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

Evarista Onokpasa* Sebastian Wild* Prudence W.H. Wong*

February 22, 2023

Abstract

In this paper, we use the biological domain knowledge incorporated into stochastic models for *ab initio* RNA secondary-structure prediction to improve the state of the art in joint compression of RNA sequence and structure data (Liu et al., BMC Bioinformatics, 2008). Moreover, we show that, conversely, 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.

Our results build on expert stochastic context-free grammar models of RNA secondary structures (Dowell & Eddy, BMC Bioinformatics, 2004; Nebel & Scheid, Theory in Biosciences, 2011) combined with different (static and adaptive) models for rule probabilities and arithmetic coding. We provide a prototype implementation and an extensive empirical evaluation, where we illustrate how grammar features and probability models affect compression ratios.

1. Introduction

In this article, we explore the interplay and potential symbiosis between data compression and probabilistic methods for predicting the folding structure of (non-coding) RNA molecules. Ribonucleic acid (RNA) is a bio-polymer that serves various roles in the coding, decoding, expression and regulation of genes in cells. An RNA molecule consists of a chain of *nucleotides* each having a *base* attached to it (either adenine (`A`), cytosine (`C`), guanine (`G`), or uracil (`U`)); this string of bases forms the *sequence* of the molecule. Unlike the related DNA, RNA is usually single-stranded and forms spatial structures by folding onto itself (similar to proteins), with complementary bases forming a stabilizing hydrogen bond. The set of (indices of the) bases that form such pairs is the *secondary structure* of the molecule; it can be encoded by the dot-bracket notation, (see Figure 1; a formal definition is given in Section 2).

The secondary structure is instrumental for the biological function of non-coding RNA molecules and of great interest to biologists. Much research has hence been devoted to computationally *predict* the secondary structure from a known RNA sequence (*ab initio* RNA secondary-structure prediction) [4, 9, 26], including human swarm intelligence [15], and it remains an active research area [22, 7, 23]. We explore areas around RNA secondary structures where innovations in compression methods are central for further progress.

*University of Liverpool, UK, {evarista.onokpasa, sebastian.wild, pwong}@liverpool.ac.uk

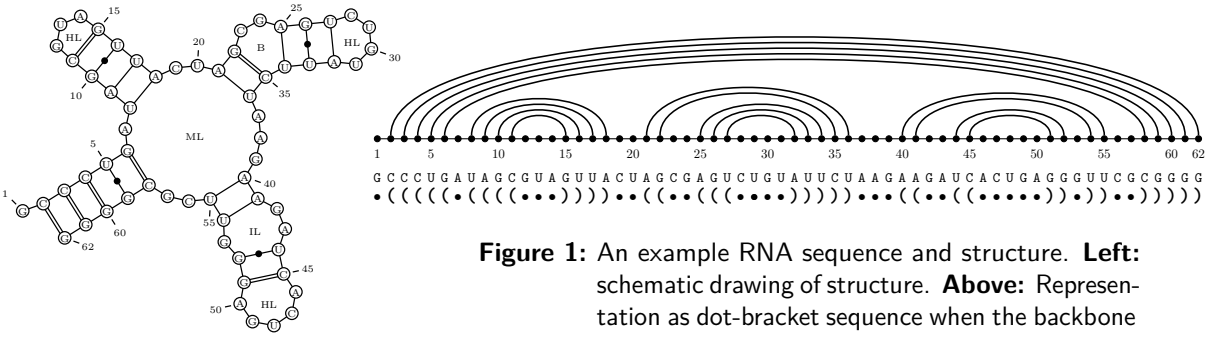

Figure 1: An example RNA sequence and structure. **Left:** schematic drawing of structure. **Above:** Representation as dot-bracket sequence when the backbone is "pulled straight".

Better RNA Compression. Our first goal is to use the domain knowledge on RNA foldings incorporated into secondary-structure prediction models for improved methods for the joint compression of the sequence and secondary structure of RNA sequences. With biological databases ever increasing, compressed representations become desirable. In the case of databases for non-coding RNA sequences with known secondary structures, the data volume has long remained manageable, but growth is now accelerating: For example, *RNA Central* [2] now aggregates over 25 million trusted secondary structures 8 years after its first release; 1.8 million of these come from the *rfam* database [11], collected over its 20 years of existence.

The need for space-efficient representations of joint RNA sequence and secondary structure databases has been identified by Liu et al. in 2008 [16]. Their algorithm *RNACompress*, based on a stochastic context-free grammar (SCFG, defined below), has been recognized as an early application of ideas from grammar-based compression in the data-compression community [17, 12]. As we demonstrate in this article, substantially better compression ratios can be achieved than Liu et al. report; interestingly, by carefully extending their very method to a general framework of SCFG-based compression. Improvements are then realized by applying this framework on tried and tested grammars from the RNA secondary structure *prediction* literature [3, 20] (as well as further, orthogonal refinements).

Apart from the practical utility of less space, compression methods are of direct interest in bioinformatics as a way to upper bound the Kolmogorov complexity [13] of a dataset, and hence its inherent information content [8]. For example in the context of RNA sequences, one can ask how much additional information is contained in the secondary structure of the RNA when the sequence is known.

Compression as a proxy for predictive power. Our second and main goal is to test our hypothesis that for comparing probabilistic models for RNA secondary structures, *compression ratio can serve as proxy for prediction quality in RNA secondary-structure prediction*. Advances in next-generation sequencing allows determining the sequence of many molecules at scale, whereas secondary structures need to be determined by much more expensive techniques like X-ray crystallography [26]. A much cheaper and faster alternative is to computationally *predict* the structure from a known sequence. The state-of-the-art approaches either build on a chemical model of the molecules and try to identify a structure with minimal free energy or use a machine-learning approach. Both can formally be described by stochastic context-free grammars (see Section 2).

