RNA Pairing Pattern Recognition Using Stochastic Context-Free Grammars and Evolutionary History

Ali Ahmadi Esfidi

January 24, 2025

Abstract

This study explores advancements in RNA pairing pattern recognition through the integration of Stochastic Context-Free Grammars (SCFGs) with evolutionary information. A refined parsing approach is proposed to improve accuracy in predicting complex structural motifs, including pseudoknots. By leveraging evolutionary models and enhancing dynamic programming techniques—such as the Gap-Bracket CYK algorithm—the method achieves notable improvements in predictive performance. Applications to curated datasets demonstrate these accuracy gains while highlighting ongoing challenges in managing sequence variability and optimizing computational efficiency for scalability.

1 Introduction

RNA molecules take part in most of the major biological processes, ranging from the regulation of gene expression to catalyzing enzymatic reactions. Key to understanding the function of RNAs lies with nucleotide pairing in forming secondary structures—helical stems and loops—but most traditional approaches to prediction, ignoring complex motifs like pseudoknots, cannot but are constrained to sacrificing the accuracy of structure prediction.

This lecture will focus on how to extend the capabilities of Stochastic Context-Free Grammars (SCFGs) to capture more adequately RNA pairing patterns, including pseudoknots, by incorporating evolutionary information from multiple sequence alignments. We will delve into great detail the foundational KH99 (Knudsen–Hein) [6] and Pfold [7] algorithms, which combine phylogenetic analysis with predictive modeling for improved accuracy. Building on these methods, we describe a multi-pass algorithm that iteratively revisits sequence alignments, flags identified stem pairs, and refines predictions to uncover crossing pairs. By designing grammars capable of accommodating such complex motifs combined with iterative parsing, this approach extends our abilities in the prediction of biologically relevant RNA structures and offers deeper insights into RNA functionality.

2 Related Work

Over the years, a number of algorithms have been developed to predict RNA secondary structures, addressing both non-crossing and crossing base-pair interactions. KH-99 (Knudsen and Hein, 1999) [6] introduced a framework that combined SCFGs with evolutionary information, which improved the accuracy by leveraging phylogenetic relationships. Pfold [7] and its extension, PPFold [10], further refined SCFG-based methods by integrating explicit phylogenetic trees for greater efficiency and scalability. More complex approaches, such as Stochastic Multiple Context-Free Grammars (SMCFGs) [5] or Pair Stochastic Tree Adjoining Grammars (PSTAGs) [8], have been developed in order to predict pseudoknots, which are crossing interactions that are not captured by classical SCFGs, though at a computational cost. Complementing these methods, tools like Knotify [1] focus on pseudoknot detection as a post-processing step.

Deep learning, in only recent years, has become a powerful tool in RNA structure prediction. Models such as E2Efold [2] and SPOT-RNA [9] exploit neural networks to predict base-pairing probabilities directly, including pseudoknots, balancing accuracy with efficiency. Those kinds of approaches show that a machine learning-based approach can in principle transcend some of the basic limitations of the traditional algorithms, yielding improved performances over complex RNA structures.

3 Methodology

It takes an alignment of RNA sequences as input and outputs a consensus RNA pairing pattern present in the sequences. At the core of this model are two

key components: the SCFG to model the structural pairing features, and the evolutionary model that incorporates phylogenetic information to improve prediction accuracy.

