Stochastic Multiple Context-Free Grammar for RNA Pseudoknot Modeling

Yuki Kato
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
yuuki-ka@is.naist.jp

Hiroyuki Seki
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
seki@is.naist.jp

Tadao Kasami
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
kasami@naist.jp

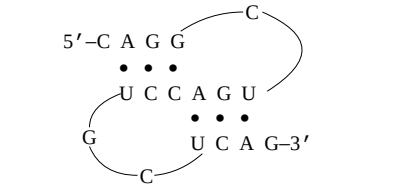
Abstract

Several grammars have been proposed for modeling RNA pseudoknotted structure. In this paper, we focus on multiple context-free grammars (MCFGs), which are natural extension of context-free grammars and can represent pseudoknots, and extend a specific subclass of MCFGs to a probabilistic model called SMCFG. We present a polynomial time parsing algorithm for finding the most probable derivation tree and a probability parameter estimation algorithm. Furthermore, we show some experimental results of pseudoknot prediction using SMCFG algorithm.

1 Introduction

Non-coding RNAs fold into characteristic structures determined by interactions between mostly Watson-Crick complementary base pairs. Such a base paired structure is called the *secondary structure*. *Pseudoknot* (Figure 1 (a)) is one of the typical substructures found in the secondary structures of several RNAs, including rRNAs, tmRNAs and viral RNAs. An alternative graphic representation of a pseudoknot is arc depiction where arcs connect base pairs (Figure 1 (b)). It has been recognized that pseudoknots play an important role in RNA functions such as ribosomal frameshifting and regulation of translation.

Many attempts have so far been made at modeling RNA secondary structure by formal grammars. In a grammatical approach, secondary structure prediction can be viewed as parsing problem. However, there may be many different derivation trees for an input sequence. Thus, it is necessary to have a method of extracting biologically realistic



(a) Pseudoknot

(b) Arc depiction of (a)

Figure 1: Example of RNA secondary structure

derivation trees among them. One solution to this problem is to extend a grammar to a probabilistic model and find the most likely derivation tree, and another is to take free energy minimization into account. Eddy and Durbin (1994), and Sakakibara et al. (1994) modeled RNA secondary structure without pseudoknots by using stochastic context-free grammars (stochastic CFGs or SCFGs). For pseudoknotted structure (Figure 1 (a)), however, another approach has to be taken since a single CFG cannot represent crossing dependencies of base pairs in pseudoknots (Figure 1 (b)) for the lack of generative power. Brown and Wilson (1996) proposed a model based on intersections of SCFGs to describe RNA pseudoknots. Cai et al. (2003) introduced a model based on parallel communication grammar systems using a single CFG synchronized with a number of regular grammars. Akutsu (2000) provided dynamic programming algorithms for RNA pseudoknot prediction without using grammars. On the other hand, several grammars have been proposed where the grammar itself can fully describe pseudoknots. Rivas and Eddy (1999, 2000) provided a dynamic programming

Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms, pages 57–64,
Sydney, July 2006.

algorithm for predicting RNA secondary structure including pseudoknots, and introduced a new class of grammars called RNA pseudoknot grammars (RPGs) for deriving sequences with gap. Uemura et al. (1999) defined specific subclasses of tree adjoining grammars (TAGs) named SL-TAGs and extended SL-TAGs (ESL-TAGs) respectively, and predicted RNA pseudoknots by using parsing algorithm of ESL-TAG. Matsui et al. (2005) proposed pair stochastic tree adjoining grammars (PSTAGs) based on ESL-TAGs and tree automata for aligning and predicting pseudoknots, which showed good prediction accuracy. These grammars have generative power stronger than CFGs and polynomial time algorithms for parsing problem.

In our previous work (Kato et al., 2005), we identified RPGs, SL-TAGs and ESL-TAGs as subclasses of *multiple context-free grammars* (MCFGs) (Kasami et al., 1988; Seki et al., 1991), which can model RNA pseudoknots, and showed a candidate subclass of the minimum grammars for representing pseudoknots. The generative power of MCFGs is stronger than that of CFGs and MCFGs have a polynomial time parsing algorithm like the CYK (Cocke-Younger-Kasami) algorithm for CFGs. In this paper, we extend the above candidate subclass of MCFGs to a probabilistic model called a stochastic MCFG (SMCFG). We present a polynomial time parsing algorithm for finding the most probable derivation tree, which is applicable to RNA pseudoknot prediction. In addition, we mention a probability parameter estimation method based on the EM (expectation maximization) algorithm. Finally, we show some experimental results on pseudoknot prediction for three RNA families using SMCFG algorithm, which show good prediction accuracy.

