社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－BIO－3（2）
2005／12／22

RNA 構造解析のための確率多重文脈自由文法

加藤 有己　　関 浩之
奈良先端科学技術大学院大学 情報科学研究科

シュードノットと呼ばれる部分構造を含む，RNA の２次構造をモデル化する形式文法がいくつか提案されている．本論文では，文脈自由文法の自然な拡張でありシュードノットを表現できる多重文脈自由文法（MCFG）に着目し，その部分クラスを確率モデルに拡張する．次に，RNA ２次構造予測に応用される，最大確率を持つ導出木を求める多項式時間のアルゴリズムを与える．また，EM アルゴリズムに基づく確率パラメータ推定法にも触れる．

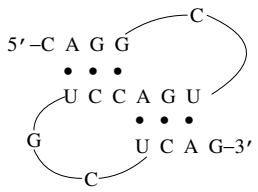
Stochastic Multiple Context-Free Grammar for RNA Structure Analysis

Yuki Kato and Hiroyuki Seki
Graduate School of Information Science, Nara Institute of Science and Technology

Several formal grammars have been proposed for modeling RNA secondary structure including substructure called pseudoknot. 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. We present a polynomial time parsing algorithm for finding the most probable derivation tree, which is applicable to RNA secondary structure prediction. In addition, we mention a probability parameter estimation method based on the EM (expectation maximization) algorithm.

1 Introduction

Functional 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.



(a) Pseudoknot

(b) Arc depiction of (a)

Figure 1: Example of RNA secondary structure

Many attempts have so far been made at modeling RNA secondary structure by formal grammars in order to analyze RNA structure. In a grammatical approach, an application to secondary structure prediction can be made by designing and implementing a parsing algorithm for the grammar. Eddy and Durbin [4], and Sakakibara et al. [11] modeled RNA secondary structure without pseudoknots by using stochastic context-free grammars (stochastic CFGs, SCFGs). For pseudoknotted structure, however, another approach has to be taken since a single CFG cannot represent crossing dependencies of base pairs in pseudoknots (Figure 1 (b)) for lack of generative power. Brown and Wilson [1] proposed a model based on intersections of SCFGs to model RNA pseudoknots. Cai et al. [2] introduced a model based on parallel communication grammar systems using a single CFG synchronized with a number of regular grammars to represent pseudoknots. On the other hand,

several grammars have been proposed where the grammar itself can fully describe pseudoknotted structure. Uemura et al. [13] defined specific subclasses of tree adjoining grammars and specified pseudoknots. Rivas and Eddy [9, 10] provided a dynamic programming algorithm for predicting RNA secondary structure including pseudoknots and introduced a new class of grammars for deriving sequences with gap. These grammars have generative power stronger than CFGs and polynomial time algorithms for parsing problem. However, probabilistic models of these grammars have not been developed.

In our previous work [7], we identified some of the grammars mentioned above as subclasses of *multiple context-free grammars* (MCFGs) [5, 6, 12], 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 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 secondary structure prediction including pseudoknots. In addition, we mention a probability parameter estimation method based on the EM (expectation maximization) algorithm.

2 Multiple Context-Free Grammar

For an alphabet Σ, let Σ^* denote the set of all finite sequences over Σ. The empty sequence is denoted by ε. For a sequence $w \in \Sigma^*$, let $|w|$ denote the length of w, that is, the number of symbols occurring in w.

A *multiple context-free grammar* (MCFG) [5, 6, 12] is a 5-tuple $G = (N, T, F, P, S)$ where N is a finite set of nonterminals, T a finite set of terminals, F a finite set of functions, P a finite set of (production) rules and $S \in N$ 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 $(T^*)^{d_0}$ satisfying the following condition (F):

(F) Let $\overline{x_i} = (x_{i1}, \ldots, x_{id_i})$ denote the ith argument of f for $1 \leq i \leq k$. The hth component of the function value for $1 \leq h \leq d_0$, denoted by $f^{[h]}$, is defined as

$$f^{[h]}[\overline{x_1}, \ldots, \overline{x_k}] = \beta_{h0} z_{h1} \beta_{h1} z_{h2} \cdots z_{hv_h} \beta_{hv_h} \qquad (*)$$

where $\beta_{hl} \in T^*$ $(0 \leq l \leq v_h)$ and $z_{hl} \in \{x_{ij} \mid 1 \leq i \leq k,\ 1 \leq j \leq d_i\}$ $(1 \leq l \leq v_h)$. The total number of occurrences of x_{ij} in the right hand sides of $(*)$ from $h = 1$ through d_0 is at most one.