3.1 Context-Free Grammar

Context-Free Grammar can be defined as a formal system for defining the syntax of languages, presenting the rules in which symbols are expanded into sequences. This hierarchical nature of CFG renders it efficient and effective in modeling natural language, programming languages, and biological entities like RNA. Extensions with probabilities added to CFG result in a Probabilistic Context-Free Grammar, or PCFG for short, wherein possible derivations of sequences are ranked by their likelihood. This probabilistic approach is particularly valuable in computational biology and natural language processing, where the ability to deal with ambiguity and variability is essential.

$$
\begin{aligned}
S &\to L\ S \mid \$M\ E \mid s \\
F &\to \$M\ E \mid L\ S \\
L &\to \$M\ E \mid s \\
\$M &\to B\ F \\
B &\to d \\
E &\to d
\end{aligned}
$$

The grammar above, models the formation of RNA loops and stems. The non-terminal S generates loops, either generating additional loops LS recursively or starting a stem F. Stems are produced by F, which pairs up a start base of B with an end base of E via a match of BF, using $\$M$. A loop, as the term suggests, cannot be less than two positions because of stability; it is ruled by L which may comprise an unmatched base s or beginning a new stem $\$ME$. The unpaired bases are denoted by s, which are produced by both S and L, whereas the paired bases within the stems are denoted by d, produced by B and E.

3.2 Inside-Outside Algorithm

The inside-outside algorithm [3] is a dynamic programming approach that calculates the inside and outside probabilities of a probabilistic context-free grammar. This makes it instrumental in expectation-maximization methods of training PCFGs by defining two kinds of probabilities: the inside probability, which expresses the likelihood that a given nonterminal will generate a substring of the input, and the outside probability, which gives the probability that the remainder of the string is generated, omitting the substring in question. By combining these probabilities, the algorithm helps estimate the contribution of each grammar rule to the observed data. This process is crucial for adjusting the probabilities of rules during model training so that the PCFG can fit the data better. The inside-outside algorithm is widely used in fields like natural language processing, bioinformatics, and other areas involving probabilistic parsing or structure prediction.

3.3 CYK Parser

The CYK, or the Cocke–Youger–Kasami algorithm, works principally on dynamic programming, parsing strings to see whether they belong to some particular language defined by a context-free grammar in Chomsky Normal Form. It builds a triangular table in which every cell contains the non-terminal symbols that can derive specific substrings of the input. The probabilistic version is called the PCYK algorithm, which extends that to incorporate probabilities from a probabilistic context-free grammar or PCFG. Instead of mere testing for derivability, PCYK calculates the probabilities of each of the possible derivations by combining the probabilities of grammar rules and partial parses. This allows the algorithm to pick out the most probable parse for a given string, and it is of particular use in applications involving ambiguity and probabilistic inference.

3.4 Gap-Bracket CYK Parser

The Gap-Bracket CYK Parser extends the traditional CYK parsing algorithm to handle sentences with special tokens ($<$ and $>$) in a more sophisticated manner. By filtering out these tokens and tracking their counts during parsing, this method incorporates the imbalance of unmatched bracket-like symbols into the scoring process. Specifically, nonterminal rules that begin with '$', dynamically adjust their probability scores based on the counts of unpaired angle brackets, and if a specific nonterminal has been used as a nonterminal on the right side in the last step of the generation branch (If yes acceleration occurs). This adjustment is controlled by three parameters: $start\_ratio$, $accelerat\_ratio$, and $flag\_ratio$. These parameters respectively account for the initialization, acceleration, and weighting of bracket-related imbalances in the parse probabilities.

3.5 Probabilities Of Columns

RNA families were taken from the Rfam database. For a solid basis of further analyses, phylogenetic trees are calculated with the tool `phyml`, which guarantees reliable ML inference. ML trees, together with

sequence alignments, allowed determination of both frequencies (P) and rates of mutation (R) for pairing and nonpairing bases. Calculated values were stored for further analysis.

Base frequencies were calculated counting occurrences at individual positions as well as by globally determining the overall base frequency in a dataset. Of course, an XY base pair was considered no different from a YX base pair, due to the fact that pairing is bidirectional.