2 Stochastic Multiple Context-Free Grammar

A *stochastic multiple context-free grammar* (stochastic MCFG, or SMCFG) is a probabilistic extension of MCFG (Kasami et al., 1988; Seki et al., 1991) or *linear context-free rewriting system* (Vijay-Shanker et al., 1987). An SMCFG is a 5-tuple $G = (N, T, F, P, S)$ where N is a finite set of nonterminals, T is a finite set of terminals, F is a finite set of functions, P is a finite set of (production) rules and $S \in N$ is the start symbol. For each $A \in N$, a positive integer denoted by $\dim(A)$ is given and A derives $\dim(A)$-tuples of terminal sequences. For the start symbol S, $\dim(S) = 1$. For each $f \in F$, positive integers d_i $(0 \leq i \leq k)$ are given and f is a total function from $(T^*)^{d_1} \times \cdots \times (T^*)^{d_k}$ to $(T^*)^{d_0}$ where each component of f is defined as the concatenation of some components of arguments and constant sequences. Note that each component of an argument should occur in the function value at most once (linearity). For example, $f[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{21}, x_{12}x_{22})$. Each rule in P has the form of $A_0 \xrightarrow{p} f[A_1, \ldots, A_k]$ where $A_i \in N$ $(0 \leq i \leq k)$, $f : (T^*)^{\dim(A_1)} \times \cdots \times (T^*)^{\dim(A_k)} \to (T^*)^{\dim(A_0)} \in F$ and p is a real number with $0 \leq p \leq 1$ called the *probability* of this rule. The summation of the probabilities of the rules with the same left-hand side should be one. If we are not interested in p, we just write $A_0 \to f[A_1, \ldots, A_k]$. If $k \geq 1$, then the rule is called a *nonterminating rule*, and if $k = 0$, then it is called a *terminating rule*. A terminating rule $A_0 \to f[\,]$ with $f^{[h]}[\,] = \beta_h$ $(1 \leq h \leq \dim(A_0))$ is simply written as $A_0 \to (\beta_1, \ldots, \beta_{\dim(A_0)})$.

We recursively define the relation $\overset{*}{\Rightarrow}$ by the following (L1) and (L2): **(L1)** if $A \xrightarrow{p} \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, then we write $A \overset{*}{\Rightarrow} \overline{\alpha}$ with probability p, and **(L2)** if $A \xrightarrow{p} f[A_1, \ldots, A_k] \in P$ and $A_i \overset{*}{\Rightarrow} \overline{\alpha_i} \in (T^*)^{\dim(A_i)}$ $(1 \leq i \leq k)$ with probabilities $p_1, \ldots, p_k$, respectively, then we write $A \overset{*}{\Rightarrow} f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ with probability $p \cdot \prod_{i=1}^{k} p_i$. In parallel with the relation $\overset{*}{\Rightarrow}$, we define derivation trees as follows: **(D1)** if $A \xrightarrow{p} \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, then the ordered tree with the root labeled A which has $\overline{\alpha}$ as the only one child is a derivation tree for $\overline{\alpha}$ with probability p, and **(D2)** if $A \xrightarrow{p} f[A_1, \ldots, A_k] \in P$, $A_i \overset{*}{\Rightarrow} \overline{\alpha_i} \in (T^*)^{\dim(A_i)}$ $(1 \leq i \leq k)$ and $t_1, \ldots, t_k$ are derivation trees for $\overline{\alpha_1}, \ldots, \overline{\alpha_k}$ with probabilities $p_1, \ldots, p_k$, respectively, then the ordered tree with the root labeled A (or $A : f$ if necessary) which has $t_1, \ldots, t_k$ as (immediate) subtrees from left to right is a derivation tree for $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ with probability $p \cdot \prod_{i=1}^{k} p_i$. Example rules are $A \xrightarrow{0.3} f[A]$ where $f[(x_1, x_2)] = (ax_1b, cx_2d)$ and $A \xrightarrow{0.7} (ab, cd)$. Then, $A \overset{*}{\Rightarrow} (ab, cd)$ by the second rule, which is followed by $A \overset{*}{\Rightarrow} f[(ab, cd)] = (aabb, ccdd)$ by the first rule. The probability of the latter derivation is $0.3 \cdot 0.7 = 0.21$. The language generated by an SMCFG G is defined as $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow}$