Each rule in P has the form of $A_0 \to f[A_1, \ldots, A_k]$ where $A_i \in N$ $(0 \leq i \leq k)$ and $f : (T^*)^{\dim(A_1)} \times \cdots \times (T^*)^{\dim(A_k)} \to (T^*)^{\dim(A_0)} \in F$. 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 $A_0 \to (\beta_1, \ldots, \beta_{\dim(A_0)})$.

Example 1. (1) Let $G_1 = (N_1, T_1, F_1, P_1, S)$ be an MCFG where $N_1 = \{S, A\},\ T_1 = \{a, b\}$ and $P_1 = \{S \to J[A],\ A \to f_a[A] \mid f_b[A] \mid (\varepsilon, \varepsilon)\}$ where $\dim(S) = 1,\ \dim(A) = 2,\ J[(x_1, x_2)] = x_1 x_2$ and $f_\alpha[(x_1, x_2)] = (\alpha x_1, \alpha x_2)$ with $\alpha = a, b$.

(2) Let $G_2 = (N_2, T_2, F_2, P_2, S)$ be an MCFG where $N_2 = \{S, A\},\ T_2 = \{a_i \mid 1 \leq i \leq 2m\}$ and $P_2 = \{S \to J_m[A],\ A \to g[A] \mid (\varepsilon, \ldots, \varepsilon)\}$ where $\dim(S) = 1,\ \dim(A) = m,\ J_m[(x_1, \ldots, x_m)] = x_1 \cdots x_m$ and $g[(x_1, \ldots, x_m)] = (a_1 x_1 a_2, \ldots, a_{2m-1} x_m a_{2m})$. □

For a function f defined by $(*)$ in condition (F) and tuples of terminal sequences $\overline{\alpha_i} = (\alpha_{i1}, \ldots, \alpha_{id_i}) \in (T^*)^{d_i}$ $(1 \leq i \leq k)$, let $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ denote the tuple of terminal sequences obtained from the right hand sides of $(*)$ by substituting α_{ij} $(1 \leq i \leq k,\ 1 \leq j \leq d_i)$ into x_{ij}. For instance, $f_a[(bba, ab)] = (abba, aab)$ in Example 1 (1). We recursively define the relation $\overset{*}{\Rightarrow}$ by the following (L1) and (L2):

(L1) If $A \to \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, then we write $A \overset{*}{\Rightarrow} \overline{\alpha}$.

(L2) If $A \to f[A_1, \ldots, A_k] \in P$ and $A_i \overset{*}{\Rightarrow} \overline{\alpha_i}$ $(1 \leq i \leq k)$, then we write $A \overset{*}{\Rightarrow} f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$.

The language generated by an MCFG G is defined as $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$. In parallel with the relation $\overset{*}{\Rightarrow}$, we recursively define derivation trees as follows:

(D1) If $A \to \overline{\alpha} \in P$ ($\overline{\alpha} \in (T^*)^{\dim(A)}$), then a derivation tree for $\overline{\alpha}$ is the tree with the root labeled A which has $\overline{\alpha}$ as the only one child.

(D2) If $A \to f[A_1, \ldots, A_k] \in P$, $A_i \overset{*}{\Rightarrow} \overline{\alpha_i}$ $(1 \leq i \leq k)$ and $t_1, \ldots, t_k$ are derivation trees for $\overline{\alpha_1}, \ldots, \overline{\alpha_k}$, then a derivation tree for $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ is the 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.

Example 1 (continued)**.** (1) By (L1), $A \overset{*}{\Rightarrow} (\varepsilon, \varepsilon)$ since $A \to (\varepsilon, \varepsilon) \in P$. Since $f_a[(\varepsilon, \varepsilon)] = (a, a)$ and $f_b[(a, a)] = (ba, ba)$, we have $A \overset{*}{\Rightarrow} (a, a)$ and $A \overset{*}{\Rightarrow} (ba, ba)$ by (L2). Also by $S \to J[A]$, $S \overset{*}{\Rightarrow} J[(ba, ba)] = baba$. In fact, $L(G_1) = \{ww \mid w \in \{a, b\}^*\}$.