RNA secondary-structure prediction plays a vital role in studying the biological function of RNA molecules and for designing artificial RNA sequences, and so numerous software packages implement different algorithms for this task. Comparing their prediction quality is a delicate undertaking, because no definitive similarity metric is known to judge how close the predicted secondary structure is from an experimentally determined one [18]. Indeed, the method of

choice in the literature to compare structure prediction is solely based on individual base pairs [18, 3, 21, 20]: One compares the *sensitivity* and *positive predictive value (PPV)* of different approaches (defined in Section 2).

We will use the *compressed size* (in bits per base) of the reference structure under the trained stochastic model as a more direct means to compare how well different models capture RNA folding behavior. This compressed size effectively reflects the *log-likelihood* of the reference structure and hence has a natural interpretation as the information content that model assigns to the RNA structure.

This has several advantages over sensitivity/PPV: (a) It directly evaluates the quality of the *model*, separating it from the method to produce a (single) predicted secondary structure. There are different options to predict a structure; one can use the most likely structure, or a consensus structure containing the most likely individual pairs, or return a sample of several nearly optimal structures. No choice clearly dominates the others, but they affect the sensitivity and PPV scores. (b) Log-likelihood is a single natural metric derived from first principles of information theory; it does not need trade-offs or further parameters.

Contributions. Our contributions are as follows. First, we improve the compression ratio achieved for joint RNA sequence and structure data by 45% over the state of the art, Liu et al.'s RNACompress [16]; compared to the general-purpose compressor paq8l (*http://mattmahoney.net/dc/#paq*), we see a 70% improvement. The improvement over RNACompress is the combined result of several refinements, but a 30% reduction in compressed size is observed when keeping everything but the used SCFG constant. This clearly shows the relevance of the grammar and the validity of our approach to employ structure-prediction grammars. The proposal and implementation of the more sophisticated grammars (such as the one based on [20]) is hence a useful contribution. Second, we demonstrate that compression ratio can be used as a robust predictor of how well a grammar will perform for *ab initio* secondary-structure prediction. To our knowledge, this is the first such attempt to identify suitable probabilistic models for RNA structure prediction that is not based on comparing predicted structures to a benchmark dataset. Finally, we reproduce and confirm the computational study of [3] with an independent implementation and additional modifications to their grammars.

Related Work. Liu et al. [16] proposed RNACompress in 2008; we discuss their methodology in detail in Section 3. Naganuma et al. [19] 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 [14]. For a broader context of grammar-based compression, see the recent survey of Kieffer and Yang [12]. Friemel [6] also targets the joint RNA compression problem, but using a different approach. He encodes RNA structures as labeled trees with each node representing a nucleotide and the branches representing the bonds; unpaired bases yield unary nodes. Friemel's algorithm RNAContract contracts sequences of unary nodes (similar to compact tries) 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.

Outline. The rest of this paper is structured as follows. Section 2 collects required concepts. Section 3 explains the grammar-based compression of RNA. Then we report on our two studies: Section 4 discusses the compression achieved with various grammars and Section 5 explores the connection between compressed size and prediction quality. We conclude in Section 6 with future work. In the appendix, we give details on the comparison with a general-

purpose compressor (Appendix A), list the precise grammars we used (Appendix B), and investigate further differences between our approach and [16] (Appendix C). Further details, all datasets and code to produce the figures in this article are available online as supplementary material: *https://www.wild-inter.net/publications/onokpasa-wild-wong-2023*; the code is available on GitHub: *https://github.com/evita35/joint-rna-compression*.

2. Preliminaries

Dot-bracket notation. An RNA sequence is a string of bases `A`, `C`, `G`, `U`. Stable hydrogen bonds are possible between `A` and `U` resp. `C` and `G` (the Watson-Crick pairs) and to a lesser extent also between `G` and `U`. RNA secondary structures[1] can be represented by the dot-bracket notation [10]: a well-nested string over $\{\bullet, (,)\}$ where a base pair is denoted by matching parentheses `()` and an unpaired base by $\bullet$; see Figure 1 for an example. We use "RNA" as an abbreviation for "a pair of an RNA sequence and its secondary structure".

SCFG. Dot-bracket strings can be generated by a context-free grammar (CFG). A CFG is a tuple (N, T, R, S) where N and T are finite sets of *nonterminals* and *terminals*, respectively, $R \subseteq N \times (N \cup T)^*$ is a finite set of *production rules*, and $S \in N$ is the *start symbol*. A rule in R is written as $A \to \alpha$. A *stochastic context-free grammar* (SCFG) is a tuple $G = (N, T, R, S, W)$ such that (N, T, R, S) is a CFG and $W : R \to [0, 1]$ is a function satisfying $\sum_{(A\to\alpha)\in R} W(A \to \alpha) = 1$ for all $A \in N$. For every $A \in N$, W represents a probability distribution over the set of rules with left-hand side A.

Earley Parser. The Earley Parsing algorithm [5] is able to process any SCFG and efficiently determine whether a string belongs to the language of the grammar. We use the Earley parser implementations by [25, 27] when comparing various SCFGs since it does not require a rigid normal form for grammars.

RNA secondary-structure prediction. A stochastic context-free grammar 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 used SCFGs allow many different derivations (and hence secondary structures) for a given sequence and the rule probabilities induce a probability distribution over those. Using a classical machine-learning approach, the rule probabilities are chosen as maximum likelihood parameters w.r.t. a given training dataset (with known secondary structures). For predicting/inferring the (unknown) secondary structure of a new RNA sequence, a probabilistic parser determines the maximum-likelihood derivation (Viterbi parse) of the RNA sequence in the SCFG, which encodes the most likely secondary structure (under the given probabilistic model).

We measure the quality of prediction by *sensitivity* and *positive predictive value (PPV)*: the fraction of correctly predicted base pairs among all pairs in the reference structure resp. all pairs in the predicted structure. More formally, let TP, TN, FP, FN be the number of base pairs that are true positives, true negatives, false positives, and false negatives, respectively. Then Sensitivity $= \frac{TP}{TP+FN}$ and PPV $= \frac{TP}{TP+FP}$.