Table 1: SMCFG G_s

Type	Rule set	Function	Transition probability	Emission probability
E	$W_v \to (\varepsilon, \varepsilon)$		1	1
S	$W_v \to J[W_y]$	$J[(x_1, x_2)] = x_1 x_2$	$t_v(y)$	1
D	$W_v \to SK[W_y]$	$SK[(x_1, x_2)] = (x_1, x_2)$	$t_v(y)$	1
B_1	$W_v \to C_1[W_y, W_z]$	$C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22})$	1	1
B_2	$W_v \to C_2[W_y, W_z]$	$C_2[x_1, (x_{21}, x_{22})] = (x_{21} x_1, x_{22})$	1	1
B_3	$W_v \to C_3[W_y, W_z]$	$C_3[x_1, (x_{21}, x_{22})] = (x_{21}, x_1 x_{22})$	1	1
B_4	$W_v \to C_4[W_y, W_z]$	$C_4[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1)$	1	1
U_{1L}	$W_v \to UP_{1L}^{a_i}[W_y]$	$UP_{1L}^{a_i}[(x_1, x_2)] = (a_i x_1, x_2)$	$t_v(y)$	$e_v(a_i)$
U_{1R}	$W_v \to UP_{1R}^{a_j}[W_y]$	$UP_{1R}^{a_j}[(x_1, x_2)] = (x_1 a_j, x_2)$	$t_v(y)$	$e_v(a_j)$
U_{2L}	$W_v \to UP_{2L}^{a_k}[W_y]$	$UP_{2L}^{a_k}[(x_1, x_2)] = (x_1, a_k x_2)$	$t_v(y)$	$e_v(a_k)$
U_{2R}	$W_v \to UP_{2R}^{a_l}[W_y]$	$UP_{2R}^{a_l}[(x_1, x_2)] = (x_1, x_2 a_l)$	$t_v(y)$	$e_v(a_l)$
P	$W_v \to BP^{a_i a_l}[W_y]$	$BP^{a_i a_l}[(x_1, x_2)] = (a_i x_1, x_2 a_l)$	$t_v(y)$	$e_v(a_i, a_l)$

w with probability greater than $0\}$.

In this paper, we focus on an SMCFG $G_s = (N, T, F, P, S)$ that satisfies the following conditions: G_s has m different nonterminals denoted by $W_1, \ldots, W_m$, each of which uses the only one type of a rule denoted by E, S, D, B_1, B_2, B_3, B_4, U_{1L}, U_{1R}, U_{2L}, U_{2R} or P [1] (see Table 1). The type of W_v is denoted by type(v) and we predefine type(1) = S, that is, W_1 is the start symbol. Consider a sample rule set $W_v \to UP_{1L}^{\alpha}[W_y] \mid UP_{1L}^{\alpha}[W_z]$ where $UP_{1L}^{\alpha}[(x_1, x_2)] = (\alpha x_1, x_2)$ and $\alpha \in T$. For each rule r, two real values called *transition probability* p_1 and *emission probability* p_2 are specified in Table 1. The probability of r is simply defined as $p_1 \cdot p_2$. In application, $p_1 = t_v(y)$ and $p_2 = e_v(a_i), \ldots$ in Table 1 are the parameters of the grammar, which are set by hand or by a training algorithm (Section 3.3) depending on the set of possible sequences to be analyzed.

3 Algorithms for SMCFG

In RNA structure analysis using stochastic grammars, we have to deal with the following three problems: **(1)** calculate the optimal alignment of a sequence to a stochastic grammar (alignment problem), **(2)** calculate the probability of a sequence given a stochastic grammar (scoring problem), and **(3)** estimate optimal probability parameters for a stochastic grammar given a set of example sequences (training problem). In this section, we give solutions to each problem for the specific SMCFG $G_s = (N, T, F, P, S)$.

3.1 Alignment Problem

The alignment problem for G_s is to find the most probable derivation tree for a given input sequence. This problem can be solved by a dynamic programming algorithm similar to the CYK algorithm for SCFGs (Durbin et al., 1998), and in this paper, we also call the parsing algorithm for G_s the CYK algorithm. We fix an input sequence $w = a_1 \cdots a_n$ ($|w| = n$). Let $\gamma_v(i, j)$ and $\gamma_y(i, j, k, l)$ be the logarithm of maximum probabilities of a derivation subtree rooted at a nonterminal W_v for a terminal subsequence $a_i \cdots a_j$ and of a derivation subtree rooted at a nonterminal W_y for a tuple of terminal subsequences $(a_i \cdots a_j, a_k \cdots a_l)$ respectively. The variables $\gamma_v(i, i-1)$ and $\gamma_y(i, i-1, j, j-1)$ are the logarithm of maximum probabilities for an empty sequence ε and a pair of ε. Let $\tau_v(i, j)$ and $\tau_y(i, j, k, l)$ be traceback variables for constructing a derivation tree, which are calculated together with $\gamma_v(i, j)$ and $\gamma_y(i, j, k, l)$. We define $\mathcal{C}_v = \{y \mid W_v \to f[W_y] \in P,\ f \in F\}$. To avoid non-emitting cycles, we assume that the nonterminals are numbered such that $v < y$ for all $y \in \mathcal{C}_v$. The CYK algorithm uses five dimensional dynamic programming matrix to calculate γ, which leads to $\log P(w, \hat{\pi} \mid \theta)$ where $\hat{\pi}$ is the most probable derivation tree and θ is an entire set of probability parameters. The detailed description of the CYK algorithm is as follows:

