RNA secondary structures: from ab initio prediction to better compression, and back

Evarista Onokpasa

Joint work with Sebastian Wild Prudence Wong

Data Compression Conference 2023

March 22-24, 2023

UNIVERSITY OF LIVERPOOL

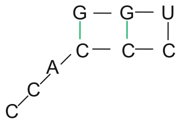
Table of Contents

Introduction: RNA Basics

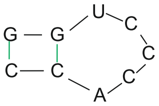
RNA, abbreviation of ribonucleic acid, are complex compounds of high molecular weight that functions in cellular protein synthesis and occur in living organisms. The nitrogenous bases in RNA are adenine(A), guanine(G), cytosine(C) and uracil(U), these bases are linked together in a single strand which forms the primary structure(shown in 1). However, this strand can fold over in different ways to form secondary bonds, this folding over is known as its secondary structure(shown in 2 and 3).

(1) G−G−U−C−C−C−A−C−C

(2)

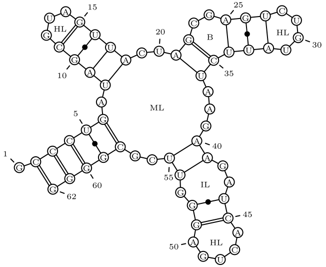


(3)



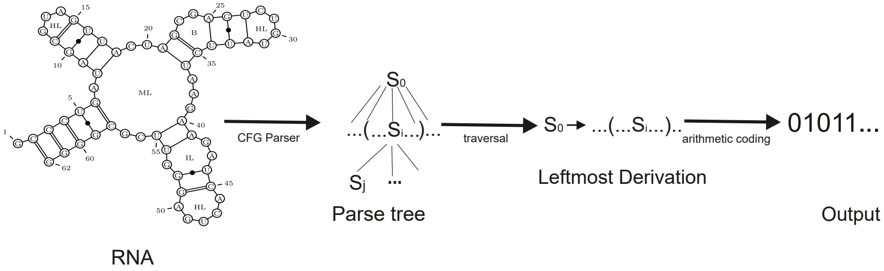
Introduction: Summary of our work

- In this research work, we improve the state of the art in joint compression of RNA (Ribonucleic acids) sequence and structure data (Liu et al., BMC Bioinformatics, 2008).



- we show that compression ratio can serve as a cheap and robust proxy for comparing the prediction quality of different stochastic models, which may help guide the search for better RNA structure prediction models.

Procedure



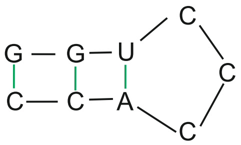
HL
B
HL
ML
IL
HL
CFG Parser
S_0
...(...S_i...)...
S_j
...
Parse tree
traversal
$S_0 \rightarrow$ ...(...S_i...)..
Leftmost Derivation
arithmetic coding
01011...
Output
RNA

Preliminaries: DotBracket Notation

DotBracket Notation for RNA An RNA sequence is a string of bases A, C, G, U. RNA secondary structures can be represented by the dot-bracket notation: a string over {•, (,)} where a base pair is denoted by matching parentheses () and an unpaired base by •; below is a very simple example.

(1) G−G−U−C−C−C−A−C−C

(2)

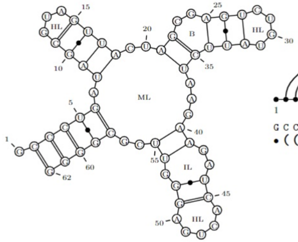


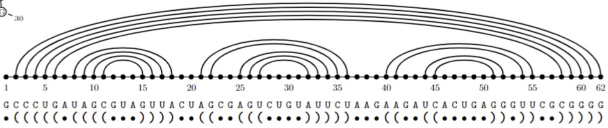
The dotbracket notation for this sample RNA is:

```
GGUCCCACC
(((...)))
```

We use “RNA” as an abbreviation for “a pair of an RNA sequence and its secondary structure”.

Preliminaries: More Detailed Example





1 5 10 15 20 25 30 35 40 45 50 55 60 62
G C C C U G A U A G C G U A G U U A C U A G C G A G U C U G U A U U C U A A G A A G A U C A C U G A G G G U U C G C G G G
•(((((•((((•••))))••((••(((••••)))))))•••((••((•••••))•))••)))))

An example RNA sequence and structure. **Left:** schematic drawing of structure. **Above:** Representation as dot-bracket sequence when the backbone is “pulled straight”.

