob}(t \mid G) \right)}{\text{norm}},$$

which is a double sum over all sequences t and all nodes m, where G is the old grammar and "norm" is the appropriate normalizing constant such that $\sum_{Y_1,\ldots,Y_k} \hat{\mathcal{P}}(S \rightarrow Y_1 \ \cdots \ Y_k) = 1$. A new current grammar is created by replacing all $\mathcal{P}(S \rightarrow Y_1 \cdots Y_k)$ with the re-estimated probabilities $\hat{\mathcal{P}}(S \rightarrow Y_1 \cdots Y_k)$.

1.5 Overfitting and regularization

Attempts to estimate a grammar with too many free parameters from a small set of training sequences will encounter the *overfitting* problem—i.e., the grammar fits the training sequences well, but poorly fits other, related (test) sequences. One solution is to use *regularization* to control the effective number of free parameters. We regularize our grammars by taking a Bayesian approach to the parameter estimation problem, similar to our approach with protein HMMs [KBM+94, BHK+93b].

Before training the grammars, we construct a prior probability density for each of their "important" parameter sets. This prior density takes the form of a Dirichlet distribution [SD89]. The "important" productions are of two forms: $S \rightarrow aSb$ and $S \rightarrow aS$, where terminal symbols $a, b \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. $S \rightarrow aSb$ productions, which generate base pairs, come in groups of 16, corresponding to all possible pairs of terminal symbols. $S \rightarrow aS$ productions, which generate nucleotides in loop regions, come in groups of four. For the base-pairing productions, we employ prior information about which productions are most likely. For instance, Watson-Crick pairs are more frequently observed than other base pairs. To calculate precise prior information about base-pair probabilities, we obtain the 16 parameters of a Dirichlet density over possible base-paired position distributions from a large alignment of 16S rRNA sequences [LOM$^+$93]. We similarly use the alignment to calculate a four-parameter Dirichlet prior for nucleotide distributions in loop region positions. (We present further details elsewhere [BHK$^+$93b].) These parameters constitute our regularizer. We add them as "pseudocounts" during each re-estimation step of Tree-Grammar EM (step 3b in Figure 4). Thus, at each iteration, TG Reestimator computes mean posterior estimates of the model parameters rather than maximum likelihood estimates.

In a similar manner, we regularize probability distributions for other production types, including chain rules $S \rightarrow S$, branch productions $S \rightarrow SS$ and insert productions $S \rightarrow aS$. We regularize loops with very large uniform pseudocounts over the four possible nucleotides so that their probability distributions will be fixed at uniform values rather than estimated from the training data. This is equivalent to the regularization we used for the insert states of HMMs [KBM$^+$94]. This further reduces the number of parameters to be estimated, helping to avoid overfitting.

1.6 The initial grammar

In the initial grammar, we model a loop that is typically l nucleotides in length by an HMM model with l match states as described in our previous protein work [KBM$^+$94], except that the four-letter nucleic-acids alphabet replaces the twenty-letter amino-acids alphabet. Nucleotide distributions in such a loop are defined by probabilities of l match-nonterminals' productions. Longer or shorter loops can be derived using special nonterminals and productions that allow position-specific insertions and deletions.

A helix l base pairs in length consists of l nonterminals. Each nonterminal has 16 productions that derive possible base pairs for its position in the helix. Each nonterminal has its own probability distribution over these 16 productions. These distributions, like those for match nonterminals in loops, are initially defined using Dirichlet priors (Section 1.5). Other nonterminals and productions are added to allow deletions of base pairs, enabling helix length variations.

Special treatment of nonterminals involved in branch productions of the form $S \rightarrow SS$ can also be included. In particular, we specify that certain branch productions may also, with some probability, omit one of the nonterminals on the right-hand side. This allows the grammar to derive tRNAs that lack certain substructures (such as arms or loops). These probability values are initialized to default prior values and then re-estimated during training on actual sequences, as are all grammar parameters.

1.7 The Eddy and Durbin algorithms

Sean Eddy and Richard Durbin have described an algorithm for obtaining an SCFG's productions themselves, as well as their probabilities (see Figure 6). As discussed in their work [ED94], covariance models (CMs) are describable as SCFGs. To deduce a covariance model's structure (essentially, to choose an SCFG's productions), they use the standard Nussinov/Zuker dynamic-programming algorithm for RNA folding [NPGK78, Zuk89], with the difference that the cost function being optimized is a function of the "mutual information" of two columns in the input multiple alignment (based on Gutell *et al.*'s MIXY procedure [GPH+92]), rather than of the number of base pairs or the thermodynamic stacking energy.