Algorithm 1 (CYK)**.**
Initialization:
for $i \leftarrow 1$ **to** $n+1$, $j \leftarrow i$ **to** $n+1$, $v \leftarrow 1$ **to** m
  do if type$(v) =$ E
    then $\gamma_v(i, i-1, j, j-1) \leftarrow 0$
  else $\gamma_v(i, i-1, j, j-1) \leftarrow -\infty$
Iteration:
for $i \leftarrow n$ **downto** 1, $j \leftarrow i-1$ **to** n, $k \leftarrow n+1$
downto $j+1$, $l \leftarrow k-1$ **to** n, $v \leftarrow 1$ **to** m
  do if type$(v) =$ E
    then if $j = i-1$ **and** $l = k-1$
      then skip

[1] These types stand for END, START, DELETE, BIFURCATION, UNPAIR and PAIR respectively.

else $\gamma_v(i,j,k,l) \leftarrow -\infty$
if type$(v) = \mathrm{S}$
then $\gamma_v(i,j)$
$\leftarrow \max_{y \in \mathcal{C}_v} \max_{h=i-1,\ldots,j} [\log t_v(y) + \gamma_y(i,h,h+1,j)]$
$\tau_v(i,j)$
$\leftarrow \arg\max_{(y,h)} [\log t_v(y) + \gamma_y(i,h,h+1,j)]$
if type$(v) = \mathrm{B}_1$ **and** $W_v \rightarrow C_1[W_y, W_z]$
then $\gamma_v(i,j,k,l)$
$\leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(i,h) + \gamma_z(h+1,j,k,l)]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{(y,z,h)} [\gamma_y(i,h) + \gamma_z(h+1,j,k,l)]$
if type$(v) = \mathrm{B}_2$ **and** $W_v \rightarrow C_2[W_y, W_z]$
then $\gamma_v(i,j,k,l)$
$\leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(h+1,j) + \gamma_z(i,h,k,l)]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{(y,z,h)} [\gamma_y(h+1,j) + \gamma_z(i,h,k,l)]$
if type$(v) = \mathrm{B}_3$ **and** $W_v \rightarrow C_3[W_y, W_z]$
then $\gamma_v(i,j,k,l)$
$\leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i,j,h+1,l) + \gamma_y(k,h)]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{(y,z,h)} [\gamma_z(i,j,h+1,l) + \gamma_y(k,h)]$
if type$(v) = \mathrm{B}_4$ **and** $W_v \rightarrow C_4[W_y, W_z]$
then $\gamma_v(i,j,k,l)$
$\leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i,j,k,h) + \gamma_y(h+1,l)]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{(y,z,h)} [\gamma_z(i,j,k,h) + \gamma_y(h+1,l)]$
if type$(v) = \mathrm{P}$
then if $j = i-1$ **or** $l = k-1$
then $\gamma_v(i,j,k,l) \leftarrow -\infty$
else $\gamma_v(i,j,k,l)$
$\leftarrow \max_{y \in \mathcal{C}_v} [\log e_v(a_i, a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{y} [\log e_v(a_i, a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$
else $\gamma_v(i,j,k,l)$
$\leftarrow \max_{y \in \mathcal{C}_v} [\log e_v(a_i, a_j, a_k, a_l) + \log t_v(y) + \gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$
$\tau_v(i,j,k,l)$
$\leftarrow \arg\max_{y} [\log e_v(a_i, a_j, a_k, a_l) + \log t_v(y) + \gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$