RNAs as Pairs of Characters

In this research work we define RNAs as Pairs of Characters i.e. Each RNA Primary structure symbol is paired with its corresponding secondary structure Symbol from {•,(,)}. So the RNA

```
GGUCCCACC
(((...)))
```

as pairs of characters is:

$$\begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{U}\\(\end{bmatrix}\begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{A}\\)\end{bmatrix}\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$$

This representation is a simplification to [LYC^{+}08] as it requires a single grammar to produce each RNA.

Preliminaries: SCFG and Earley Parser

SCFG Dot-bracket strings can be generated by a context-free grammar (CFG). A *stochastic context-free grammar* (SCFG) is a tuple $G = (N, T, R, S, P)$ such that (N, T, R, S) is a CFG and $P : R \to [0, 1]$ is a function satisfying $\sum_{(A\to\alpha)\in R} P(A \to \alpha) = 1$ for all $A \in N$.

Earley Parser
The Earley Parsing algorithm [Ear70], similar to the CYK (Cocke–Younger–Kasami) algorithm, is able to process any SCFG and efficiently determine whether a string belongs to the language of the grammar. We use the Earley parser implementation by [Td17] when comparing compression using various SCFGs and for prediction since it does not require the rigid Chomsyky's normal form for grammars.

Preliminaries: Arithmetic Coding

Arithmetic Coding

- Arithmetic Coding [WNC87] "assigns one codeword to each possible data set... with the codeword drawn from the interval $[0, 1)$". Each codeword uniquely identify the subinterval, in which the probabilities of occurrence of the given dataset can be found. **(Unlike the Huffman encoding which spits out codewords per event in a dataset, arithmetic encoding produces one codeword at the end of all the events in a given data set.)**.

- An arithmetic encoder "must work in conjunction with a modeler that estimates the probability of each possible event at each point in the coding [...]. The models can be
  - **adaptive** (dynamically estimating the probability of each event based on all events that precede it),
  - semi-adaptive (using a preliminary pass of the input file to gather statistics), or **nonadaptive** (using fixed or
  - **static** probabilities for all files)" [HV94].

  The next slide illustrates the Arithmetic encoding using a static model:

Preliminaries: Arithmetic coding

RNA compression using SCFG and Arithmetic Coding
This compression is explained using the RNA sequence $\begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$ with the grammar of Liu et al.: $G_L = (N, T, R, S)$ has $N = \{S, L\}$,
$T = \{\begin{bmatrix}\mathtt{A}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{A}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}\}$, and rules R shown in the table below

rule	prob.	interval	rule	prob.	interval	rule	prob.	interval
$S \to LS$	0.65	$[0.00, 0.65)$	$L \to \begin{bmatrix}\mathtt{C}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.10	$[0.20, 0.30)$	$L \to \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}$	0.10	$[0.50, 0.60)$
$S \to \varepsilon$	0.35	$[0.65, 1.00)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$	0.05	$[0.30, 0.35)$	$L \to \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}$	0.15	$[0.60, 0.75)$
$L \to \begin{bmatrix}\mathtt{A}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.05	$[0.00, 0.05)$	$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.05	$[0.35, 0.40)$	$L \to \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}$	0.10	$[0.75, 0.85)$
$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{A}\\)\end{bmatrix}$	0.15	$[0.05, 0.20)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.10	$[0.40, 0.50)$	$L \to \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}$	0.15	$[0, 85, 1.00)$

The grammar in this table is the Liu grammar [LYC+08] but the probabilites and partitions were just made up for illustrative purpose.
The (unique) leftmost derivation using the grammar is as follows:

$$S \Rightarrow LS \Rightarrow \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}S\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}S \Rightarrow \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}LS\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}S \Rightarrow \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}S\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}S \Rightarrow \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}\varepsilon\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}S \Rightarrow \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{C}\\)\end{bmatrix}\varepsilon = \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}\begin{bmatrix}\mathtt{C}\\)\end{bmatrix},$$

preliminaries: SCFG and arithmetic coding continued

rule	prob.	interval	rule	prob.	interval	rule	prob.	interval
$S \to LS$	0.65	$[0.00, 0.65)$	$L \to \begin{bmatrix}\mathtt{C}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.10	$[0.20, 0.30)$	$L \to \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}$	0.10	$[0.50, 0.60)$
$S \to \varepsilon$	0.35	$[0.65, 1.00)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$	0.05	$[0.30, 0.35)$	$L \to \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}$	0.15	$[0.60, 0.75)$
$L \to \begin{bmatrix}\mathtt{A}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.05	$[0.00, 0.05)$	$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.05	$[0.35, 0.40)$	$L \to \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}$	0.10	$[0.75, 0.85)$
$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{A}\\)\end{bmatrix}$	0.15	$[0.05, 0.20)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.10	$[0.40, 0.50)$	$L \to \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}$	0.15	$[0, 85, 1.00)$

where the sequence on applied production rules is

$$S \to LS, \quad L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}, \quad S \to LS, \quad L \to \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}, \quad S \to \varepsilon, \quad S \to \varepsilon.$$