(2) Likewise, $A \overset{*}{\Rightarrow} (\varepsilon, \ldots, \varepsilon)$ by (L1), $A \overset{*}{\Rightarrow} f[(\varepsilon, \ldots, \varepsilon)] = (a_1 a_2, \ldots, a_{2m-1} a_{2m})$ by (L2), etc. This tells us that $L(G_2) = \{a_1^n \cdots a_{2m}^n \mid n \geq 0\}$. □

To introduce subclasses of MCFGs, we define a few terminologies. Let $G = (N, T, F, P, S)$ be an arbitrary MCFG. For a function $f : (T^*)^{d_1} \times \cdots \times (T^*)^{d_k} \to (T^*)^{d_0}$, let $\dim(f) = \max\{d_j \mid 0 \leq j \leq k\}$, $\mathrm{rank}(f) = k$ and $\deg(f) = \Sigma_{j=0}^{k} d_j$, which are called the *dimension* of f, the *rank* of f and the *degree* of f respectively. Finally, let us define the dimension, rank and degree of G as $\dim(G) = \max\{\dim(f) \mid f \in F\}$, $\mathrm{rank}(G) = \max\{\mathrm{rank}(f) \mid f \in F\}$ and $\deg(G) = \max\{\deg(f) \mid f \in F\}$ respectively. By definition, $\deg(G) \leq \dim(G)(\mathrm{rank}(G) + 1)$. With these parameters, we define subclasses of MCFGs. An MCFG G with $\dim(G) \leq m$ and $\mathrm{rank}(G) \leq r$ is called an (m, r)-MCFG. Likewise, an MCFG G with $\dim(G) \leq m$ is called an m-MCFG.

In our previous work [7], we have shown that a candidate subclass of the minimum grammars which can represent repeating pseudoknots is the class of $(2, 2)$-MCFGs with degree five or less. In the following, we will focus on a $(2, 2)$-MCFG with degree five or less and extend it to a probabilistic model in the next section.

Example 2. Consider a $(2, 2)$-MCFG $G_3 = (\{S, A, X_1, X_2\}, \{a, c, g, u\}, F_3, P_3, S)$ with $\deg(G_3) \leq 5$ for generating RNA sequences where P_3 and F_3 are as follows:

$S \to J[A]$,	$J[(x_1, x_2)] = x_1 x_2$,
$A \to UP_{1L}^{\alpha}[A]$,	$UP_{1L}^{\alpha}[(x_1, x_2)] = (\alpha x_1, x_2)$,
$X_1 \to UP_{1L}^{\alpha}[A]$,	
$A \to UP_{1R}^{\alpha}[A]$,	$UP_{1R}^{\alpha}[(x_1, x_2)] = (x_1 \alpha, x_2)$,
$A \to UP_{1R}^{\alpha}[X_1]$,	
$A \to UP_{2L}^{\alpha}[A]$,	$UP_{2L}^{\alpha}[(x_1, x_2)] = (x_1, \alpha x_2)$,
$X_2 \to UP_{2L}^{\alpha}[A]$,	
$A \to UP_{2R}^{\alpha}[A]$,	$UP_{2R}^{\alpha}[(x_1, x_2)] = (x_1, x_2 \alpha)$,
$A \to UP_{2R}^{\alpha}[X_2]$,	
$(\alpha \in \{a, c, g, u\})$	
$A \to BP^{\alpha\beta}[A]$,	$BP^{\alpha\beta}[(x_1, x_2)] = (\alpha x_1, x_2 \beta)$,
$((\alpha, \beta) \in \{(a, u), (u, a), (c, g), (g, c)\})$	
$A \to C_1[S, A]$,	$C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22})$,
$A \to C_2[S, A]$,	$C_2[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1)$,
$A \to (\varepsilon, \varepsilon)$.	