[1] As is often done in the area, we do not consider structures with pseudoknots in this paper.

3. RNA compression using stochastic context-free grammars

We now show how to jointly compress an RNA sequence and secondary structure using a SCFG G. This method has been used by Liu et al. [16] on a fixed grammar; we generalize it here to arbitrary grammars G and rule-probability models. The terminals of G are pairs of characters, e.g., $\left[\begin{smallmatrix}\texttt{A}\\(\end{smallmatrix}\right]$ for base `A` in the RNA sequence and `(` in the (dot-bracket representation of the) secondary structure.[2] To encode an RNA, we determine the sequence of rules in a leftmost derivation of the RNA and then encode this sequence of rules using a model for the rule probabilities using a standard code; Liu et al. use a fixed Huffman code; we employ arithmetic coding [28].

We illustrate the process on the RNA sequence $\left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right]\left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right]\left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right]$ with the grammar of Liu et al.: $G_L = (N, T, R, S)$ has $N = \{S, L\}$, $T = \{\left[\begin{smallmatrix}\texttt{A}\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{C}\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{U}\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{A}\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{G}\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{U}\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{C}\\\bullet\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{G}\\\bullet\end{smallmatrix}\right], \left[\begin{smallmatrix}\texttt{U}\\\bullet\end{smallmatrix}\right]\}$, and rules R shown in Table 1. The (unique) leftmost derivation using the grammar is as follows:

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

Table 1: A (fictitious) set of rule probabilities for the grammar of Liu et al. [16], including the partition of the unit interval as used in arithmetic coding.

$$S \Rightarrow LS \Rightarrow \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right] S \Rightarrow \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] LS \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right] S \Rightarrow \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right] S \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right] S \Rightarrow \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right] \varepsilon \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right] S \Rightarrow \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right] \varepsilon = \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right] \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right],$$

where the sequence on applied production rules is

$$S \to LS, \quad L \to \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right], \quad S \to LS, \quad L \to \left[\begin{smallmatrix}\texttt{A}\\\bullet\end{smallmatrix}\right], \quad S \to \varepsilon, \quad S \to \varepsilon.$$

Since we always replace the leftmost nonterminal, the next nonterminal to replace is known inductively, and we can reconstruct the leftmost derivation from only the (index of the) used right-hand sides: $1, 4, 1, 7, 2, 2$, using the order of rules in Table 1; (the 4 indicates that the second used rule, where we know it expends L, is the 4th rule with left-hand side L, i.e., $L \to \left[\begin{smallmatrix}\texttt{G}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\texttt{C}\\)\end{smallmatrix}\right]$). Now suppose we have the (static) rule probabilities for R from Table 1 and we use arithmetic coding to store the right-hand sides. 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`.

The example above (and [16]) uses a *static rule-probability model*, usually obtained from a training dataset with known structures by counting how often each rule is used in the dataset derivations. With arithmetic coding, we can easily replace this by an *adaptive rule-probability model*, where rule probabilities are computed as relative frequencies in the prefix encoded so far (starting with some initial value for counters, typically 1). This entirely avoids the need for a second pass or a training dataset, as well as storing the rule probabilities. For long inputs, the adaptive model converges to the sequence-specific relative rule frequencies; we hence also include the *semi-adaptive model* where rule counts are determined for the given sequence in a first pass. Unless one also stores the rule counts, this model does not allow decoding, but indicates the limiting behavior of the adaptive model.

[2] Liu et al. use 2 grammars instead – one for the sequence and one for the secondary structure – the two descriptions are equivalent.

4. Joint compression of RNA sequence and secondary structure

To investigate the effectiveness of different parameters, we have developed a generic prototype implementation in Java that allows us to combine arbitrary SCFGs, rule-probability models, and final encoders (Huffman or arithmetic coding). We use an existing open-source Earley Parser implementation [25] for obtaining a parse tree (given a SCFG and an RNA with sequence and structure).[3] Apart from G_L from [16], we use the structure-prediction grammars from [3] and [20]. Since non-canonical bonds are regularly found in experimentally determined secondary structures, all our grammars come in two versions: one that only allows the Watson-Crick and "`G-U` wobble" pairs, and one that allows all 16 pairs. The difference for compression is small: while most RNA structures do contain non-canonical bonds, most contain only very few of them.

For the compression-quality study, we use the "friemel" dataset, consisting of 17 000 ribosomal RNAs from [1] where ambiguously sequenced bases, non-canonical base pairs and pseudoknots have been removed [6]. Information of each RNA in the given datasets is stored in a text file, using the dot-bracket notation. 24 contained empty hairpin loops; since 2 grammars from [3] exclude these, we replaced the innermost pair by two unpaired bases; for the evaluation, we exclude these 24 RNAs.

Figure 2 shows the compression quality of different grammars, normalized to the (average) number of bits per base in the RNA. It is striking that the current state-of-the-art method from the literature, Liu et al.'s RNACompress [16], performs much worse than all the structure-prediction grammars (for all rule-probability models), indicating that these grammars indeed incorporate effective domain knowledge about RNA structures. Also note that a simplistic encoding of the RNA sequence alone would use 2 bits/base; the most sophisticated grammars come very close to that for the joint encoding of sequence *and* structure: 2.21 bits/base on average for the grammar of Nebel and Scheid [20]. The large grammars G_2, G_7, and G_8 [3] (those with "stacking parameters") and the huge grammar by Nebel and Scheid [20] perform overall best. But some much smaller grammars like G_6 come very close, despite having a factor 10 fewer parameters. This shows that it is the structure of the grammar, not merely the number of parameters of the model, that improve compression of RNA secondary structures.