To estimate mutation rates [6], pairs of sequences from the above-described database were obtained. All ordered pairs were generated using sequences with at least 85% identical base composition. For every pair of sequences, if the same position in one sequence had base X and base Y in the other, the corresponding counter c_{XY} and c_{YX} was incremented. Columns containing gaps in the paired sequences were excluded from consideration. For a given pair p, let t_p represent the time interval between the sequences, N_p denote the number of aligned columns in the two-sequence alignment, and P_s indicate the probability of a base occurring at a single position. We have for $X \neq Y$:

$$R_{XY} \approx \frac{c_{XY}}{2P_X P_s \sum_p (t_p N_p)}$$

and for the rest we have:

$$R_{XX} = -\sum_{X \neq Y} R_{XY}$$

Since the same mutation rates are calculated for paired bases, the pairs were also counted symmetrically.

Of course, in order to have equal reflection of influence with respect to each family in mutation rate and frequency, there is a need to give specific weights to the sequences.

Next, calculation of the probability of all possible paired and unpaired columns would be possible in the input alignment. For that, a maximum-likelihood tree of the input alignment should be constructed using *phyml*. Further, column probabilities could be calculated using post-order traversal in the constructed ML tree.

The probability for the configuration of nucleotides at the tips and interior nodes [4] of the tree in Figure 1 is:

$$P(GUC, AB \mid \tau, v, \theta) = P_A \cdot p_{AG}(v_1) \cdot p_{AU}(v_2) \cdot_{BA} (v_4) \cdot p_{BC}(v_3)$$

Which P_A denotes the frequency or proportion of nucleotide A. $p_{AB}(x)$ is the element in cell AB in the matrix $e^{R \cdot x}$ where R is a rate matrix, while x represents an evolutionary distance or time.

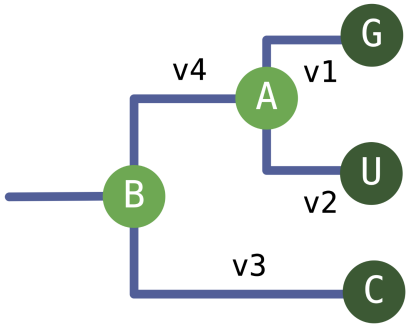

Figure 1: A phylogenetic tree with G, U and C nucleotides in taxons, interior nodes, A and B are unknown

To calculate the unconditional probability of the observed states at the tips of the tree, we sum over all possible combinations of nucleotide states that can be assigned to the interior nodes of the tree:

$$P(GUC \mid \tau, v, \theta) = \sum_{A \in \{A,U,G,C\}} \left(\sum_{B \in \{A,U,G,C\}} P(GUC, AB \mid \tau, v, \theta) \right)$$

Additionally, when integrating an evolutionary model, treating a gap as a fifth nucleotide poses several challenges. To overcome these issues, gaps are treated as unknown nucleotides [7], enabling more accurate and biologically relevant outcomes.

So, probability of any column in input alignment, can be calculated.

3.6 First, Second, and Third Passes

The grammar parameters are estimated using the consensus pairing patterns of RNA families, as described in the "Probabilities of Columns" section. This process is carried out by applying the inside-outside algorithm to a set of pairing patterns. The algorithm is executed once, and the resulting grammar parameters are saved for subsequent analysis and processing.

When an s symbol appears in a production rule, it corresponds to a column in the alignment of sequences. A column has an associated probability incorporated into the production probability whenever s is produced. The rules that produce base pairs multiply the production probabilities by the probabilities of the corresponding columns, provided the columns form a valid base pair [6]. This naturally extends the grammar to generate columns in alignment.