Note: $e_v(a_i, a_j, a_k, a_l) = e_v(a_i)$ for type$(v) = \mathrm{U}_{1\mathrm{L}}$, $e_v(a_i, a_j, a_k, a_l) = e_v(a_j)$ for type$(v) = \mathrm{U}_{1\mathrm{R}}$, $e_v(a_i, a_j, a_k, a_l) = e_v(a_k)$ for type$(v) = \mathrm{U}_{2\mathrm{L}}$, $e_v(a_i, a_j, a_k, a_l) = e_v(a_l)$ for type$(v) = \mathrm{U}_{2\mathrm{R}}$, $e_v(a_i, a_j, a_k, a_l) = 1$ for the other types except P. Also, $\Delta_v^{1L} = 1$ for type$(v) = \mathrm{U}_{1\mathrm{L}}$, $\Delta_v^{1R} = 1$ for type$(v) = \mathrm{U}_{1\mathrm{R}}$, $\Delta_v^{2L} = 1$ for type$(v) = \mathrm{U}_{2\mathrm{L}}$, $\Delta_v^{2R} = 1$ for type$(v) = \mathrm{U}_{2\mathrm{R}}$, and $\Delta_v^{1L}, \ldots \Delta_v^{2R}$ are set to 0 for the other types except P. □

When the calculation terminates, we obtain $\log P(w, \hat{\pi} \mid \theta) = \gamma_1(1,n)$. If there are b BIFURCATION nonterminals and a other nonterminals, the time and space complexities of the CYK algorithm are $O(amn^4 + bn^5)$ and $O(mn^4)$, respectively. To recover the optimal derivation tree, we use the traceback variables τ. Due to limitation of the space, the full description of the traceback algorithm is omitted (see (Kato and Seki, 2006)).

3.2 Scoring Problem

As in SCFGs (Durbin et al., 1998), the scoring problem for G_s can be solved by the inside algorithm. The inside algorithm calculates the summed probabilities $\alpha_v(i,j)$ and $\alpha_y(i,j,k,l)$ of all derivation subtrees rooted at a nonterminal W_v for a subsequence $a_i \cdots a_j$ and of all derivation subtrees rooted at a nonterminal W_y for a tuple of subsequences $(a_i \cdots a_j, a_k \cdots a_l)$ respectively. The variables $\alpha_v(i,i-1)$ and $\alpha_y(i,i-1,j,j-1)$ are defined for empty sequences in a similar way to the CYK algorithm. Therefore, we can easily obtain the inside algorithm by replacing max operations with summations in the CYK algorithm. When the calculation terminates, we obtain the probability $P(w \mid \theta) = \alpha_1(1,n)$. The time and space complexities of the algorithm are identical with those of the CYK algorithm.

In order to re-estimate the probability parameters of G_s, we need the outside algorithm. The outside algorithm calculates the summed probability $\beta_v(i,j)$ of all derivation trees excluding subtrees rooted at a nonterminal W_v generating a subsequence $a_i \cdots a_j$. Also, it calculates $\beta_y(i,j,k,l)$, the summed probability of all derivation trees excluding subtrees rooted at a nonterminal W_y generating a tuple of subsequences $(a_i \cdots a_j, a_k \cdots a_l)$. In the algorithm, we will use $\mathcal{P}_v = \{y \mid W_y \rightarrow f[W_v] \in P,\ f \in F\}$. Note that calculating the outside variables β requires the inside variables α. Unlike CYK and inside algorithms, the outside algorithm recursively works

its way inward. The time and space complexities of the outside algorithm are the same as those of CYK and inside algorithms. Formally, the outside algorithm is as follows:

Algorithm 2 (Outside)**.**
Initialization:
$\beta_1(1,n) \leftarrow 1$
Iteration:
for $i \leftarrow 1$ **to** $n+1$, $j \leftarrow n$ **downto** $i-1$, $k \leftarrow j+1$ **to** $n+1$, $l \leftarrow n$ **downto** $k-1$, $v \leftarrow 1$ **to** m
do if type$(v) = \mathrm{S}$ **and** $W_y \rightarrow C_1[W_v, W_z]$
then $\beta_v(i,j) \leftarrow \sum_{h=j}^{n} \sum_{k'=h+1}^{n+1} \sum_{l'=k'-1}^{n} \beta_y(i,h,k',l') \alpha_z(j+1,h,k',l')$
if type$(v) = \mathrm{S}$ **and** $W_y \rightarrow C_2[W_v, W_z]$
then $\beta_v(i,j) \leftarrow \sum_{h=1}^{i} \sum_{k'=j+1}^{n+1} \sum_{l'=k'-1}^{n} \beta_y(h,j,k',l') \alpha_z(h,i-1,k',l')$
if type$(v) = \mathrm{S}$ **and** $W_y \rightarrow C_3[W_v, W_z]$
then $\beta_v(i,j) \leftarrow \sum_{h=1}^{i} \sum_{k'=h-1}^{i-1} \sum_{l'=j}^{n} \beta_y(h,k',i,l') \alpha_z(h,k',j+1,l')$
if type$(v) = \mathrm{S}$ **and** $W_y \rightarrow C_4[W_v, W_z]$
then $\beta_v(i,j) \leftarrow \sum_{h=1}^{i} \sum_{k'=h-1}^{i-1} \sum_{l'=k'+1}^{i} \beta_y(h,k',l',j) \alpha_z(h,k',l',i-1)$
if type$(v) \neq \mathrm{S}$ **and** $W_y \rightarrow C_1[W_z, W_v]$
then $\beta_v(i,j,k,l) \leftarrow \sum_{h=1}^{i} \beta_y(h,j,k,l) \alpha_z(h,i-1)$
if type$(v) \neq \mathrm{S}$ **and** $W_y \rightarrow C_2[W_z, W_v]$
then $\beta_v(i,j,k,l) \leftarrow \sum_{h=j}^{k-1} \beta_y(i,h,k,l) \alpha_z(j+1,h)$
if type$(v) \neq \mathrm{S}$ **and** $W_y \rightarrow C_3[W_z, W_v]$
then $\beta_v(i,j,k,l) \leftarrow \sum_{h=j+1}^{k} \beta_y(i,j,h,l) \alpha_z(h,k-1)$
if type$(v) \neq \mathrm{S}$ **and** $W_y \rightarrow C_4[W_z, W_v]$
then $\beta_v(i,j,k,l) \leftarrow \sum_{h=l}^{n} \beta_y(i,j,k,h) \alpha_z(l+1,h)$
else $\beta_v(i,j,k,l) \leftarrow \sum_{y \in \mathcal{P}_v} \beta_y(i-\Delta_y^{1L}, j+\Delta_y^{1R}, k-\Delta_y^{2L}, l+\Delta_y^{2R}) e_y(a_{i-\Delta_y^{1L}}, a_{j+\Delta_y^{1R}}, a_{k-\Delta_y^{2L}}, a_{l+\Delta_y^{2R}}) t_y(v)$ □

3.3 Training Problem

The training problem for G_s can be solved by the EM algorithm called the inside-outside algorithm where the inside variables α and outside variables β are used to re-estimate probability parameters.

First, we consider the probability that a nonterminal W_v is used at positions i, j, k and l in a derivation of a single sequence w. If type$(v) = \mathrm{S}$, the probability is $\frac{1}{P(w|\theta)} \alpha_v(i,j)\beta_v(i,j)$, otherwise $\frac{1}{P(w|\theta)} \alpha_v(i,j,k,l)\beta_v(i,j,k,l)$. By summing these over all positions in the sequence, we can obtain the expected number of times that W_v is used for w as follows: for type$(v) = \mathrm{S}$, the expected count is

$$\frac{1}{P(w \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \alpha_v(i,j)\beta_v(i,j),$$

otherwise

$$\frac{1}{P(w \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \alpha_v(i,j,k,l) \beta_v(i,j,k,l).$$

Next, we extend these expected values from a single sequence w to multiple independent sequences $w^{(r)}$ $(1 \leq r \leq N)$. Let $\alpha^{(r)}$ and $\beta^{(r)}$ be the inside and outside variables calculated for each input sequence $w^{(r)}$. Then we can obtain the expected number of times that a nonterminal W_v is used for training sequences $w^{(r)}$ $(1 \leq r \leq N)$ by summing the above terms over all sequences: for type$(v) = \mathrm{S}$,

$$E(v) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \alpha_v^{(r)}(i,j) \beta_v^{(r)}(i,j),$$

otherwise

$$E(v) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \alpha_v^{(r)}(i,j,k,l) \beta_v^{(r)}(i,j,k,l).$$

Similarly, for a given W_y, the expected number of times that a rule $W_v \rightarrow f[W_y]$ is applied can be

obtained as follows: for type$(v) = \mathrm{S}$,

$$E(v \to y) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{h=i-1}^{j} \frac{1}{P(w^{(r)} \mid \theta)} \beta_v^{(r)}(i,j) t_v(y) \alpha_y^{(r)}(i,h,h+1,j),$$

otherwise

$$E(v \to y) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \beta_v^{(r)}(i,j,k,l) e_v(a_i, a_j, a_k, a_l) t_v(y) \alpha_y^{(r)}(i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R}).$$