5. Compression ratio vs. prediction quality

We have seen that the choice of the grammar heavily influences the compression quality of our generic joint RNA compressor. In this section, we take a closer look at this grammar dependence from the perspective of both compression and secondary-structure prediction. For that, we reproduced the classic study of Dowell and Eddy [3] comparing several hand-crafted SCFG for their ability to correctly infer RNA secondary structures given only the RNA sequence as input. Due to the bugs from [25], we here used the probabilistic Earley parser from [27]. We use the original datasets from [3] (available at *http://eddylab.org/software/conus/*): The "benchmark" dataset was used in [3] to compare the prediction quality of SCFGs, whose rule probabilities have been trained on their "mixed80" dataset; see [3] for further details. Both datasets contain many non-canonical bonds and 8 RNAs contain empty hairpin loops; we again eliminated the latter. Mixed80 contains numerous ambiguous bases; these were randomly replaced with a compatible base.

Figure 3 shows the results of comparing for each grammar how well it compresses the

[3] This parser has been reported to yield incorrect results for certain inputs; for the compression experiment, we could confirm that it works correctly on all our inputs and grammars.

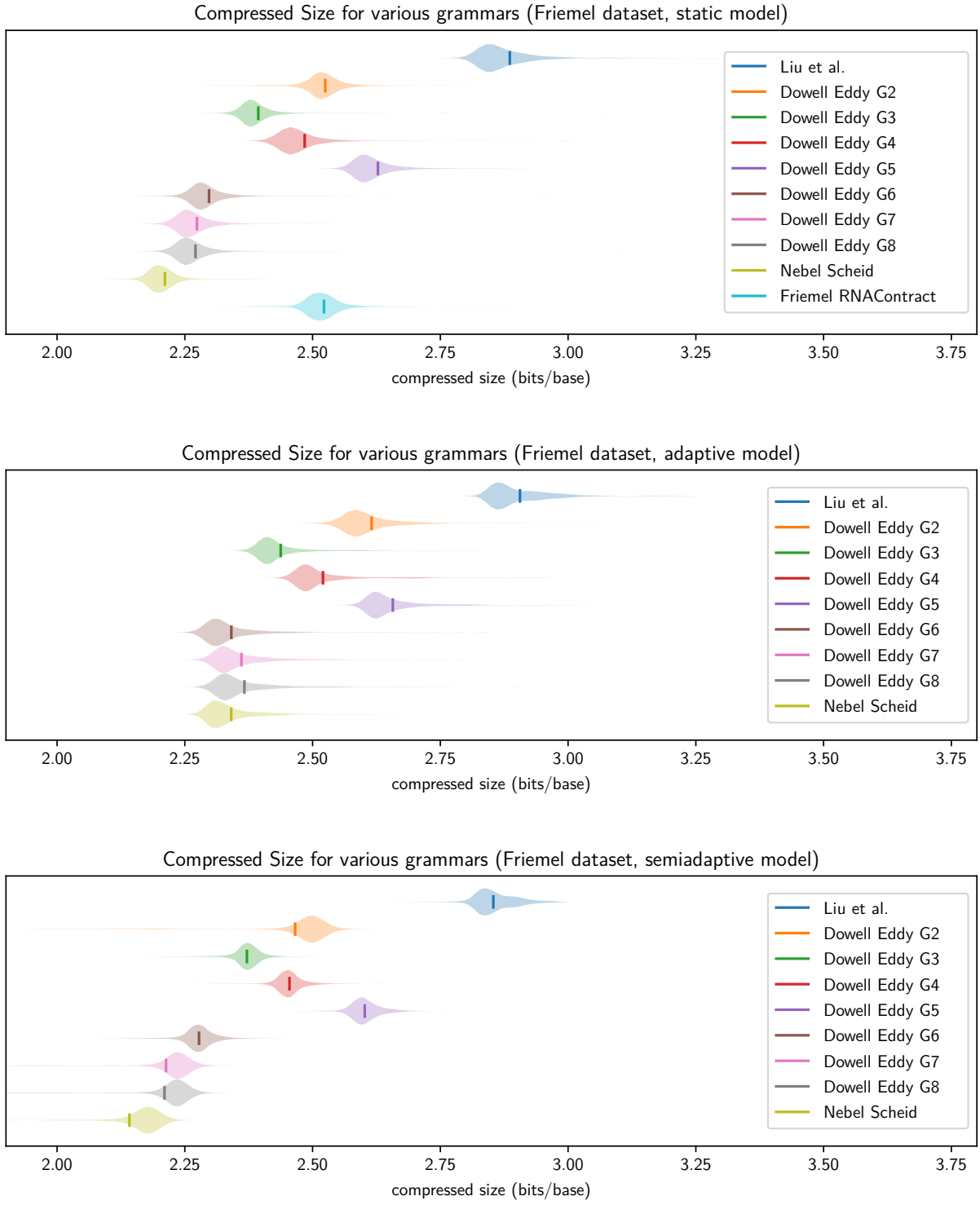

Figure 2: Means (vertical bars) and distributions (shaded violin plot) of the normalized compressed size using various grammars on Friemel's RNA dataset. All compressed sizes are shown as bits per base. **Top:** results using static rule probabilities, determined from the entire dataset. **Middle:** results using adaptive rule-probabilities model (LaPlace model). **Bottom:** semi-adaptive rule probabilities (ignoring space for storing rule probabilities).