Functions have mnemonic names where UP, BP and C stand for unpair, base pair and concatenation respectively. The RNA sequence `agacuu` in Figure 2 can be generated by the above rules as follows: $A \overset{*}{\Rightarrow} BP^{gc}[(\varepsilon, \varepsilon)] = (g, c)$, $A \overset{*}{\Rightarrow} BP^{au}[(g, c)] = (ag, cu)$, $X_2 \overset{*}{\Rightarrow} UP_{2L}^{a}[(ag, cu)] = (ag, acu)$, $A \overset{*}{\Rightarrow} UP_{2R}^{u}[(ag, acu)] = (ag, acuu)$ and $S \overset{*}{\Rightarrow} J[(ag, acuu)] = agacuu$. G_3 has a derivation tree (Figure 3) for `agacuu` which represents the pseudoknot shown in Figure 2. □

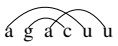


Figure 2: Example of a pseudoknot

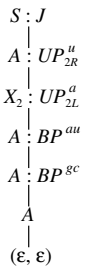


Figure 3: A derivation tree in G_3

3 Stochastic Multiple Context-Free Grammar

First, we briefly review stochastic (probabilistic) grammars. In a stochastic grammar, each rule has an associated probability such that the sum of the probabilities of rules with the same left hand side is 1. Sample rules of an SCFG are $S \overset{0.4}{\rightarrow} SS$ and $S \overset{0.6}{\rightarrow} ab$ where S is a nonterminal, a and b are terminals and the number over an arrow is a probability that the rule is applied. A stochastic grammar with probability parameters θ generates different sequences w with probabilities $P(w \mid \theta)$ [1] and defines the probability distribution over sequences, that is, $\sum_w P(w \mid \theta) = 1$. Note that a non-stochastic grammar either generates a sequence w or does not. In the following, we will regard nonterminals as states and applying rules as state transitions for convenience. If the right hand side of a rule contains one or more terminals, the nonterminal in the left hand side can be interpreted to emit symbols.

Next, we extend an MCFG to a probabilistic model, which we call a *stochastic multiple context-free grammar* (stochastic MCFG, SMCFG). Let G_s be a $(2,2)$-MCFG with $\deg(G_s) \leq 5$. G_s has m different nonterminals denoted by $W_1, \ldots, W_m$, and the rules and functions are shown in Table 1. Each nonterminal W_v $(1 \leq v \leq m)$ has one of the nine types of the rules which are S, U_{1L}, U_{1R}, U_{2L}, U_{2R}, P, B_1, B_2 and E for indicating start, unpair, pairwise, bifurcation and end states respectively. The type of W_v is denoted by type(v) and we predefine type$(1) = S$, that is, W_1 is the start symbol. We assign a symbol emission probability and state transition probability to each rule as shown in Table 1. Notice that in the table, a_i, a_j, a_k and a_l are the ith, jth, kth and lth terminals in a sequence respectively. For each nonterminal W_v, Δ_v^{1L}, Δ_v^{1R}, Δ_v^{2L} and Δ_v^{2R} are defined as the number of symbols emitted by W_v (see Table 2). Let $\mathcal{C}_v$ be the set of the children of a node W_v in a derivation tree, which is represented by index y such that W_v can make a transition to W_y. Let $\mathcal{P}_v$ be the set of the parents of W_v in a derivation tree, which is represented by index y such that W_y can make a transition to W_v. We will use these notations for simple description of the algorithms in the next section. It is easy to see that G_s can generate RNA sequences corresponding to pseudoknots (see Example 2).

Table 1: SMCFG G_s

Type	Rule	Function	Emission	Transition
S	$W_v \rightarrow J[W_y]$	$J[(x_1, x_2)] = x_1 x_2$	1	$t_v(y)$
U_{1L}	$W_v \rightarrow UP_{1L}[W_y]$	$UP_{1L}[(x_1, x_2)] = (a_i x_1, x_2)$	$e_v(a_i)$	$t_v(y)$
U_{1R}	$W_v \rightarrow UP_{1R}[W_y]$	$UP_{1R}[(x_1, x_2)] = (x_1 a_j, x_2)$	$e_v(a_j)$	$t_v(y)$
U_{2L}	$W_v \rightarrow UP_{2L}[W_y]$	$UP_{2L}[(x_1, x_2)] = (x_1, a_k x_2)$	$e_v(a_k)$	$t_v(y)$
U_{2R}	$W_v \rightarrow UP_{2R}[W_y]$	$UP_{2R}[(x_1, x_2)] = (x_1, x_2 a_l)$	$e_v(a_l)$	$t_v(y)$
P	$W_v \rightarrow BP[W_y]$	$BP[(x_1, x_2)] = (a_i x_1, x_2 a_l)$	$e_v(a_i, a_l)$	$t_v(y)$
B_1	$W_v \rightarrow C_1[W_y, W_z]$	$C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22})$	1	$t_v(y, z)$
B_2	$W_v \rightarrow C_2[W_y, W_z]$	$C_2[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1)$	1	$t_v(y, z)$
E	$W_v \rightarrow (\varepsilon, \varepsilon)$		1	1

[1] $P(w \mid \theta)$ represents conditional probability, that is, the probability of w when θ is given.