Algorithm 1: Gap-Bracket CYK Parser

Input : A sentence, flag ratios: start_ratio, accelerate_ratio, flag_ratio
Output: Table of probabilities and parse table

Function `Gap-Bracket CYK(`*sentence, start_ratio, accelerate_ratio, flag_ratio*`):`
  Split *sentence* into *words*;
  $filtered\_words, filtered\_indices \leftarrow$ filter non-bracket words;
  $total\_mismatch \leftarrow$ `calculate_mismatch(`*words*`)`;
  Initialize P and *status* as default dictionaries of $-\infty$ and $False$;
  Initialize *table* as an empty dictionary;
  $length \leftarrow$ size of $filtered\_words$;
  for $i \leftarrow 1$ ***to*** *length* **do**
    for $(A \rightarrow w) \in grammar.unary\_rules$ **do**
      if $w = filtered\_words[i-1]$ **then**
        $P[i, i, A] \leftarrow \log(rule\_probs[A, w])$;
        $table[i, i, A] \leftarrow [(i, i, w)]$;
  for $l \leftarrow 2$ ***to*** *length* **do**
    for $i \leftarrow 1$ ***to*** $length + 1 - l$ **do**
      $j \leftarrow i + l - 1$;
      for $k \leftarrow i$ ***to*** $j - 1$ **do**
        for $(A \rightarrow BC) \in grammar.binary\_rules$ **do**
          if $P[i, k, B] > -\infty$ ***and*** $P[k+1, j, C] > -\infty$ **then**
            $start\_idx \leftarrow filtered\_indices[i-1]$;
            $end\_idx \leftarrow filtered\_indices[j-1]$;
            $sign\_count \leftarrow total\_mismatch[start\_idx][end\_idx]$;
            if *A is flagged* **then**
              if $status[i, k, B]$ ***or*** $status[k+1, j, C]$ **then**
                $Prob \leftarrow P[i, k, B] + \log(rule\_probs[A, B, C]) + P[k+1, j, C] + \log(flag\_ratio^{sign\_count} \cdot accelerate\_ratio)$;
              else
                $Prob \leftarrow P[i, k, B] + \log(rule\_probs[A, B, C]) + P[k+1, j, C] + \log(flag\_ratio^{sign\_count} \cdot start\_ratio)$;
            else
              $Prob \leftarrow P[i, k, B] + \log(rule\_probs[A, B, C]) + P[k+1, j, C]$;
            if $Prob > P[i, j, A]$ **then**
              $P[i, j, A] \leftarrow Prob$;
              $table[i, j, A] \leftarrow [(i, k, B), (k+1, j, C)]$;
              $status[i, j, A] \leftarrow (B$ or C is flagged$)$;
  return $P[1, length, "S"], table$;

First Pass: The pairing pattern (excluding crossing pairs) is determined using the Gap-Bracket CYK algorithm applied to the extended grammar, with input alignment.

Second Pass: Base pairs identified in the first pass are flagged, marking the corresponding $\$M$ nonterminals. Another Gap-Bracket CYK algorithm is then applied to the expanded grammar, allowing for the prediction of some other pairing patterns, which are our crossing pairs.

Third Pass (Optional): The process can be repeated to predict all crossing pairs.

4 Implementation

To evaluate evolutionary parameters, six families from the Rfam dataset were initially tested, and three were ultimately selected to represent distinct RNA types for calculating frequencies, mutation rates, and grammar parameters.

Rfam ID	Family Name	Primary Organisms
RF00001	5S rRNA	All domains of life
RF00005	tRNA	All domains of life
RF01704	Glutamine-II Riboswitch	Cyanobacteria, bacteriophages
RF02035	Cobalamin Riboswitch	Bacteria
RF03000	LOOT RNA Motif	Bacteria
RF03054	Xanthine Riboswitch	Bacteria

Table 1: Rfam Families tested for first set of families

Additionally, eight smaller families were chosen to estimate optimal values for the Gap-Bracket parameter.

The first set of families is larger and exhibits a variety of observable mutations, making them suitable for analyzing evolutionary dynamics. In contrast, the second set comprises smaller families with distinct characteristics, enabling the selection of effective parameters for the Gap-Bracket calculation.

4.1 Evolutionary Parameters

The frequencies obtained during the analysis are clearly shown in Table 2. In the stems of the studied sequences, it is seen that GC/CG base pairs dominantly occupy the landscape due to their considerably high binding energy. This high binding energy contributes much to their overall stability as well as their predominance in the observed sequences.