Once a model structure exists, they train the model's parameters (production probabilities) using the Viterbi approximation of EM. This consists of iterating the following two-step procedure until the model parameters stabilize: First, they align each sequence to the model using the same alignment algorithm that we use, a Viterbi approximation to the inside-outside algorithm [LY90, Bak79]. Second, they apply the Viterbi approximation of EM to maximize a Bayesian posterior-probability estimate [ED94]. In this way, training consists of optimizing the alignment scores of the input training sequences. Eddy and Durbin use a variant of their alignment algorithm to perform database searches as well.

```
1. Start with an initial (possibly random) alignment denoted $A_0$. Set $A_{curr} = A_0$.
2. Use alignment $A_{curr}$ and the Nussinov/Zuker-based RNA folding algorithm to build
a covariance model. Estimate its parameters using the Viterbi approximation of EM.
Denote this initial model $C_0$. Set $C_{old} = \emptyset$ and $C_{new} = C_0$.
3. While $C_{new} \neq C_{old}$ do the following: {
  3a. Set $C_{old} = C_{new}$.
  3b. Use alignment $A_{curr}$ and the Nussinov/Zuker-based RNA folding algorithm
  to restructure the covariance model (choose new productions). Re-estimate the
  model's parameters using the iterative Viterbi approximation of EM, producing
  a new model $C_{new}$.
  3c. Use model $C_{new}$ and the Viterbi inside-outside aligning algorithm to produce
  a new multiple alignment $A_{new}$. Set $A_{curr} = A_{new}$.
}
```

Fig. 6. Pseudocode for the Eddy and Durbin training algorithm.

2 Discussion

In a recent paper [SBH$^+$93], we present our results in detail and compare these with the results of Sean Eddy and Richard Durbin's covariance models. Both methods have been used to produce models that perform three tasks: discriminate tRNA sequences from non-tRNA sequences, produce multiple alignments and ascertain the secondary structure of new sequences. The results show that all our grammars, except one trained on zero sequences, can perfectly discriminate nonmitochondrial tRNA sequences from non-tRNA sequences, and that our multiple alignments are nearly identical to the published tRNA alignments (full details are given elsewhere [SBH$^+$93]). Similarly, Eddy and Durbin's covariance models performed very well in database searches and produced multiple alignments nearly identical to the published. Both methods achieve local optima, rather than global, but the resulting models seem adequate for tRNA. Both methods use an inside-outside-based algorithm as part of the training process: we use one for parsing training trees, while they use one for producing a multiple alignment.

It appears that our basic grammar training algorithm, which is quite different from theirs, may be somewhat faster. Further, our custom-designed grammars and greater emphasis on learned, as opposed to constructed, Bayesian prior probability densities [BHK$^+$93a] may allow us to train with fewer training sequences. However, Eddy and Durbin have developed an exciting new technique to learn the structure of the grammar itself from unaligned training sequences, rather than just learn the probabilities of the productions and rely on prior information to specify the structure of the grammar (as we do). Also, they use a database searching algorithm while we do not.

The SCFG methods discussed in this paper represent a new direction in computational biosequence analysis. SCFGs provide a flexible and highly effective statistical method for solving a number of RNA sequence analysis problems including discrimination, multiple alignment and prediction of secondary structures. They may prove useful in maintaining, updating and revising existing multiple sequence alignments. In addition, a grammar itself may be a valuable tool for representing an RNA family or domain such as group I introns [MW90, MECS90], group II introns [MUO89], RNAse P RNA [BHJ$^+$91, TE93], small nuclear RNAs [GP88] and 7S RNA (signal recognition particle RNA) [Zwi89].

The main difficulties in applying this work to other families of RNA will be the development of appropriate initial grammars and the computational cost of parsing longer sequences. The latter problem can only be solved by the development of fundamentally different parsing methods, perhaps relying more on branch-and-bound methods [LS94] or heuristics. It is currently not clear which approach will be best. The former problem might be solved by the development of effective methods for learning the grammar itself from training sequences. The work of Eddy and Durbin is an important step in this direction [ED94]. Their method relies on correlations between columns in a multiple alignment [GPH$^+$92, Lap92, KB93, Wat89, WOW$^+$90, San85, Wat88] to discover the essential base-pairing structure in an RNA family. Another approach would be to use a method like that proposed by Waterman [Wat89] to find helices in a rough initial multiple alignment, use these helices to design a simple initial grammar in a semi-automated fashion using our high-level RNA grammar specification language, then use the grammar to obtain a better multiple alignment, and iterate this process until a suitable result is obtained. We are currently exploring this approach.