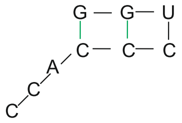
We obtain the corresponding sequence of intervals from the rules, $[0.00, 0.65), [0.30, 0.35), [0.00, 0.65), [0.50, 0.60), [0.65, 1.00), [0.65, 1.00)$; which we encode using arithmetic coding to obtain the final binary codeword: 0011010100100.

Preliminaries: RNA Secondary Structure

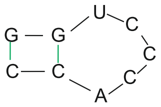
Secondary Structure RNAs have a Primary structure, which is a sequence of characters from A,C,G,U.(show in 1) Biologically, this sequence can fold over in different ways to form its secondary structure as shown in 2 and 3 below

(1) G−G−U−C−C−C−A−C−C

(2)



(3)



Preliminaries: Secondary Structure Prediction and Prediction Quality

Determining RNA secondary structure requires expensive, techniques like X-ray crystallography [TM16]. This has driven the interest in RNA secondary structure prediction.
SCFG can be used for RNA secondary structure prediction where terminals correspond to bases and the leftmost derivation of an RNA sequence encodes a secondary structure of the sequence.

The prediction quality can be measured using the metrics sensitivity and positive predictive value (Predicted base pairs that are in the trusted structure are true positives (TP); predicted base-pairs not in the trusted structure are false positives (FP); base-pairs in the trusted structure but not predicted by the algorithm are false negatives (FN)).
Then Sensitivity $= \frac{TP}{TP+FN}$ and PPV $= \frac{TP}{TP+FP}$.

Preliminaries: Earley Parser and Secondary Structure Prediction

Earley Parser and Secondary Structure Prediction
Given an RNA primary sequence $w \in \{a|c|g|u\}^*$ and a Stochastic Context Free Grammar G for defining or generating RNA secondary structure. How accurately can we predict the secondary structure for w? Earley Parser is a dynamic programming algorithm that can explore all possible parse trees (to find the one with largest value).The steps are as follows:

- splitting w recursively into its substrings
- producing all the possible parse trees for the splits
- obtaining the scores (i.e. the product of probabilities of rules applied) for each parse tree.
- selecting the parse tree with the maximum score.
- this parse tree is returned as the predicted secondary structure

preliminaries continued

Suppose we have a simple RNA with primary sequence GUC and an SCFG, $G_L = (N, T, R, S)$ has $N = \{S, L\}$, $T = \{\begin{bmatrix}\mathtt{A}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\(\end{bmatrix}, \begin{bmatrix}\mathtt{A}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}, \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}, \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}\}$, with the rules and probability distribution shown here again.

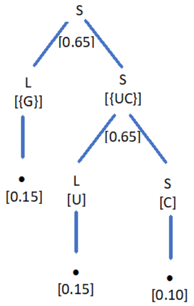
rule	**prob.**	**interval**	**rule**	**prob.**	**interval**	**rule**	**prob.**	**interval**
$S \to LS$	0.65	$[0.00, 0.65)$	$L \to \begin{bmatrix}\mathtt{C}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.10	$[0.20, 0.30)$	$L \to \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}$	0.10	$[0.50, 0.60)$
$S \to \varepsilon$	0.35	$[0.65, 1.00)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$	0.05	$[0.30, 0.35)$	$L \to \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}$	0.15	$[0.60, 0.75)$
$L \to \begin{bmatrix}\mathtt{A}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.05	$[0.00, 0.05)$	$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.05	$[0.35, 0.40)$	$L \to \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}$	0.10	$[0.75, 0.85)$
$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{A}\\)\end{bmatrix}$	0.15	$[0.05, 0.20)$	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.10	$[0.40, 0.50)$	$L \to \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}$	0.15	$[0, 85, 1.00)$

Using the Earley parser algorithm, we compute the parsetrees for the substring of the primary sequence GUC using the following steps

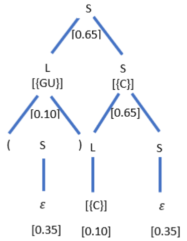
- The substrings are:
  1. {{G},{U},{C}}
  2. {{G},{UC}}
  3. {{GU},{C}}
  4. {{GUC}}
- We obtain the possible parsetrees for the substrings using the grammar
  - From the grammar definition the start symbol S has one rule
    $S \Rightarrow LS$
    we cannot arrive at the 3 substrings {{G},{U},{C}} because the rhs of the rule has only 2 Non-terminals. The score for this substring split is 0.

