Towards Optimal Grammars for RNA Structures

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

January 29, 2024

Abstract

In past work (Onokpasa, Wild, Wong, DCC 2023), we showed that (a) for joint compression of RNA sequence and structure, stochastic context-free grammars are the best known compressors and (b) that grammars which have better compression ability also show better performance in *ab initio* structure prediction. Previous grammars were manually curated by human experts. In this work, we develop a framework for automatic and systematic search algorithms for stochastic grammars with better compression (and prediction) ability for RNA. We perform an exhaustive search of small grammars and identify grammars that surpass the performance of human-expert grammars.

1. Introduction

In this paper, we study the fundamental question of capturing typical folding structures of 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 stabilizing hydrogen bonds. The (well-nested) 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 *predicting* the secondary structure from a known RNA sequence (*ab initio*

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

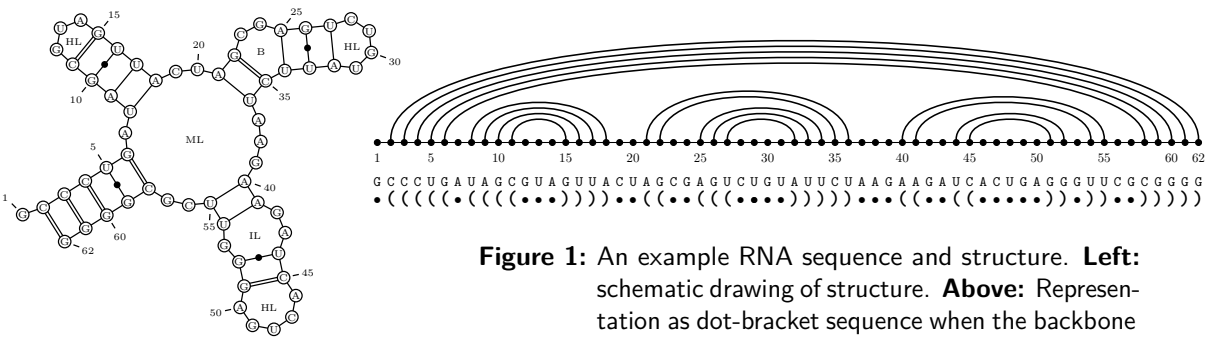

Figure 1: An example RNA sequence and structure. **Left:** schematic drawing of structure. **Above:** Representation as dot-bracket sequence when the backbone is "pulled straight".

RNA secondary-structure prediction) [2, 5, 14]. In our recent work [12], we showed that joint compression of RNA sequence and structure data can serve as a robust proxy for the prediction quality of different stochastic context-free grammar (SCFG) models of RNA secondary structures, a state-of-the-art formalism for *ab initio* structure prediction. We also showed that the RNA-specific SCFG-based compression outperforms by far the best general-purpose compressors such as *paq8l* (*http://mattmahoney.net/dc/#paq*) on RNA data.

In [12], we use SCFGs designed resp. collected by human domain experts [1, 8, 10]. **In this work, we move** from these isolated examples **towards a framework for systematic and automated search for *optimal* grammars.** By comparing SCFGs using their achieved compressed size instead of structure prediction performance, we eradicate intricacies and ongoing debates of how to measure the distance between predicted and true secondary structures [9]; our work thus paves the way for a well-defined open contest on finding SCFGs that best capture the essence of stable (minimum-free-energy) RNA structures.

On the technical side, we provide reference implementations of all key components needed for testing and evaluating SCFGs for RNA compression and prediction and we explore best practices for improving the efficiency of the search for good grammars.

Moreover, we report results from an initial exploration of the space of grammars. We find that the vast majority of grammars give rather poor compression, but a very small number achieve substantial compression. Among those, we could identify several new grammars that surpass the performance of similar-sized human expert grammars from the literature, indicating that further improvements are likely to be possible and that the intuitive grasp of RNA structures even among domain experts has limitations.

The rest of this paper is structured as follows. In Section 2, we introduce basic notation and summarize how SCFGs can be used to represent RNA. In Section 3, we introduce our new normal form for SCFGs for RNA compression. Our experimental setup and results are described in Section 4, and we conclude in Section 5. Datasets and code to produce figures and tables in this article are available online as supplementary material: *https://www.wild-inter.net/publications/onokpasa-wild-wong-2024* ; the code is available on GitHub: *https://github.com/evita35/better-grammars*.

2. Preliminaries

We give a few basic definitions on strings and grammars, before we introduce SCFGs as probabilistic models for RNAs.

We abbreviate $[a..b) = \{a, a+1, \ldots, b-1\}$. For a string $S \in \Sigma^n$, we denote by $S[i]$ for $i \in [0..n)$ the ith character of S (with 0-based indexing). $S[i..j)$ denotes the substring $S[i]S[i+1]\ldots S[j-1]$.

RNA as strings. 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`. (Pseudoknot-free) RNA (secondary) structures[1] can then be represented by the dot-bracket notation [6]: a well-nested string over $\{\bullet, (,)\}$ where a base pair is denoted by a matching pair of 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". Formally, they are strings over pairs of characters (see also [12]), 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.

[1] As is often done in the area, we do not consider structures with "pseudoknots" in this paper, i.e., we assume that all bonds are well nested.