Paired Bases		Single Bases	
GC/CG	53%	A	37%
AU/UA	31%	G	19%
UG/GU	0.1%	U	25%
Others	18%	C	19%
Total	**61%**	**Total**	**39%**

Table 2: Base frequencies calculated from five choosen families.

The mutation rates for single bases are shown in Table 3, with their respective values showing significant variations. Of particular interest is the fact that transitions, namely the A–G and T–C mutations within the DNA structure, occur more frequently than transversions, which include all other kinds of mutations. This observation is in perfect agreement with established biological principles known in the field.

X/Y	A	C	G	U
A	-0.44	0.12	0.17	0.15
C	0.24	-0.77	0.12	0.41
G	0.32	0.12	-0.62	0.17
U	0.21	0.32	0.13	-0.66

Table 3: The entries for the single bases rate matrix

The frequencies for mutations related to the most frequent base pairs are listed in Table 4. As shown in this table, there is evidence that mutations of only one altered base occur most frequently while the ones which require two transversions are very rare and seldom seen.

X/Y	AU	UA	GC	CG	UG	GU
AU	-1.66	0.28	0.68	0.20	0.03	0.25
UA	0.28	-1.66	0.20	0.68	0.25	0.03
GC	0.37	0.11	-1.13	0.24	0.03	0.26
CG	0.11	0.37	0.24	-1.13	0.26	0.03
UG	0.10	0.76	0.16	1.47	-2.84	0.03
GU	0.76	0.10	1.47	0.16	0.03	-2.84

Table 4: Some entries for the pairing bases rate matrix

4.2 Grammar Parameters

After running Inside-Outside Algorithm on the consensus pairing patterns of selected families, the

production probabilities obtained were the following:

$S \rightarrow L\ S\ (91\%) \mid \$M\ E\ (2\%) \mid s\ (7\%)$
$F \rightarrow \$M\ E\ (67\%) \mid L\ S\ (33\%)$
$L \rightarrow \$M\ E\ (7\%) \mid s\ (93\%)$
$\$M \rightarrow B\ F\ (100\%)$
$B \rightarrow d\ (100\%)$
$E \rightarrow d\ (100\%)$

4.3 Gap-Bracket Parameter

The implementation of the Gap-Bracket CYK parser will involve determining appropriate values for start ratio, accelerate ratio, and flag ratio. Since flagged words do not come in the first pass, flag ratio is not needed, and therefore, only start and accelerate ratios need to be optimized in the first pass, while tuning for start ratio, accelerate ratio, and flag ratio features during the second pass.

A genetic algorithm is used to find the optimal values of these parameters. It uses eight small alignments with known consensus pairing patterns to maximize the F1-Score.

To avoid the possibility of converging into local optima, diversity parameters are also used in the implementation of the genetic algorithm to explore the solution space more widely.

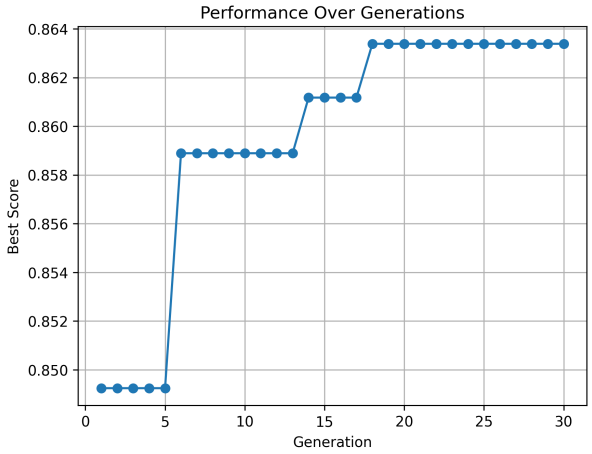

Figure 2: Improvement of genetic algorithm for finding suitable Gap-Bracket parameters (Score without changing hyperparameters: 0.812)

The selected values for Gap-Bracket CYK parser in first and second passes are shown in Table 5. It can be observed that the *start ratio* decreases while the *accelerate ratio* increases in the second pass, reflecting the adjustments needed due to the different nature of the two passes.