Preliminaries: Secondary Structure Prediction Example

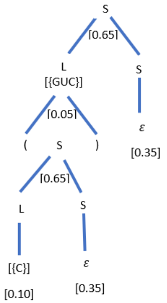
- **for {{G},{UC}} there is only one possible way to arrive at the substrings this is shown in the parse tree below. Multiplying all the probabilities for rules ($0.65^2 * 0.15^2 * 0.10$) applied we obtain the score of $\underline{9.5 * 10^{-3}}$ [HIGHEST SCORE].**



- **for the substrings {{GU}{C}} The parse tree is shown here and the score obtained is ($0.65^2 * 0.35^2 * 0.10^2 =$ $5.18 * 10^{-4}$).**

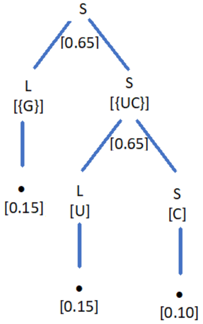


- **for the substrings {{GUC}} The only unique parse tree which can be produced which is different from the other substring is shown below and its score is ($0.65^2 * 0.35^2 * 0.10 * 0.05 =$ $2.58 * 10^{-4}$).**



Preliminaries continued

The parse tree with the highest score of $9.5 * 10^{-3}$ is:



Thus the predicted secondary structure is: • • •

Related Work

- Liu et al. [LYC$^+$08] produced a compressor they termed RNACompress by creating a simple RNA grammar and applying an LL(1) parser and Huffman codes. The compression ratio results showed an improvement over 3 other compression methods existing at the time [LYC$^+$08].

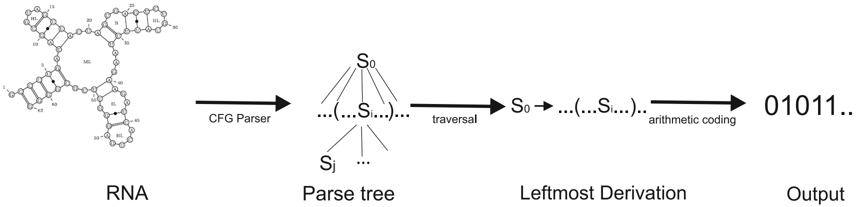
- Naganuma et al. [NHY$^+$20] explore a related method of SCFG compression closer to grammar-based compression using straight-line programs. They create a stochastic grammar from the text to compress with a variation of the RePair heuristic [LM00].

- Friemel [Fri20] explored labelled trees property of RNA sequences for compression. His algorithm contracts tree nodes formed from sequences of multiple dots in the secondary structure or a sequence of multiple nested brackets in the dot-bracket notation. After the node contraction the algorithm encodes the contracted node tree using Huffman coding. RNAContract, is the term he used for his compression method. In his work, the results of RNAContract outperformed RNAcompress in terms of compression ratio.

- Dowell and Eddy [DE04] used simple SCFGs to predict RNA secondary structure and compared their prediction quality. Similarly, Schulz [Sch12] used probabilistic models and SCFGs to predict RNA secondary structures.

Our Approach: Compression

For Compression



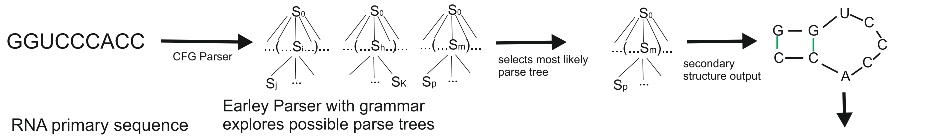
17000 RNA samples[Fri20]

8 Grammars(G1, G2,...,G8) from [De04] and 1 huge grammar by Nebel and Scheid. G2, G7, and G8 (have "stacking parameters") and G1,G3,G4,G5,G6 have fewer parameters

Arithmetic Coding is carried out using static, adaptive and semi adaptive model. static model(Liu et al. used, static model with huffman encoding) which protects from overfitting (if the grammar has many parameters and the RNA is not huge). the adaptive model is helpful if not all RNA structures share the same features. The semi-adaptive shows how good an adaptive model could get.