For a given terminal a or a pair of terminals (a,b), the expected number of times that a rule containing a (or a and b) is applied is as shown below: for type$(v) = \mathrm{U_{1L}}$,

$$E(v \to a) = \sum_{r=1}^{N} \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \delta(a_i^{(r)} = a) \beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l),$$

for type$(v) = \mathrm{U_{1R}}$,

$$E(v \to a) = \sum_{r=1}^{N} \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \delta(a_j^{(r)} = a) \beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l),$$

for type$(v) = \mathrm{U_{2L}}$,

$$E(v \to a) = \sum_{r=1}^{N} \sum_{i=1}^{n-1} \sum_{j=i-1}^{n-1} \sum_{k=j+1}^{n} \sum_{l=k}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \delta(a_k^{(r)} = a) \beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l),$$

for type$(v) = \mathrm{U_{2R}}$,

$$E(v \to a) = \sum_{r=1}^{N} \sum_{i=1}^{n-1} \sum_{j=i-1}^{n-1} \sum_{k=j+1}^{n} \sum_{l=k}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \delta(a_l^{(r)} = a) \beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l),$$

and for type$(v) = \mathrm{P}$,

$$E(v \to ab) = \sum_{r=1}^{N} \sum_{i=1}^{n-1} \sum_{j=i}^{n-1} \sum_{k=j+1}^{n} \sum_{l=k}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \delta(a_i^{(r)} = a, a_l^{(r)} = b) \beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l),$$

where $\delta(C)$ is 1 if the condition C in the parenthesis is ture, and 0 if C is false.

Now, we re-estimate probability parameters by using the above expected counts. Let $\hat{t}_v(y)$ be re-estimated transition probabilities from W_v to W_y. Also, let $\hat{e}_v(a)$ and $\hat{e}_v(a,b)$ be re-estimated emission probabilities that W_v emits a symbol a and two symbols a and b respectively. We can obtain each re-estimated probability by the following equations:

$$\hat{t}_v(y) = \frac{E(v \to y)}{E(v)}, \ \hat{e}_v(a) = \frac{E(v \to a)}{E(v)}, \ \hat{e}_v(a,b) = \frac{E(v \to ab)}{E(v)}. \tag{3.1}$$

Note that the expected count correctly corresponding to its nonterminal type must be substituted for the above equations. In summary, the inside-outside algorithm is as follows:

Algorithm 3 (Inside-Outside)**.**
Initialization: Pick arbitrary probability parameters of the model.

Iteration: Calculate the new probability parameters using (3.1). Calculate the new log likelihood $\sum_{r=1}^{N} \log P(w^{(r)} \mid \theta)$ of the model.

Termination: Stop if the change in log likelihood is less than predefined threshold. □

4 Experimental Results

4.1 Data for Experiments

The dataset for experiments was taken from an RNA family database called "Rfam" (version 7.0) (Griffiths-Jones et al., 2003) which is a database of multiple sequence alignment and covariance models (Eddy and Durbin, 1994) representing non-coding RNA families. We selected three viral RNA families with pseudoknot annotations named Corona_pk_3 (Corona), HDV_ribozyme (HDV) and Tombus_3_IV (Tombus) (see Table 2). Corona_pk_3 has a simple pseudoknotted structure, whereas HDV_ribozyme and Tombus_3_IV have more complicated structures with pseudoknot.

Table 2: Three RNA families from Rfam ver. 7.0

Family	Range of length	# of annotated sequences	# of test sequences
Corona_pk_3	62–64	14	10
HDV_ribozyme	87–91	15	10
Tombus_3_IV	89–92	18	12

Table 3: Prediction results

Family	Precision [%]			Recall [%]			CPU time [sec]		
	Average	Min	Max	Average	Min	Max	Average	Min	Max
Corona_pk_3	99.4	94.4	100.0	99.4	94.4	100.0	27.8	26.0	30.4
HDV_ribozyme	100.0	100.0	100.0	100.0	100.0	100.0	252.1	219.0	278.4
Tombus_3_IV	100.0	100.0	100.0	100.0	100.0	100.0	244.8	215.2	257.5

4.2 Implementation

We specified a particular SMCFG G_s by utilizing secondary structure annotation of each family. Rules were determined by considering consensus secondary structure. Probability parameters were estimated in a few selected sequences by the simplest pseudocounting method known as the Laplace's rule (Durbin et al., 1998): to add one extra count to the true counts for each base configuration observed in a few selected sequences. Note that the inside-outside algorithm was not used in the experiments. The other sequences in the alignment were used as the test sequences for prediction (see Table 2). We implemented the CYK algorithm with traceback in ANSI C on a machine with Intel Pentium D CPU 2.80 GHz and 2.00 GB RAM. Straightforward implementation gives rise to a serious problem of lack of memory space due to the higher order dynamic programming matrix (remember that the space complexity of the CYK algorithm is $O(mn^4)$). The dynamic programming matrix in our specified model is sparse, and therefore, we successfully implemented the matrix as a hash table storing only nonzero probability values (equivalently, finite values of the logarithm of probabilities).