-	Start	Accelerate	Flag
1st Pass	0.19	1.48	-
2nd Pass	0.88	1.78	0.98

Table 5: The optimal values for Gap-Bracket CYK in first pass and second pass

5 Results

To evaluate this method, eight families were selected from the Rfam database, and the model's output was compared against their consensus pairing patterns to calculate the F1-score, providing a measure of accuracy and reliability.

Table 6 presents the mean weighted F1-score for each test family. The lowest score is observed for RF01084, which will be analyzed in the next section.

Family	Gap-Bracket
RF00556	86.85%
RF00499	92.24%
RF01084	67.46%
RF03004	100.00%
RF01089	77.76%
RF01093	86.73%
RF01072	81.91%
RF01737	78.40%
Total	82.35%

Table 6: All families used for testing the model are listed in the table. These sequences were not utilized during the implementation process.

Nevertheless, this score represents an improvement compared to the results obtained using the standard CYK parser.

6 Discussion

There are two main problems, that make wrong decisions in the model:

1. Sufficient data was not available for predicting the consensus pairing pattern, as the type of input alignment differed significantly from the families used to implement the model.

2. While treating with gaps as unknown nucleotides, some pairs formed by columns which aren't conserved enough.

To address the first issue, the addition of more families is recommended. To improve the model's performance, the families could be clustered into two or three groups, with a separate model implemented for

.

RF01862

```
Seq1 -------------UCCUAGAGAGCUUGGCCGUCGGGCCUUAUA-CCCGACUUGCUGAGUUUCUCUAGGAGAGU-CC...
Seq2 UUUGAAAGGGGUCUCCUAGAGAGCUUGGCCGUCGGGCCUUACACCCCGACUUGCUGAGUUUCUCUAGGAGAGU-CC...
Seq3 UUUGAAAGGGGUCUCCUAGAGAGCUUGCC--UAGGGCCUUAUAACCCGACUUGCUGAGCUUCUAUAGGAAAAAACC...
Seq4 UUUGAAAGGGGUCUCCUAGAGAGCUUGGCCGUCGGGCCUUAUACCCCGACUUGCUGAGCUUCUCUAGAAGAGU-CC...
Seq5 UUUGAAAGGGGUCUCCUAGAGAGCUUGCC--UAGGGCCUUAUAACCCGACUUGCUGAGCUUCUAUAGGAAAAAACC...
Cons :::<<<<<<<<<<<<<<<<<<<<<<<<-<<<<<<--------->>>>>>--<<<<<<<<<->>>>>>>>>->.>>...
SCYK ...(((((((...........((((.....((((((........))))))......))))).......[[.....))...
GCYK ...(((((((.(((((((((((((((((.((((((........))))))..)))))))).)))))))))))..[))...
```

Cons represents the consensus pairing pattern provided in Rfam, while $SCYK$ shows the output generated using the standard CYK parser, and $GCYK$ displays the output obtained using the Gap-Bracket CYK parser. The test family above illustrates how the Gap-Bracket CYK parser enhances prediction accuracy.