Our Approach: Prediction

For Prediction



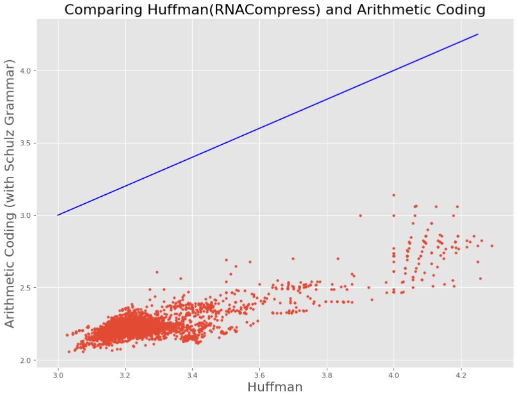
403 RNA sequences [De04]

8 Grammars(G1, G2,...,G8) from [De04] and 1 huge grammar by Nebel and Scheid[NS11]. G2, G7, and G8 (have "stacking parameters") and G1,G3,G4,G6 have fewer parameters

Results - Joint Compression of RNA Structure

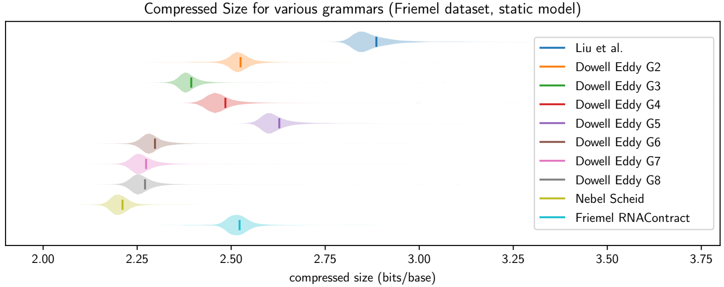
- In this research we obtain the joint compression of RNA structure (i.e. the combined primary and secondary structure). This compression was done using 8 refined stochastic context free grammars from [DE04], 1 very detailed grammar from [Sch12] and the [LYC+08] grammaer using 17 000 rna samples from [Fri20].

- The compressor which we called the joint-rna-compressor was designed using java.The compression results(using the [Sch12]) showed a 45% improvement over RNACompress by Liu et al [LYC+08]!



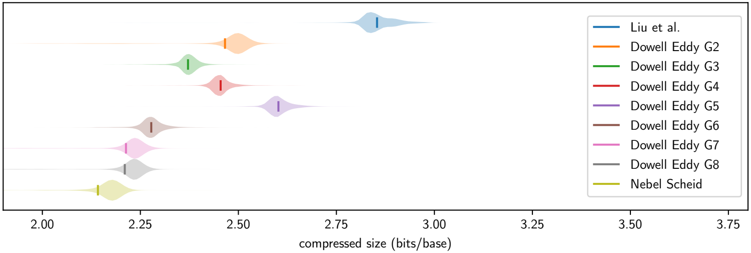
Results - Comparing The compression Results for Refined grammars

- **Compression** of entire data set of 17,000 RNA samples from [Fri20] using 8 grammars from [DE04], 1 grammar from [NS11] and 1 grammar from [LYC$^+$08]. The results are shown for the 3 different probability models and the static model is compared along side Friemel's RNAContract result. This is shown below and in the next slide. The means (are shown with vertical bars) and distributions (are the shaded violin plot) of the normalized compressed size using various grammars on Friemel's RNA dataset.
(All compressed sizes are shown as bits per base. The figure below shows the compression quality of different grammars, normalized to the (average) number of bits per base in the RNA).

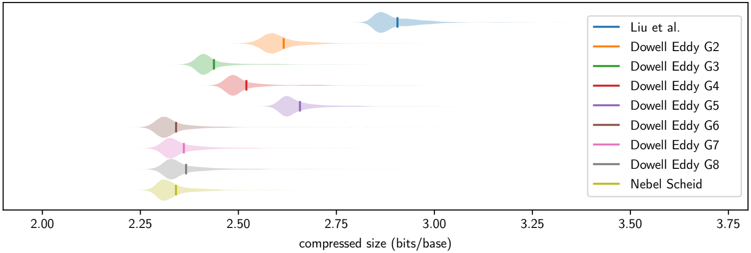


Results Continued

Compressed Size for various grammars (Friemel dataset, semiadaptive model)



Compressed Size for various grammars (Friemel dataset, adaptive model)



Results Continued

- **Compression vs. Prediction Quality.** using Mixed 80 data set from [DE04] as training data, compression and prediction was carried out on the benchmark data from [DE04]. The geometric mean of the prediction quality results (positive predictive value and sensitivity) was obtained, and the result below was produced.

