RNA Structure Prediction Including Pseudoknots Based on Stochastic Multiple Context-Free Grammar

Yuki Kato, Hiroyuki Seki, and Tadao Kasami

Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{yuuki-ka, seki, 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 method based on the EM algorithm. Furthermore, we show some experimental results of pseudoknot prediction using SMCFG algorithm.

1 Introduction

Many attempts have so far been made at modeling RNA secondary structure including pseudoknots. Brown and Wilson [2] proposed a model based on intersections of stochastic context-free grammars (stochastic CFGs, SCFGs) [5, 13] to describe RNA pseudoknots. Cai et al. [3] introduced a model based on parallel communication grammar systems using a single CFG synchronized with a number of regular grammars. Akutsu [1] provided dynamic programming algorithms for predicting pseudoknots without using grammars. On the other hand, several grammars have been proposed where the grammar itself can fully describe pseudoknots. Rivas and Eddy [11, 12] designed a dynamic programming algorithm for predicting RNA pseudoknotted structure, and introduced a new class of grammars called RNA pseudoknot grammars (RPGs). Uemura et al. [15] defined specific subclasses of tree adjoining grammars (TAGs) named SL-TAGs and ESL-TAGs respectively, and predicted RNA pseudoknots by using parsing algorithm of ESL-TAG. Matsui et al. [10] proposed pair stochastic tree adjoining grammars (PSTAGs) based on ESL-TAGs and tree automata for aligning and predicting pseudoknots. These grammars have generative power stronger than CFGs and polynomial time algorithms for parsing problem.

In our previous work [8], we identified RPGs, SL-TAGs and ESL-TAGs as subclasses of *multiple context-free grammars* (MCFGs) [7, 14], which can model

pseudoknots, and showed a candidate subclass of the minimum grammars for representing pseudoknots. In this paper, we extend the above candidate subclass of MCFGs to a probabilistic model called SMCFG. We then present a polynomial time parsing algorithm for finding the most probable derivation tree and a probability parameter estimation method based on the EM algorithm. Finally, we show some experimental results of pseudoknot prediction using SMCFG parsing algorithm.

2 Multiple Context-Free Grammar

A *multiple context-free grammar* (MCFG) [7, 14] 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, $\dim(S) = 1$. 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*. Examples of rules are $A \to f[A]$ and $A \to g[\,]$ where $f, g \in F$ and are defined by $f[(x_1, x_2)] = (ax_1b, cx_2d)$ and $g[\,] = (ab, cd)$. A sample derivation is $A \Rightarrow (ab, cd)$ by the second rule, which in turn gives $A \Rightarrow f[(ab, cd)] = (aabb, ccdd)$ together with the first rule. Due to limitation of the space, definition of derivation tree of MCFG is omitted (see [14]). Intuitively, MCFGs can derive terminal sequences with arbitrary number of gaps, which leads to generative power stronger than CFGs.

3 SMCFG

We extend a subclass of MCFG to a probabilistic model called stochastic MCFG (SMCFG). An SMCFG 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 for indicating End, Start, Delete, Bifurcation, Unpair and Pair respectively (see Table 1). Delete nonterminals are used to deal with gaps in sequence alignment. The type of W_v is denoted by $\mathrm{type}(v)$ and we predefine $\mathrm{type}(1) = \mathrm{S}$, that is, W_1 is the start symbol. Consider a sample rule set $W_v \to UP_{1L}[W_y] \mid UP_{1L}[W_z]$ for U_{1L} where $UP_{1L}^{\alpha}[(x_1, x_2)] = (\alpha x_1, x_2)$ and α is a variable for which a terminal is substituted. Let $t_v(y)$ be the *transition probability* that $W_v \to UP_{1L}[W_y]$ is applied. Let $e_v(a)$ be the *emission probability* that $\alpha = a$ where a is a terminal. The sum of the transition/emission probabilities with the same left hand side is one. All the transition probabilities of Bifurcation nonterminals are defined as one since most of the nonterminals for modeling RNA structure have the type of either Unpair or Pair, and Bifurcation nonterminals are sometimes used to deal with concatenating and wrapping operation. This single choice of transition for Bifurcation nonterminal reduces time complexities of SMCFG algorithms. Let $\mathcal{C}_v$ be the set of indices

y such that W_v can make a transition to W_y. To avoid non-emitting cycles, we assume that the nonterminals are numbered such that $v < y$ for all $y \in \mathcal{C}_v$. The probability of a derivation tree is defined as the product of transition and emission probabilities of all rules used in the derivation.