Another important direction for further research is the development of stochastic grammars for tRNA and other RNA families that can be used to search databases for these structures at the DNA level. In order to do this, the grammar must be modified to allow for the possibility of introns in the sequence, and the parsing method must be modified so that it can search efficiently for RNAs that are embedded within larger sequences. Durbin and Eddy have done the latter modifications in their tRNA experiments and report good results in searching the GenBank structural RNA database and 2.2 Mb of *C. elegans* genomic sequence for tRNAs, even without using special intron models. In our earlier work [SBM$^+$94], we reported some very preliminary results on modifying tRNA grammars to accommodate introns. We are currently planning to do further work in this direction. We see no insurmountable obstacles in developing effective stochastic grammar-based search methods, but predict that the main practical problem will be dealing with the long computation time required by the present methods.

Finally, there is the question of what further generalizations of hidden Markov models, beyond SCFGs, might be useful. The key advantage of our method over the HMM method is that it allows us to explicitly deal with the secondary structure of the RNA sequence. By extending stochastic models of strings to stochastic models of trees, we can model the base-pairing interactions of the molecule, which determine its secondary structure. This progression is similar to the path taken by the late King Sun Fu and colleagues in their development of the field of syntactic pattern recognition [Fu82]. Modeling pseudoknots and higher-order structure would require still more general methods. One possibility would be to consider *stochastic graph grammars* (see the introductory survey by Engelfriet and Rozenberg [ER91]) in hopes of obtaining a more general model of the interactions present in the molecule beyond the primary structure. If a stochastic graph grammar framework could be developed that included both an efficient method of finding the most probable folding of the molecule given the grammar and an efficient EM method for estimating the grammar's parameters from folded examples, then extensions of our approach to more challenging problems, including RNA tertiary structure determination and protein folding, would be possible. This is perhaps the most interesting direction for future research suggested by our results.

References

[AU72] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing.* Prentice Hall, Englewood Cliffs, N.J., 1972.

[Bak79] J. K. Baker. Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[BHJ+91] J. W. Brown, E. S. Haas, B. D. James, D. A. Hunt, J. S. Liu, and N. R. Pace. Phylogenetic analysis and evolution of RNase P RNA in proteobacteria. *Journal of Bacteriology*, 173:3855–3863, 1991.

[BHK+93a] M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Dirichlet mixture priors for HMMs. In preparation, 1993.

[BHK+93b] M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, pages 47–55, Menlo Park, CA, July 1993. AAAI/MIT Press.

[ED94] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. Submitted to *Nucleic Acids Research*, 1994.

[ER91] J. Engelfriet and G. Rozenberg. Graph grammars based on node rewriting: An introduction to NLC graph grammars. In E. Ehrig, H.J. Kreowski, and G. Rozenberg, editors, *Lecture Notes in Computer Science*, volume 532, pages 12–23. Springer-Verlag, 1991.

[Fu82] K. S. Fu. *Syntactic pattern recognition and applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[FW75] G. E. Fox and C. R Woese. 5S RNA secondary structure. *Nature*, 256:505–507, 1975.

[Gou87] M. Gouy. Secondary structure prediction of RNA. In M. J. Bishop and C. R. Rawlings, editors, *Nucleic acid and protein sequence analysis, a practical approach*, pages 259–284. IRL Press, Oxford, England, 1987.

[GP88] C. Guthrie and B. Patterson. Spliceosomal snRNAs. *Annual Review of Genetics*, 22:387–419, 1988.

[GPH+92] 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 Research*, 20:5785–5795, 1992.

[HKMS93] D. Haussler, A. Krogh, I. S. Mian, and K. Sjölander. Protein modeling using hidden Markov models: Analysis of globins. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 1, pages 792–802, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[KB93] T. Klinger and D. Brutlag. Detection of correlations in tRNA sequences with structural implications. In Lawrence Hunter, David Searls, and Jude Shavlik, editors, *First International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, 1993. AAAI Press.

[KBM+94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.

[Lap92] Allan Lapedes. Private communication, 1992.