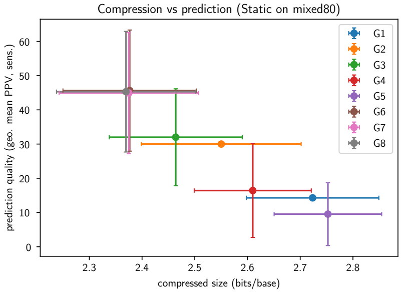


Figure: Compression vs. Prediction (Static probabilities on Mixed)

Results Continued

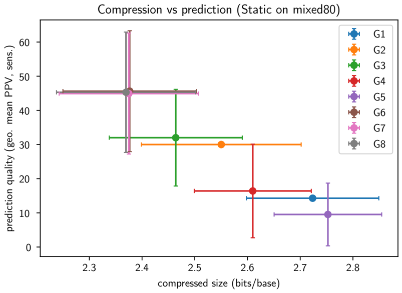


Figure: Compression vs. Prediction (Static probabibilities on Mixed)

This scatter plot shows a clear and strong negative correlation between compressed size and prediction quality; in particular, there is a clearly distinct cluster of grammars that simultaneously give the best compression and the best prediction.
At least for the grammars from [DE04], this shows that one can use compressed size as a more rigidly defined and robust proxy for secondary-structure prediction quality.

Grammars which produce better compression produce better prediction!

Conclusion

In this paper,

- we demonstrated how domain knowledge of RNA secondary structures encapsulated in stochastic context-free grammars for structure prediction can be used to obtain the best single-RNA compression ratios known for this type of data.
- We showed promising first evidence for the utility of compression ability as a cheap and robust proxy for prediction quality for RNA secondary-structure prediction.
- For future research, we could use compression ability as simpler guide, to discover new promising models for secondary-structure prediction. It would be interesting to investigate whether the robust correlation between prediction quality and compressed size continues to hold for large grammars with many parameters.
- Since many natural RNA secondary structures contain "pseudoknots", a principled approach for compressing such structures would be interesting. If the compression-prediction correlation can be demonstrated in this domain as well.

Thank you!

References I

Robin D Dowell and Sean R Eddy.
Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction.
BMC Bioinformatics, 5, 2004.

Jay Earley.
An efficient context-free parsing algorithm.
Communications of the ACM, 13, 1970.

Jonas Friemel.
Contraction-Based Compression of RNA Secondary Structures.
BSc dissertation, Universitat Bielefeld, 2020.

P.G. Howard and J.S. Vitter.
Arithmetic coding for data compression.
Proceedings of the IEEE, 82, 1994.

N.J. Larsson and A. Moffat.
Off-line dictionary-based compression.
Proceedings of the IEEE, 88(11):1722–1732, 2000.

References II

Qi Liu, Yu Yang, Chun Chen, Jiajun Bu, Yin Zhang, and Xiuzi Ye.
RNACompress: Grammar-based compression and informational complexity measurement of rna secondary structure.
BMC Bioinformatics, 9, 2008.

Hiroaki Naganuma, Diptarama Hendrian, Ryo Yoshinaka, Ayumi Shinohara, and Naoki Kobayashi.
Grammar compression with probabilistic context-free grammar.
In *2020 Data Compression Conference (DCC)*. IEEE, 2020.

Markus E. Nebel and Anika Scheid.
Evaluation of a sophisticated SCFG design for RNA secondary structure prediction.
Theory in Biosciences, 130(4):313–336, 2011.

Anika Schulz.
Sampling and Approximation in the Context of RNA Secondary Structure Prediction Algorithms and Studies Based on Stochastic Context-Free Modeling.
Phd dissertation, Technische Universität Kaiserslautern, 2012.

Maarten Trompper (digitalheir).
Probabilistic earley parser, 2017.

References III

Douglas H. Turner and David H. Mathews, editors.
RNA Structure Determination.
Springer New York, 2016.

Ian H. Witten, Radford M. Neal, and John G. Cleary.
Arithmetic coding for data compression.
Communications of the ACM, 30, 1987.