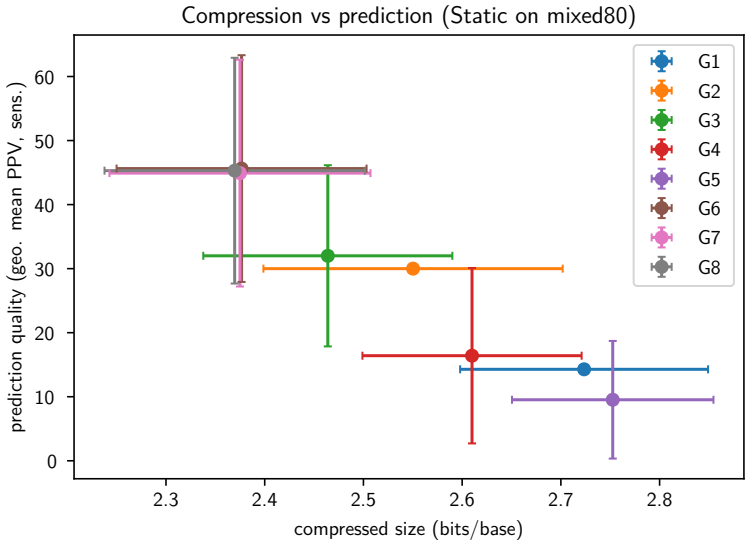

Figure 3: Scatter plot of compression vs. prediction quality for the grammars from [3]. Each grammar is presented as one point with error bars. The x-axis shows the compressed size (in bits per base) for joint compression of RNA sequence and secondary structure, averaged over the benchmark dataset [3]. Horizontal error bars show one standard deviation of compressed size over the benchmark dataset. The y-axis shows the geometric mean of sensitivity and PPV (for each predicted RNA secondary structure, averaged over the benchmark dataset); error bars show one standard deviation. For the ambiguous grammars G_1 and G_2, no vertical error bars are available (we did not reproduce predictions for these; the average is taken from [3]). Both compression and prediction use the same training dataset (mixed80 from [3]) to determine the parameters of the grammars; compression here uses the static model for rule probabilities.

benchmark dataset of RNAs and how well it predicts secondary structures of this set (using the setup and parameters as in [3]). Taking into account the variability across different RNAs within the dataset, a clear and strong negative correlation is visible 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 [3], this shows that one can use compressed size as a more rigidly defined and robust proxy for secondary-structure prediction quality.

Figure 4 takes a closer look at the correlation on a per-RNA level. Even there, a correlation remains visible; in particular very accurately predicted structures are also well compressed. The right panel in Figure 4 shows that compressed size for different grammars is very strongly correlated; pictures for other grammar pairs are similar (excluding the poor performing G_1, G_4, and G_5). Note that despite the strong correlation at RNA level, there is a significant difference in the (mean) compression ratio between different grammars. This might indicate that there are intrinsically more and less "surprising" RNA secondary structures (knowing only the RNA sequence).

6. 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. Moreover, we showed promising

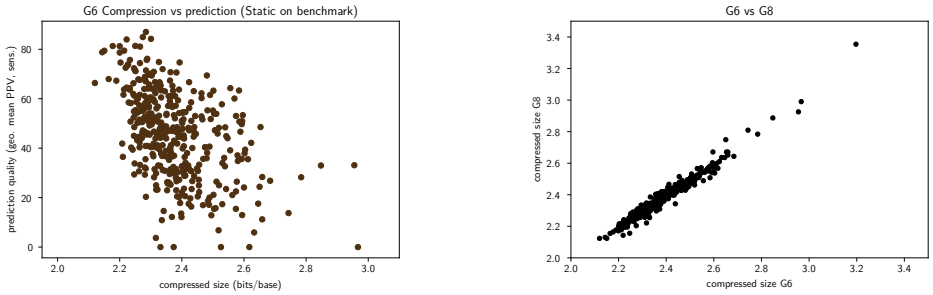

Figure 4: Scatter plots with one point per RNA sequence in the benchmark dataset. **Left:** compressed size against prediction quality using G_6. **Right:** compressed size using G_6 against compressed size using G_8. All compression methods use the static rule probabilities trained on mixed80.

first evidence for the utility of compression ability as a cheap and robust proxy for prediction quality for RNA secondary-structure prediction.

This work opens up several enticing avenues for future research. Using compression ability as simpler guide, we are working on an approach 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; here prediction could suffer due to overfitting issues, whereas compression might continue see improvements from additional 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, the lack of reliably free-energy models for pseudoknotted RNA structures and the relative lack of high-fidelity training data would make compression ability of even greater value in the search for better predictions models.

References

[1] J. J. Cannone et al. The Comparative RNA Web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3, 2002.

[2] R. Consortium. RNAcentral 2021: secondary structure integration, improved sequence search and new member databases. *Nucleic Acids Research*, 49(D1):D212–D220, 2020.

[3] R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. *BMC bioinformatics*, 5(1):1–14, 2004.

[4] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[5] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13, 1970.

[6] J. Friemel. *Contraction-Based Compression of RNA Secondary Structures*. BSc dissertation, Universitat Bielefeld, 2020.

[7] L. Fu, Y. Cao, J. Wu, Q. Peng, Q. Nie, and X. Xie. UFold: fast and accurate RNA secondary structure prediction with deep learning. *Nucleic Acids Research*, 50(3):e14–e14, 2021.

[8] R. Giancarlo, D. Scaturro, and F. Utro. Textual data compression in computational biology: a synopsis. *Bioinformatics*, 25, 2009.

[9] J. Gorodkin and W. L. Ruzzo, editors. *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*. Humana Press, 2014.

[10] I. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Chemical monthly*, 125:167–168, 1994.

[11] I. Kalvari et al. Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic Acids Research*, 49(D1):D192–D200, 2020.

[12] J. C. Kieffer and E. hui Yang. Survey of grammar-based data structure compression. *IEEE BITS the Information Theory Magazine*, pages 1–12, 2022.

[13] A. Kolmogorov. On tables of random numbers. *Theoretical Computer Science*, 207, 1998.

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

[15] J. Lee et al. RNA design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences*, 111(6):2122–2127, 2014.

[16] Q. Liu, Y. Yang, C. Chen, J. Bu, Y. Zhang, and X. Ye. RNACompress: Grammar-based compression and informational complexity measurement of RNA secondary structure. *BMC bioinformatics*, 9(1):1–12, 2008.