Table 1. SMCFG G_s

Type	Rule set	Function	Transition	Emission
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}	$W_v \to UP_{1L}[W_y]$	$UP_{1L}[(x_1, x_2)] = (a_i x_1, x_2)$	$t_v(y)$	$e_v(a_i)$
U_{1R}	$W_v \to UP_{1R}[W_y]$	$UP_{1R}[(x_1, x_2)] = (x_1 a_j, x_2)$	$t_v(y)$	$e_v(a_j)$
U_{2L}	$W_v \to UP_{2L}[W_y]$	$UP_{2L}[(x_1, x_2)] = (x_1, a_k x_2)$	$t_v(y)$	$e_v(a_k)$
U_{2R}	$W_v \to UP_{2R}[W_y]$	$UP_{2R}[(x_1, x_2)] = (x_1, x_2 a_l)$	$t_v(y)$	$e_v(a_l)$
P	$W_v \to BP[W_y]$	$BP[(x_1, x_2)] = (a_i x_1, x_2 a_l)$	$t_v(y)$	$e_v(a_i, a_l)$

4 Algorithms for SMCFG

We mention the way to find the most probable derivation tree of G_s for an input sequence. This can be solved by a dynamic programming algorithm similar to CYK algorithm for SCFGs [4], 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. 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 a set of probability parameters. The detailed description of the CYK algorithm is shown in Fig. 1. 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 ($m = a + b$), the time and space complexities of the CYK algorithm are $O(amn^4 + bn^5)$ and $O(mn^4)$, respectively. Note that we need traceback to recover the optimal derivation tree.

In order to re-estimate the probability parameters of G_s, we design the inside-outside algorithm based on the EM algorithm. For further information, refer to [9].

5 Experimental Results

The dataset for experiments was taken from an RNA family database called "Rfam" (version 7.0) [6] which is a database of multiple sequence alignment and covariance models [5] 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). Corona_pk_3 ranges in length from 62 to 64 and has a simple pseudoknotted structure, whereas HDV_ribozyme and Tombus_3_IV range in length from 87 to 92 and have more complicated structures with pseudoknot. We specified 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 [4]. The other sequences in the alignment were used as the test sequences for prediction. 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.

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 2. A nearly correct prediction (94.4% precision and recall) for Corona_pk_3 is shown in Fig. 2 where underlined base pairs agree with trusted ones. The secondary structures predicted by our algorithm agree very well with the trusted structures. Furthermore, we compared the prediction accuracy of our SMCFG algorithm with that of PSTAG algorithm [10] (see Table 3).

Table 2. Prediction results (The numbers of test sequences are 10 in Corona_pk_3, 10 in HDV_ribozyme and 12 in Tombus_3_IV respectively.)

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

6 Conclusion

In this paper, we have proposed a probabilistic model named SMCFG, and designed a polynomial time parsing algorithm and a parameter estimation method 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.

Table 3. Comparison of SMCFG with 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