[LOM+93] N. Larsen, G. J. Olsen, B. L. Maidak, M. J. McCaughey, R. Overbeek, T. J. Macke, T. L. Marsh, and C. R. Woese. The ribosomal database project. *Nucleic Acids Research*, 21:3021–3023, 1993.

[LS94] R. H. Lathrop and T. F. Smith. A branch-and-bound algorithm for optimal protein threading with pairwise (contact potential) amino acid interactions. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[LY90] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[MECS90] F. Michel, A. D. Ellington, S. Couture, and J. W. Szostak. Phylogenetic and genetic evidence for base-triples in the catalytic domain of group I introns. *Nature*, 347:578–580, 1990.

[MUO89] F. Michel, K. Umesono, and H. Ozeki. Comparative and functional anatomy of group II catalytic introns–a review. *Gene*, 82:5–30, 1989.

[MW90] F. Michel and E. Westhof. Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis. *Journal of Molecular Biology*, 216:585–610, 1990.

[NPGK78] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matchings. *SIAM Journal of Applied Mathematics*, 35:68–82, 1978.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77(2):257–286, 1989.

[Sae84] W. Saenger. *Principles of nucleic acid structure*. Springer Advanced Texts in Chemistry. Springer-Verlag, New York, 1984.

[Sak92] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.

[San85] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.

[SBH$^+$93] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. Underwood, and D. Haussler. The application of stochastic context-free grammars to folding, aligning and modeling homologous RNA sequences. Submitted for publication, 1993.

[SBM$^+$94] Y. Sakakibara, M. Brown, I. S. Mian, R. Underwood, and D. Haussler. Stochastic context-free grammars for modeling RNA. In *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[SBU$^+$93] Y. Sakakibara, M. Brown, R. Underwood, I. S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. Technical Report UCSC-CRL-93-16, UC Santa Cruz, Computer and Information Sciences Dept., Santa Cruz, CA 95064, 1993.

[SD89] T. J. Santner and D. E. Duffy. *The Statistical Analysis of Discrete Data*. Springer Verlag, New York, 1989.

[SD93] D. B. Searls and S. Dong. A syntactic pattern recognition system for DNA sequences. In *Proc. 2nd Int. Conf. on Bioinformatics, Supercomputing and complex genome analysis*. World Scientific, 1993. In press.

[Sea92] David B. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, November–December 1992.

[Sea93a] D. B. Searls. The computational linguistics of biological sequences. In *Artificial Intelligence and Molecular Biology*, chapter 2, pages 47–120. AAAI Press, 1993.

[Sea93b] D. B. Searls. String variable grammar: a logic grammar formalism for DNA sequences, 1993. Unpublished.

[SZ90] B. A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *CABIOS*, 6(4):309–318, 1990.

[TE93] A. J. Tranguch and D. R. Engelke. Comparative structural analysis of nuclear RNase P RNAs from yeast. *Journal of Biological Chemistry*, 268:14045–1455, 1993.

[TSF88] D. H. Turner, N. Sugimoto, and S. M. Freier. RNA structure prediction. *Annual Review of Biophysics and Biophysical Chemistry*, 17:167–192, 1988.

[TUL71] I. Tinoco Jr., O. C. Uhlenbeck, and M. D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:363–367, 1971.

[TW68] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2:57–81, 1968.

[Wat88] M. S. Waterman. Computer analysis of nucleic acid sequences. *Methods in Enzymology*, 164:765–792, 1988.

[Wat89] M. S. Waterman. Consensus methods for folding single-stranded nucleic acids. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, chapter 8. CRC Press, 1989.

[WGGN83] C. R. Woese, R. R. Gutell, R. Gupta, and H. F. Noller. Detailed analysis of the higher-order structure of 16S-like ribosomal ribonucleic acids. *Microbiology Reviews*, 47(4):621–669, 1983.

[WOW$^+$90] S. Winker, R. Overbeek, C.R. Woese, G.J. Olsen, and N. Pfluger. Structure detection through automated covariance search. *Computer Applications in the Biosciences*, 6:365–371, 1990.

[WPT89] J. R. Wyatt, J. D. Puglisi, and I. Tinoco Jr. RNA folding: pseudoknots, loops and bulges. *BioEssays*, 11(4):100–106, 1989.

[Zuk89] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.

[Zwi89] C. Zwieb. Structure and function of signal recognition particle RNA. *Progress in Nucleic Acid Research and Molecular Biology*, 37:207–234, 1989.