4.3 Tests

We tested prediction accuracy by calculating precision and recall (sensitivity), which are the ratio of the number of correct base pairs predicted by the algorithm to the total number of predicted base pairs, and the ratio of the number of correct base pairs predicted by the algorithm to the total number of base pairs specified by the trusted annotation, respectively. The results are shown in Table 3. A nearly correct prediction (94.4% precision and recall) for Corona_pk_3 is shown in Figure 2 where underlined base pairs agree with trusted ones. The secondary structures predicted by our algorithm agree very well with the trusted structures.

4.4 Comparison with PSTAG

We compared the prediction accuracy of our SMCFG algorithm with that of PSTAG algorithm (Matsui et al., 2005) (see Table 4). PSTAGs, as we have mentioned before, are proposed for modeling pairwise alignment of RNA sequences with pseudoknots and assign a probability to each alignment of TAG derivation trees. PSTAG algorithm, based on dynamic programming, calculates the most likely alignment for the pair of TAG derivation trees where one of them is in the form of an unfolded sequence and the other is a TAG derivation tree for known structure. SMCFG method shows better performance in accuracy than PSTAG method in the same test sets.

5 Conclusion

In this paper, we have proposed a probabilistic model named SMCFG, and designed a polynomial time parsing and a parameter estimation algorithm for SMCFG. Moreover, we have demonstrated computational experiments of RNA secondary structure prediction with pseudoknots using SMCFG parsing algorithm, which show good performance in accuracy.

Acknowledgments

This work is supported in part by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science (JSPS). We also wish to thank JSPS Research Fellowships for Young Scientists for their generous financial assistance. The authors thank Dr. Yoshiaki Takata for his useful

Corona_pk3 (EMBL accession #: X51325.1)

[Trusted structure in Rfam]

```
CUAGUCUUAUACACAAUGGUAAGCCAGUGGUAGUAAAGGUAUAAGAAAUUUGCUACUAUGUUA
    [[[[[[[[                (((  (((((((  ]]]]]]]]   )))))))  )))
```

[Prediction by SMCFG]

```
CUAGUCUUAUACACAAUGGUAAGCCAGUGGUAGUAAAGGUAUAAGAAAUUUGCUACUAUGUUA
    [[[[[[[[                ((((((((((   ]]]]]]]]    ((((((((((
```

Figure 2: Comparison of a prediction result with a trusted structure in Rfam

Table 4: Comparison between SMCFG and PSTAG

Model	Average precision [%]			Average recall [%]		
	Corona	HDV	Tombus	Corona	HDV	Tombus
SMCFG	99.4	100.0	100.0	99.4	100.0	100.0
PSTAG	95.5	95.6	97.4	94.6	94.1	97.4

comments on implementation of high dimensional dynamic programming.

References

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

Michael Brown and Charles Wilson. 1996. RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. *Proc. Pacific Symposium on Biocomputing*, 109–125.

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

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis*, Cambridge University Press.

Sean R. Eddy and Richard Durbin. 1994. RNA sequence analysis using covariance models. *Nuc. Acids Res.*, 22(11):2079–2088.

Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, Ajay Khanna, and Sean R. Eddy. 2003. Rfam: an RNA family database. *Nuc. Acids Res.*, 31(1):439–441.

Tadao Kasami, Hiroyuki Seki, and Mamoru Fujii. 1988. Generalized context-free grammar and multiple context-free grammar. *IEICE Trans. Inf. & Syst.*, J71-D(5):758–765 (in Japanese).

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2005. On the generative power of grammars for RNA secondary structure. *IEICE Trans. Inf. & Syst.*, E88-D(1):53–64.

Yuki Kato and Hiroyuki Seki. 2006. Stochastic multiple context-free grammar for RNA pseudoknot modeling. *NAIST Info. Sci. Tech. Rep. (NAIST-IS-TR2006002)*.

Hiroshi Matsui, Kengo Sato, and Yasubumi Sakakibara. 2005. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics*, 21(11):2611–2617.

Elena Rivas and Sean R. Eddy. 1999. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068.

Elena Rivas and Sean R. Eddy. 2000. The language of RNA: A formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340.

Yasubumi Sakakibara, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjölander, Rebecca C. Underwood, and David Haussler. 1994. Stochastic context-free grammars for tRNA modeling. *Nuc. Acids Res.*, 22:5112–5120.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theor. Comput. Sci.*, 88:191–229.

Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. 1999. Tree adjoining grammars for RNA structure prediction. *Theor. Comput. Sci.*, 210:277–303.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. *Proc. 25th Annual Meeting of Association for Computational Linguistics (ACL87)*, 104–111.