References

1. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Applied Mathematics, Vol. 104 (2000) 45–62
2. Brown, M., Wilson, C.: RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. Proc. Pacific Symposium on Biocomputing (1996) 109–125
3. Cai, L., Malmberg, R.L., Wu, Y.: Stochastic modeling of RNA pseudoknotted structures: a grammatical approach. Bioinformatics, Vol. 19, suppl. 1 (2003) i66–i73
4. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis. Cambridge University Press (1998)
5. Eddy, S.R., Durbin, R.: RNA sequence analysis using covariance models. Nuc. Acids Res., Vol. 22, No. 11 (1994) 2079–2088
6. Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., Eddy, S.R.: Rfam: an RNA family database. Nuc. Acids Res., Vol. 31, No. 1 (2003) 439–441
7. Kasami, T., Seki, H., Fujii, M.: Generalized context-free grammar and multiple context-free grammar. IEICE Trans. Inf. & Syst., Vol. J71-D, No. 5 (1988) 758–765 (in Japanese)
8. Kato, Y., Seki, H., Kasami, T.: On the generative power of grammars for RNA secondary structure. IEICE Trans. Inf. & Syst., Vol. E88-D, No. 1 (2005) 53–64
9. Kato, Y., Seki, H.: Stochastic multiple context-free grammar for RNA pseudoknot modeling. NAIST Info. Sci. Tech. Rep., NAIST-IS-TR2006002 (2006)
10. Matsui, H., Sato, K., Sakakibara, Y.: Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. Bioinformatics, Vol. 21, No. 11 (2005) 2611–2617
11. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. J. Mol. Biol., Vol. 285 (1999) 2053–2068
12. Rivas, E., Eddy, S.R.: The language of RNA: A formal grammar that includes pseudoknots. Bioinformatics, Vol. 16, No. 4 (2000) 334–340
13. Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjölander, K., Underwood, R.C., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nuc. Acids Res., Vol. 22 (1994) 5112–5120
14. Seki, H., Matsumura, T., Fujii M., Kasami, T.: On multiple context-free grammars. Theor. Comput. Sci., Vol. 88 (1991) 191–229
15. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammars for RNA structure prediction. Theor. Comput. Sci., Vol. 210 (1999) 277–303

Initialization:
for $i \leftarrow 1$ **to** $n+1$, $j \leftarrow i$ **to** $n+1$, $v \leftarrow 1$ **to** m
  do if $\mathrm{type}(v) = \mathrm{E}$
    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 $\mathrm{type}(v) = \mathrm{E}$
    then if $j = i-1$ **and** $l = k-1$
      then skip
    else $\gamma_v(i, j, k, l) \leftarrow -\infty$
  if $\mathrm{type}(v) = \mathrm{S}$
    then $\gamma_v(i, j) \leftarrow \max_{y \in \mathcal{C}_v} \max_{h=i-1,\ldots,j} [\log t_v(y) + \gamma_y(i, h, h+1, j)]$
  if $\mathrm{type}(v) = \mathrm{B}_1$
    then $\gamma_v(i, j, k, l) \leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(i, h) + \gamma_z(h+1, j, k, l)]$
  if $\mathrm{type}(v) = \mathrm{B}_2$
    then $\gamma_v(i, j, k, l) \leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(h+1, j) + \gamma_z(i, h, k, l)]$
  if $\mathrm{type}(v) = \mathrm{B}_3$
    then $\gamma_v(i, j, k, l) \leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i, j, h+1, l) + \gamma_y(k, h)]$
  if $\mathrm{type}(v) = \mathrm{B}_4$
    then $\gamma_v(i, j, k, l) \leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i, j, k, h) + \gamma_y(h+1, l)]$
  if $\mathrm{type}(v) = \mathrm{P}$
    then if $j = i-1$ **or** $l = k-1$
      then $\gamma_v(i, j, k, l) \leftarrow -\infty$
    else $\gamma_v(i, j, k, l) \leftarrow \max_{y \in \mathcal{C}_v} [\log e_v(a_i, a_l) + \log t_v(y) + \gamma_y(i+1, j, k, l-1)]$
  else $\gamma_v(i, j, k, l) \leftarrow \max_{y \in \mathcal{C}_v} [\log e_v(a_i, a_j, a_k, a_l) + \log t_v(y)$
    $+ \gamma_y(i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R})]$

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

Fig. 1. CYK algorithm

Corona_pk3 (EMBL accession #: X51325.1)

[Trusted structure in Rfam]

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

[Prediction by SMCFG]

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

Fig. 2. Comparison of a prediction result with a trusted structure in Rfam