Table 2: The number of symbols emitted by nonterminals

Type	Δ_v^{1L}	Δ_v^{1R}	Δ_v^{2L}	Δ_v^{2R}
S	0	0	0	0
U_{1L}	1	0	0	0
U_{1R}	0	1	0	0
U_{2L}	0	0	1	0
U_{2R}	0	0	0	1
P	1	0	0	1
B_1	0	0	0	0
B_2	0	0	0	0
E	0	0	0	0

4 Algorithms for SMCFG

In RNA structure analysis using stochastic grammars, we must deal with the following three problems:

(1) Calculate the optimal alignment of a sequence to a parameterized stochastic grammar. (alignment problem)

(2) Calculate the probability of a sequence given a parameterized stochastic grammar. (scoring problem)

(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 introduced in the previous section.

4.1 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 [3], and in this paper, we also call the parsing algorithm for G_s the CYK algorithm. We assume that an input sequence $w = a_1 \cdots a_n$ (i.e., $|w| = n$). In fact, w is an RNA sequence composed of four symbols a, c, g and u. 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 $\gamma_y(i,j,k,l)$. The CYK algorithm uses three and five dimensional dynamic programming matrices to calculate $\gamma_v(i,j)$ and $\gamma_y(i,j,k,l)$ respectively, which leads to $\log P(w, \hat{\pi} \mid \theta)$ where $\hat{\pi}$ is the most probable derivation tree. For notational convenience, we will use $e_v(a_i,a_j,a_k,a_l)$ indicating that for type$(v) = U_{1L}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_i)$, for type$(v) = U_{1R}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_j)$, for type$(v) = U_{2L}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_k)$, for type$(v) = U_{2R}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_l)$, for type$(v) = P$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_i,a_l)$ and for the other types, $e_v(a_i,a_j,a_k,a_l) = 1$. The detailed description of the 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 $\gamma_v(i,i-1,j,j-1) \leftarrow 0$
  if type$(v) = S$
    then $\gamma_v(i,i-1) \leftarrow \max_{y \in \mathcal{C}_v} [\log t_v(y) + \gamma_y(i,i-1,i,i-1)]$

$\tau_v(i, i-1) \leftarrow \arg\max_{(y,-1)} [\log t_v(y) + \gamma_y(i, i-1, i, i-1)]$

if type$(v) = B_1$

then $\gamma_v(i, i-1, j, j-1) \leftarrow \max_{z \in \mathcal{C}_v} [\log t_v(y, z) + \gamma_y(i, i-1) + \gamma_z(i, i-1, j, j-1)]$

$\tau_v(i, i-1, j, j-1) \leftarrow \arg\max_{(y,z,-1)} [\log t_v(y, z) + \gamma_y(i, i-1) + \gamma_z(i, i-1, j, j-1)]$

if type$(v) = B_2$

then $\gamma_v(i, i-1, j, j-1) \leftarrow \max_{z \in \mathcal{C}_v} [\log t_v(y, z) + \gamma_z(i, i-1, j, j-1) + \gamma_y(j, j-1)]$

$\tau_v(i, i-1, j, j-1) \leftarrow \arg\max_{(y,z,-1)} [\log t_v(y, z) + \gamma_z(i, i-1, j, j-1) + \gamma_y(j, j-1)]$

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 $j = i-1$, $l = k-1$

then skip

else $\gamma_v(i, j, k, l) \leftarrow -\infty$

if type$(v) = S$

then if $j = i-1$

then skip

else $\gamma_v(i, j) \leftarrow \max_{y \in \mathcal{C}_v} \max_{h=i-1,\ldots,j} [\log t_v(y) + \gamma_y(i, h, h+1, j)]$

$\tau_v(i, j) \leftarrow \arg\max_{(y,h)} [\log t_v(y) + \gamma_y(i, h, h+1, j)]$

if type$(v) = B_1$

then if $j = i-1$, $l = k-1$

then skip

else $\gamma_v(i, j, k, l) \leftarrow \max_{z \in \mathcal{C}_v} \max_{h=i-1,\ldots,j} [\log t_v(y, z) + \gamma_y(i, h) + \gamma_z(h+1, j, k, l)]$