[17] S. Maneth. Grammar-based compression. In *Encyclopedia of Big Data Technologies*, pages 1–8. Springer International Publishing, 2018.

[18] D. H. Mathews. How to benchmark rna secondary structure prediction accuracy. *Methods*, 162:60–67, 2019.

[19] H. Naganuma, D. Hendrian, R. Yoshinaka, A. Shinohara, and N. Kobayashi. Grammar compression with probabilistic context-free grammar. In *2020 Data Compression Conference (DCC)*. IEEE, 2020.

[20] M. E. Nebel and A. Scheid. Evaluation of a sophisticated SCFG design for RNA secondary structure prediction. *Theory in Biosciences*, 130(4):313–336, 2011.

[21] E. Rivas, R. Lang, and S. R. Eddy. A range of complex probabilistic models for RNA secondary structure prediction that includes the nearest-neighbor model and more. *RNA*, 18(2):193–212, 2011.

[22] K. Sato, M. Akiyama, and Y. Sakakibara. RNA secondary structure prediction using deep learning with thermodynamic integration. *Nature Communications*, 12(1), 2021.

[23] K. Sato and Y. Kato. Prediction of RNA secondary structure including pseudoknots for long sequences. *Briefings in Bioinformatics*, 23(1), 2021.

[24] A. 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.

[25] M. Trompper (digitalheir). Probabilistic earley parser, 2017.

[26] D. H. Turner and D. H. Mathews, editors. *RNA Structure Determination.* Springer New York, 2016.

[27] S. Wild. *An Earley-style parser for solving the RNA-RNA interaction problem.* BSc dissertation, University of Kaiserslautern, 2010.

[28] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30, 1987.

Appendix

A. Comparison with general purpose compressors

To compare the compression quality of our approach with state-of-the-art generic compressors, we use the *paq8l* tool (*http://mattmahoney.net/dc/#paq*). We compressed each individual RNA text file (with sequence in the first line and the secondary structure as dot-bracket string in the second line) in the `friemel-modified` dataset using `paq8l -8` (the setting for best compression) and summed up the file sizes of all compressed RNAs.

The uncompressed size of `friemel-modified` is 39 284 962 bytes and all RNAs combined have 19 357 501 bases (2 bytes per base, one for sequence, one for structure, plus a small amount of metadata overhead). paq8l compressed this to 9 146 548 bytes. Dividing this total compressed size (in bytes) by the total number of bases in the dataset yields an average of 3.78 bits per base. This is 70% more than the 2.211 bits that our compressed with G_S achieves (using a static rule-probability model).

It is not unexpected that a general purpose tool like paq8l does not come anywhere close to the compression of a domain-aware model; however, it is a bit surprising that paq8l uses substantially more space than the local first order empirical entropy: All first lines of the files have letters in $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$, and thus a local entropy of at most 2 bits per character. For the second line, we only have $\{\mathtt{(}, \mathtt{)}, \bullet\}$, and hence at most $\lg(3) \approx 1.58$ bits per character. Exploiting this local entropy would result in 3.58 bits per base.

B. Grammars

Here, we list the used grammars; we use the compact notation from [3], where we only give a and $\hat{a}$ as terminals instead of the pairs introduced in Section 2. The actual RNA grammars would have 4 rules for each rule with a single "a"; instead of $A \to \alpha a \beta$, we would actually have $A \to \alpha \left[\begin{smallmatrix}\mathtt{A}\\ \bullet\end{smallmatrix}\right] \beta$, $A \to \alpha \left[\begin{smallmatrix}\mathtt{C}\\ \bullet\end{smallmatrix}\right] \beta$, $A \to \alpha \left[\begin{smallmatrix}\mathtt{G}\\ \bullet\end{smallmatrix}\right] \beta$, and $A \to \alpha \left[\begin{smallmatrix}\mathtt{U}\\ \bullet\end{smallmatrix}\right] \beta$; similarly, each rules with a "$a\hat{a}$" pair actually stands for 6 rules resp. 16 rules if we allow non-canonical base pairs. For the stacking grammars, nonterminals $B^{a\hat{a}}$ are shorthand notation for 6 resp. 16 different nonterminals, which "remember" an enclosing pair. If there are several occurrences of the same $a\hat{a}$ pair within one rule, these must be replaced consistently (with the same bases in all occurrences).

Our parsers require grammars to be free of ε-rules, so we eliminated these in all grammars.

Moreover, the fast stochastic parser used for the prediction study requires a slightly more restrictive form: the grammars are not allowed to have left-recursive rules, and the nonterminals must be ordered, so that B comes before A whenever one can derive $B\alpha$ from A. We only use the unambiguous grammars $G_3, \ldots, G_8$ for the prediction study, so we directly give those grammars in the required form.

Grammar $G_{L'}$ (`LiuGrammar`)

The first grammar is $G_{L'}$ from Liu et al. [16] where we eliminate ε-rules.

$T \to a$ (4 rules)
$T \to aS\hat{a}$ (16 rules)
$S \to T \mid TS$

Grammar G_1 (`DowellGrammar1Bound`)

Next, $G_1, \ldots, G_8$ are the grammars taken from Dowell and Eddy [3].

$U \to a$
$B \to aS\hat{a}$
$C \to B \mid U$
$X \to UX \mid SX \mid U \mid S$
$S \to C \mid CX \mid US \mid USX$

Grammar G_2 (`DowellGrammar2Bound`)

$U \to a$
$P^{a\hat{a}} \to aP^{a\hat{a}}\hat{a} \mid S$
$S \to aP^{a\hat{a}}\hat{a} \mid U \mid US \mid SU \mid SS$

Grammar G_3 (`DowellGrammar3Bound`)

$U \to a$
$B \to aS\hat{a}$
$L \to B \mid UL$
$R \to U \mid UR$
$S \to B \mid UL \mid RU \mid LS \mid U$

Grammar G_4 (`DowellGrammar4Bound`)

$U \to a$
$B \to aS\hat{a}$
$C \to B \mid U$
$D \to C \mid CD$
$Q \to B \mid BD$
$S \to U \mid US \mid Q$

Grammar G_5 (`DowellGrammar5Bound`)

$U \to a$
$B \to aS\hat{a}$
$S \to U \mid B \mid US \mid BS$

Grammar G_6 (`DowellGrammar6Bound`)

$U \to a$
$B \to aM\hat{a}$
$T \to B \mid U$
$M \to B \mid TS \mid T$
$S \to TS \mid T$

An alternative version does not have the rule $M \to T$; that grammar then disallows hairpins of length one, i.e., '`( • )`'.