RF01862

```
Seq1 ------------------------------------GCUGCCCUUGGGUUU-UACUCCUUGAACCCUUCGGAAG...
Seq2 ------------------------------------GGUACCCUUGAAAUC-ACCUCCUAGAUUUCUUCGGAAG...
Seq3 ------------------------------------GGUGCCUUUGAGAGU-UACUCUUUGCUCUCUUCGGAAG...
Seq4 ------------------------------------GUUGCCUUUGAGAGU-UACUCCUUGCUCUCUUCAGAAG...
Seq5 ------------------------------------GGUGCCUUUGAACCCUUUAUCCCGGGGUUCUUCGGAAG...
Seq6 ------------------------------------GGCGCCUUUGAAACC-AUCUCCUAGGUUUCUUCGGAAG...
Seq7 ------------------------------------GGUGCCUUUGAAACC-AUCUCCUAGGUUUCUUCGGAAG...
Seq8 ------------------------------------GGCACCUUUGAAACC-AUCUCCUAGGUUUCUUCGGAAG...
Seq9 GGCGUGGUUGACACGCAGACCUCUUACAAGAGUGUCUAGGUGCCUUUGAGAGU-UACUCUUUGCUCUCUUCGGAAG...
Cons ....................................(((((((,,<<<<<<.---AAAAA>>>>>>,,,,,,,<...
Ptr9 ..((((.....)))).(((((((((....)))).)))).(((((((..((((((...{{{{{}}}}}).......(...
GCYK [(((((([[...))))).....(((((..))))).......(((((((..((((((......[[.)))))).......(...
```

*Ptr*9 shows the pairing pattern presented for *Seq*9 in the dataset. The gap problem is evident, and the pairing pattern in regions with a high number of gaps became entirely dependent on a single sequence(*Seq*9).

each cluster. For predicting the most likely pairing pattern for an input alignment, models for each cluster could be run, and the best structure selected based on their probability.

For the second issue, pairs formed by columns that are not sufficiently conserved can be removed.

7 Conclusion

Three assumptions were considered in generating this model:

1. Stems are frequently observed to be either back-to-back or adjacent to each other.
2. The probability of an interior stem varies depending on whether it forms a loop on one side.
3. Two nucleotides are less likely to form a pair if there is a significant number of unclosed nucleotides between them, as this reduces the probability of establishing a stable interaction.

Incorporating parameters such as the *start ratio*, *accelerate ratio*, and *flag ratio*, alongside the evolutionary model, aims to simulate these assumptions and facilitate the identification of the most accurate pairing patterns.

The inclusion of additional biologically relevant parameters, such as the length of the interior loop, along with the incorporation of more families and the use of ensemble learning, can play a crucial role in improving both the accuracy and the biological relevance of the model.

The high-performance cost can be improved using parallelization techniques [11] on the Gap-Bracket CYK parser, as in the case of PPfold [10]. Alternatively, substitution of the CYK algorithm with the Earley algorithm [1] is possible due to its more efficiency in parsing some grammatical structure.

References

[1] C. Andrikos and Makris. Knotify: An efficient parallel platform for rna pseudoknot prediction using syntactic pattern recognition. *Biology*, 12:1–16, 2023.

[2] X. Chen, Y. Li, R. Umarov, X. Gao, and L. Song. Rna secondary structure prediction by learning unrolled algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[3] M. Collins. The inside-outside algorithm. Lecture notes.

[4] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[5] Y. Kato and H. Seki. Stochastic multiple context-free grammar for rna pseudoknot modeling. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 112–123. Association for Computational Linguistics, ACL, 2006.

[6] B. Knudsen and J. Hein. Secondary structure prediction, 1999. Lecture notes.

[7] B. Knudsen and J. Hein. Secondary structure prediction, 2003. Lecture notes.

[8] H. Matsui and Sato. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot rna structures. *Bioinformatics*, 21(11):2611–2617, 2005.

[9] J. Singh and Hanson. Rna secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nature Communications*, 10:5407, 2019.

[10] Z. Sükösd and Knudsen. Ppfold 3.0: fast rna secondary structure prediction using phylogeny and auxiliary data. *Bioinformatics*, 28(20):2691–2692, 2012.

[11] Y. Yi, C.-Y. Lai, S. Petrov, and K. Keutzer. Efficient parallel cky parsing on gpus. pages 175–185, October 2011.

[1] Comparison Between CYK and Earley Algorithms