$\tau_v(i, j, k, l) \leftarrow \arg\max_{(y,z,h)} [\log t_v(y, z) + \gamma_y(i, h) + \gamma_z(h+1, j, k, l)]$

if type$(v) = B_2$

then if $j = i-1$, $l = k-1$

then skip

else $\gamma_v(i, j, k, l) \leftarrow \max_{z \in \mathcal{C}_v} \max_{h=k-1,\ldots,l} [\log t_v(y, z) + \gamma_z(i, j, k, h) + \gamma_y(h+1, l)]$

$\tau_v(i, j, k, l) \leftarrow \arg\max_{(y,z,h)} [\log t_v(y, z) + \gamma_z(i, j, k, h) + \gamma_y(h+1, l)]$

else if $j = i-1$, $l = k-1$

then skip

else $\gamma_v(i, j, k, l)$

$\leftarrow \max_{y \in \mathcal{C}_v} [\log e_v(a_i, a_j, a_k, a_l) + \log t_v(y) + \gamma_y(i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R})]$

$\tau_v(i, j, k, l)$

$\leftarrow \arg\max_{y} [\log e_v(a_i, a_j, a_k, a_l) + \log t_v(y) + \gamma_y(i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R})]$ □

When the calculation terminates, we obtain $\log P(w, \hat{\pi} \mid \theta) = \gamma_1(1, n)$. The time complexity of the algorithm is $O(m^2 n^5)$ since there are at most $O(m)$ iterative loops over five position indices i, j, k, l, h and one nonterminal index $z \in \mathcal{C}_v$. It is apparent that the space complexity is $O(mn^4)$. 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.

4.2 Scoring Problem

As in SCFGs [8], 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 sum ones 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 probabilities $\beta_v(i, j)$ and $\beta_y(i, j, k, l)$ of all derivation trees excluding subtrees rooted at a nonterminal W_v generating a subsequence $a_i \cdots a_j$ and of all derivation trees excluding subtrees rooted at a nonterminal W_y generating a tuple of subsequences $(a_i \cdots a_j, a_k \cdots a_l)$ respectively. What has to be noticed is calculating the outside variables β requires the inside variables α which the inside algorithm calculates. Unlike CYK and inside algorithms, the outside algorithm recursively works its way inward. 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$(v) = S$, $\mathcal{P}_v = \{y\}$, type$(y) = B_1$, $\mathcal{C}_y = \{v, z\}$
then $\beta_v(i, j) \leftarrow \sum_{z \in \mathcal{C}_y} \sum_{h=j}^{n} \sum_{k'=h+1}^{n+1} \sum_{l'=k'-1}^{n} \beta_y(i, h, k', l') t_y(v, z) \alpha_z(j + 1, h, k', l')$
if type$(v) = S$, $\mathcal{P}_v = \{y\}$, type$(y) = B_2$, $\mathcal{C}_y = \{v, z\}$
then $\beta_v(i, j) \leftarrow \sum_{z \in \mathcal{C}_y} \sum_{h=1}^{n+1} \sum_{k'=h-1}^{n} \sum_{l'=k'+1}^{i} \beta_y(h, k', l', j) t_y(v, z) \alpha_z(h, k', l', i - 1)$
if type$(v) \neq S$, $\mathcal{P}_v = \{y\}$, type$(y) = B_1$, $\mathcal{C}_y = \{z, v\}$
then $\beta_v(i, j, k, l) \leftarrow \sum_{h=1}^{i} \beta_y(h, j, k, l) t_y(z, v) \alpha_z(h, i - 1)$
if type$(v) \neq S$, $\mathcal{P}_v = \{y\}$, type$(y) = B_2$, $\mathcal{C}_y = \{z, v\}$
then $\beta_v(i, j, k, l) \leftarrow \sum_{h=l}^{n} \beta_y(i, j, k, h) t_y(z, v) \alpha_z(l + 1, h)$
else $\beta_v(i, j, k, l)$
$\leftarrow \sum_{y \in \mathcal{P}_v} \beta_y(i - \Delta_y^{1L}, j + \Delta_y^{1R}, k - \Delta_y^{2L}, l + \Delta_y^{2R}) e_y(a_{i-\Delta_y^{1L}}, a_{j+\Delta_y^{1R}}, a_{k-\Delta_y^{2L}}, a_{l+\Delta_y^{2R}}) t_y(v)$ □

The time and space complexities of the algorithm are the same as those of CYK and inside algorithms.