Grammar G_7 (`DowellGrammar7Bound`)

$U \to a$
$B \to aV^{a\hat{a}}\hat{a}$ (16 rules)
$B^{b\hat{b}} \to aV^{a\hat{a}}\hat{a}$ ($16 \cdot 16$ rules)
$L \to B \mid UL$
$M \to UM \mid U$
$T \to U \mid UL \mid MU \mid LS$
$V^{a\hat{a}} \to B^{a\hat{a}} \mid T$ ($16 \cdot 2$ rules)
$S \to B \mid UL \mid MU \mid U \mid LS$

Grammar G_8 (`DowellGrammar8Bound`)

$U \to a$
$B \to aV^{a\hat{a}}\hat{a}$
$B^{b\hat{b}} \to aV^{a\hat{a}}\hat{a}$
$C \to U \mid B$
$D \to C \mid CD$
$E \to B \mid BD$
$N \to U \mid E \mid US \mid EU \mid EB$
$V^{a\hat{a}} \to B^{a\hat{a}} \mid N$
$S \to U \mid E \mid US$

Grammar G_S (`SchulzGrammar`)

The grammar G_S is taken from [20]; see also [24, Def. A.1.2]; we have made the modifications described below to make the grammar more suitable for compression.

Since we have to expand every occurrence of $a\ \hat{a}$ on the right-hand side into 6 (or even 16) rules in our RNA grammars, we replaced "$aL\hat{a}$" in several right-hand sides with a nonterminal that expands to $aL\hat{a}$ (A when we start a new stem and the new nonterminal I when we continue after an interior loop or bulge). This reduces the number of parameters and hence the expressive power a bit, but will keep the grammar substantially smaller.

$p'_0 : S' \to S,$

$p'_1 : S \to A, \quad p'_2 : S \to AC, \quad p'_3 : S \to TA, \quad p'_4 : S \to TAC,$

$p'_5 : T \to A, \quad p'_6 : T \to AC, \quad p'_7 : T \to TA, \quad p'_8 : T \to TAC,$

$p'_9 : T \to C,$

$p'_{10} : C \to X^C, \quad p'_{11} : C \to CX^C,$

$p'_{12} : A \to aL\hat{a},$

$p'_{13} : L \to aL\hat{a}, \quad p'_{14} : L \to M, \quad p'_{15} : L \to P, \quad p'_{16} : L \to Q,$

$p'_{17} : L \to R, \quad p'_{18} : L \to F, \quad p'_{19} : L \to G,$

$p'_{20} : G \to Ia, \quad p'_{21} : G \to IX^BX^B, \quad p'_{22} : G \to IBX^BX^B,$

$p'_{23} : G \to aI \quad p'_{24} : G \to X^BX^BI \quad p'_{25} : G \to X^BX^BBI$

$p'_{26} : B \to X^B \quad p'_{27} : B \to BX^B$

$p'_{28} : F \to X^F X^F X^F \quad p'_{29} : F \to X^F X^F X^F X^F \quad p'_{30} : F \to X^F X^F X^F X^F X^F$
$p'_{31} : F \to X^F X^F X^F X^F X^F H$

$p'_{32} : H \to X^H \quad p'_{33} : H \to H X^H$

$p'_{34} : P \to aIa \quad p'_{35} : P \to X^I I X^I X^I \quad p'_{36} : P \to X^I X^I I X^I \quad p'_{37} : P \to X^I X^I I X^I X^I$

$p'_{38} : Q \to X^I X^I I X^I X^I X^I \quad p'_{39} : Q \to X^I X^I I K X^I X^I X^I \quad p'_{40} : Q \to X^I X^I X^I I X^I X^I$
$p'_{41} : Q \to X^I X^I X^I J I X^I X^I$

$p'_{42} : Q \to X^I X^I X^I I K X^I X^I \quad p'_{43} : Q \to X^I X^I X^I J I K X^I X^I$

$p'_{44} : R \to X^I I X^I X^I X^I \quad p'_{45} : R \to X^I I K X^I X^I X^I \quad p'_{46} : R \to X^I X^I X^I I X^I \quad p'_{47} : R \to X^I X^I X^I J I X^I$

$p'_{48} : J \to X^I \quad p'_{49} : J \to J X^I$

$p'_{50} : K \to X^I \quad p'_{51} : K \to K X^I$

$p'_{52} : M \to AA \quad p'_{53} : M \to UAA \quad p'_{54} : M \to AUA \quad p'_{55} : M \to AAN$

$p'_{56} : M \to UAUA \quad p'_{57} : M \to UAAN \quad p'_{58} : M \to AUAN \quad p'_{59} : M \to UAUAN$

$p'_{60} : N \to A \quad p'_{61} : N \to UA \quad p'_{62} : N \to AN \quad p'_{63} : N \to UAN$

$p'_{64} : N \to U$

$p'_{65} : U \to X^U \quad p'_{65} : U \to U X^U$

We add the following rules:

$F \to X^F \quad F \to X^F X^F$ (allow length 1 and 2 in hairpins)

$I \to aL\hat{a}$ (new nonterminal for use inside bulges/interior loops)

$S \to C$ (allow completely unpaired sequences)

Rules for all unpaired nonterminals:
$X^B \to a, \quad X^C \to a, \quad X^F \to a, \quad X^H \to a, \quad X^I \to a, \quad X^U \to a$

C. Further results

This appendix reports on some further results that were left out of the main text due to space constraints in the proceedings version.