4.3 Training Problem

The training problem for G_s can be solved by the EM (expectation maximization) algorithm called the inside-outside algorithm where the inside variables α and outside variables β are used to re-estimate probability parameters. The expected number of times that a nonterminal W_v is used for training sequences $w^{(r)}$ $(1 \leq r \leq N)$ is

$$E(v) = \begin{cases} \sum_{r=1}^{N} \frac{1}{P(w^{(r)} \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \alpha_v^{(r)}(i, j) \beta_v^{(r)}(i, j), & (\text{type}(v) = S) \\ \sum_{r=1}^{N} \frac{1}{P(w^{(r)} \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \alpha_v^{(r)}(i, j, k, l) \beta_v^{(r)}(i, j, k, l) & (\text{type}(v) \neq S) \end{cases}$$

where $\alpha^{(r)}$ and $\beta^{(r)}$ are the inside and outside variables calculated for each input sequence $w^{(r)}$. The expected numbers of times that W_v makes a transition to W_y, or W_y and W_z, and W_v emits one or two symbols are obtained in a similar way. Then we can calculate the re-estimated values of $t_v(y)$, $t_v(y, z)$ and $e_v(a_i, a_j, a_k, a_l)$ by dividing the corresponding expected counts by $E(v)$.

5 Conclusion

We have presented a polynomial time parsing algorithm and a parameter estimation method using a specific SMCFG for RNA sequences. Our future work is to implement an initial grammar by utilizing the annotation of consensus secondary structure in a multiple alignment of an RNA family and then train the grammar by executing our algorithms. Furthermore, we will try experiments such as RNA secondary structure prediction with pseudoknots from a single sequence and database search for structural homology recognition.

Acknowledgments

This work is supported by Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists.

References

[1] M. Brown and C. Wilson, “RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search,” Pacific Symposium on Biocomputing, pp. 109–125, 1996.

[2] L. Cai, R. L. Malmberg and Y. Wu, “Stochastic modeling of RNA pseudoknotted structures: a grammatical approach,” Bioinformatics, vol. 19, suppl. 1, pp. i66–i73, 2003.

[3] R. Durbin, S. R. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis, Cambridge University Press, 1998.

[4] S. R. Eddy and R. Durbin, “RNA sequence analysis using covariance models,” Nucleic Acids Research, vol. 22, no. 11, pp. 2079–2088, 1994.

[5] T. Kasami, H. Seki and M. Fujii, “Generalized context-free grammar and multiple context-free grammar,” IEICE Transactions on Information & Systems, vol. J71-D, no. 5, pp. 758–765, 1988 (in Japanese).

[6] T. Kasami, H. Seki and M. Fujii, “On the membership problem for head languages and multiple context-free languages,” IEICE Transactions on Information & Systems, vol. J71-D, no. 6, pp. 935–941, 1988 (in Japanese).

[7] Y. Kato, H. Seki and T. Kasami, “On the generative power of grammars for RNA secondary structure,” IEICE Transactions on Information & Systems, vol. E88-D, no. 1, pp. 53–64, 2005.

[8] K. Lari and S. J. Young, “The estimation of stochastic context-free grammars using the inside-outside algorithm,” Computer Speech and Language, vol. 4, pp. 35–56, 1990.

[9] E. Rivas and S. R. Eddy, “A dynamic programming algorithm for RNA structure prediction including pseudoknots,” Journal of Molecular Biology, vol. 285, pp. 2053–2068, 1999.

[10] E. Rivas and S. R. Eddy, “The language of RNA: A formal grammar that includes pseudoknots,” Bioinformatics, vol. 16, no. 4, pp. 334–340, 2000.

[11] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood and D. Haussler, “Stochastic context-free grammars for tRNA modeling,” Nucleic Acids Research, vol. 22, pp. 5112–5120, 1994.

[12] H. Seki, T. Matsumura, M. Fujii and T. Kasami, “On multiple context-free grammars,” Theoretical Computer Science, vol. 88, pp. 191–229, 1991.

[13] Y. Uemura, A. Hasegawa, S. Kobayashi and T. Yokomori, “Tree adjoining grammars for RNA structure prediction,” Theoretical Computer Science, vol. 210, pp. 277–303, 1999.