C.1. Huffman coding vs. Arithmetic coding

We here compare the influence of the coding step on compression ratio in isolation. For that, we modify Liu et al.'s RNACompress [16] to use arithmetic coding instead of a Huffman code, leaving everything else unchanged, and compare the results.

We were not able to obtain the original implementation of RNACompress and the datasets from Liu et al. [16]. We hence re-implemented RNACompress, and used the friemel-modified dataset of 17 000 RNA samples originally taken from [1] instead of the dataset from [16]. Some of the RNAs in Friemel's dataset have non-canonical bonds (these are less stable secondary bonds). Since Liu et al. do not allow non-canonical bonds in their tool, we also removed these from Friemel's dataset, i.e., we replaced the open (and close) parenthesis for non-canonical bonds with unpaired bases • in the positions were non-canonical bonds appeared. Afterwards only the stable bonds (Watson-Crick and G–U wobble bonds) were left in all samples in the dataset, which we call friemel-modified.

Unsurprisingly, the arithmetic coding produced better compression results than Huffman coding, but the difference between the means is only 2.7%. Figure 5 shows the distribution of compressed size over the RNAs; while arithmetic coding has moderate impact on the mean compressed size, it helps a lot to bring down the right tail. The scatterplot in Figure 6 further shows that indeed, arithmetic coding (with this fixed static model) is doing better on almost all RNAs, and the effect is bigger for those RNAs that are compressed worse.

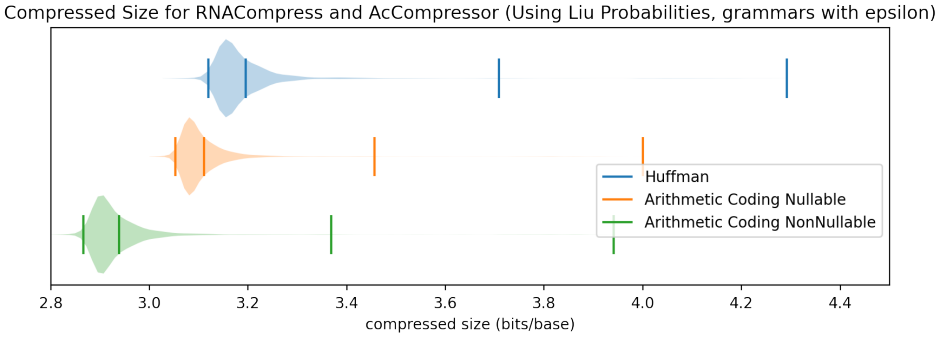

Figure 5: Compressed size in bits per base for RNACompress (original with Huffman coding) and RNACompress with arithmetic coding, and the RNACompress variant with the ε-rule-free grammar. The vertical bars show from, left to right, the 1% quantile, mean, 99% quantile, and maximum. The means are at 3.195 resp. 3.110 bits per base.

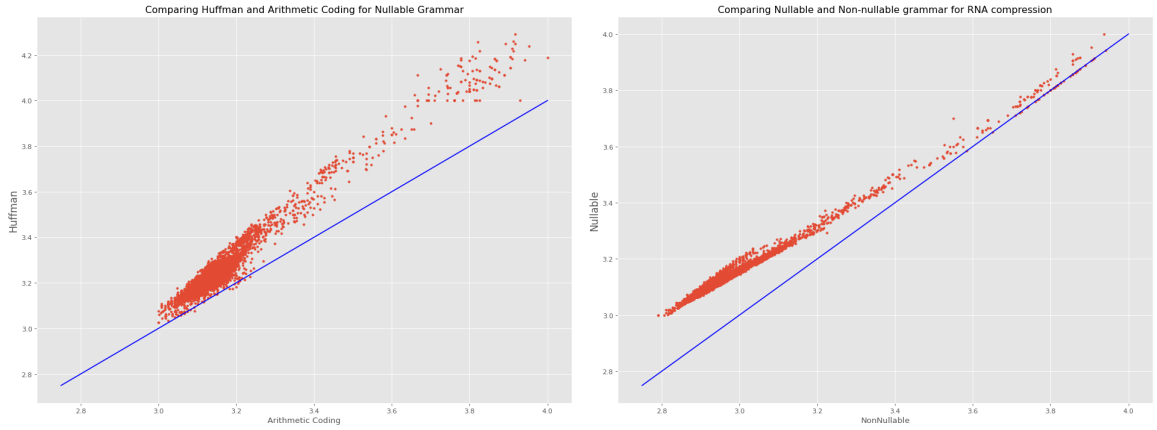

Figure 6: The same data as in Figure 5, but as scatter plots with one point per RNA.

C.2. Nullable Grammar vs. Non-Nullable Grammar

Liu et al. [16] originally use the following grammar (in the notation from Appendix B):

G_L
$L \to aS\hat{a} \mid a$
$S \to LS \mid \varepsilon$

For general parsers, ε-rules are often inconvenient; we therefore modified this grammar to $G_{L'}$ shown in Appendix B. This transformation makes the probabilistic model slightly richer and so will help compression, but it does not change the nature of the grammar; the structure of leftmost derivations of strings remain (almost) the same. (We here ignore the fact that the

empty string is no longer in the language of grammar $G_{L'}$, while it was derivable in G_L. For RNA compression, this is not relevant.) We manually implemented a parser for the original G_L grammar and compared the compression outcome. As Figure 5 shows, this very moderate enrichment of the probabilistic model has a larger impact than moving from Huffman to arithmetic coding. The scatter plot in Figure 6 (right) shows that again, we never do worse in $G_{L'}$ compared to G_L, but that this time, the biggest savings are happening for the (much larger number of) RNAs that are compressed *well*.