Context-free Grammars. Dot-bracket strings can be generated by a context-free grammar (CFG). We use standard terminology for context-free grammars, see, e.g., [7]. All our derivations are leftmost derivations.

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 $(A, \gamma) \in R$ is written as $A \to \gamma$. We will use capital letters to denote nonterminals and lowercase letters for terminals.

A *leftmost application* of a rule $A \to \gamma$ in a sentential form $\alpha \in (N \cup T)^*$, provided A is the leftmost nonterminal in α, i.e., provided $\alpha = xA\beta$ for some $x \in T^*$ and $\beta \in (N \cup T)^*$, is the sentential form $\mathrm{lmd}_{A\to\gamma}(xA\beta) = x\gamma\beta$. A *leftmost derivation* in G is a sequence of rules $r_1, \ldots, r_t$ when the rules applied in sequence always lead to a well-defined leftmost derivation, i.e., with $\alpha_0 = S$ and $\alpha_i = \mathrm{lmd}_{r_i}(\alpha_{i-1})$. We extend $\mathrm{lmd}(\cdot)$ to sequences of rules, so write $\mathrm{lmd}_{r_1,\ldots,r_t}(\alpha_0) = \alpha_t$. All derivations in this work are leftmost derivations, we will therefore omit "leftmost" for brevity. We are mostly interested in *terminal* (leftmost) derivations, i.e., derivations with $\alpha_t = w \in T^*$. The language of G is the set of all $w \in T^*$ for which there is a (leftmost) derivation producing w. We identify a derivation for word w with the sequence of rules $r_1, \ldots, r_t$ used in the derivation; we write $\mathrm{lmd}_{r_1,\ldots,r_t}(S) = w$ (for S the start symbol) to indicate that rules $r_1, \ldots, r_t$, successively applied starting with S, produce w.

2.1. Stochastic Context-free Grammars

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 for every $A \in N$, $P : R \to [0, 1]$ induces a probability distribution over the set of rules with left-hand side A.

SCFG as probabilistic models. The probability of a derivation $r_1, \ldots, r_t$ (a sequence of rules from R) in the grammar G is the product of the probabilities of all used rules: $\mathbb{P}[r_1, \ldots, r_t] = \prod_{i=1}^{t} P(r_i)$. This corresponds to the probability of obtaining this derivation in the random process, where starting with S, in each time step, we choose a random replacement for the leftmost nonterminal A in the current sentential form. For that, we sample one of the rules $A \to \gamma$ with probabilities according to P and, conditionally on having left-hand side A, independent of the past choices.

We define the probability $\mathbb{P}[w]$ of a *word* w as the sum of the probabilities of its derivations.

$$\mathbb{P}[w] \;=\; \sum_{r_1,\ldots,r_t:\ \mathrm{lmd}_{r_1,\ldots,r_t}(S)=w} \mathbb{P}[r_1, \ldots, r_t] \,.$$

The sum is understood to range of all leftmost derivations (of arbitrary length). We also define the *Viterbi value* $V(w)$ of w, the probability of the most likely derivation of w:

$$V(w) \;=\; \max_{r_1,\ldots,r_t:\ \mathrm{lmd}_{r_1,\ldots,r_t}(S)=w} \mathbb{P}[r_1, \ldots, r_t] \,.$$

If G is unambiguous, there is only one derivation and we have $\mathbb{P}[w] = V(w)$.

Derivations as representations. If G is known (by convention or because it has been stored explicitly), a leftmost derivation $d = (r_1, \ldots, r_t)$ of a word $w = \mathrm{lmd}_d(S)$, is an encoding for w: the original word can always be reconstructed from d by (leftmost) application of the rules starting with S. This is the basis for our RNA encoding [12]. Note that the grammar is not required to be unambiguous for that, although it seems plausible that unambiguous grammars would yield more effective compression.

Probabilistic Parsing. Given a SCFG $G = (N, T, R, S)$ and a word $w \in T^*$ in the language of G, a probabilistic parser determines a *Viterbi derivation*, i.e., a most likely derivation for w: $\arg\max\{\mathbb{P}[d] : \mathrm{lmd}_d(S) = w\}$.

The theory of such parsers is well established and does not require G to have any specific normal form [4, 3]. However the resulting general algorithms are rather intricate; for example, chain rules can require to (symbolically) solve infinite summations for correct stochastic parsing [3]. But such grammars immediately allow an infinite number of leftmost derivations for one word; since our compression methods specify a single derivation, such ambiguity is counterproductive for compression.

A normal form such as *Chomsky normal form (CNF)* (all rules of type $A \to c$ or $A \to BC$) can simplify parsing dramatically; here even a stochastic parser remains a simple (bottom-up) dynamic-programming algorithm (stochastic CYK) [4].

Unfortunately, CNF is inconvenient for expressing the complementarity of paired bases in an RNA. We therefore start by proposing a new normal form for our grammars in Section 3.

2.2. SCFG-based Joint RNA Compression

Our formalism from [12] unifies grammars for encoding an RNA structure, joint RNA (sequence and structure), and predicting the structure from the sequence. We specify a grammar by giving the rules, e.g., $S \to \texttt{(}S\texttt{)} \mid \bullet \mid SS$; here '|' separates the right-hand sides of rules with the same nonterminal on the left. This represents 3 different types of grammars: (1) As is, it is a grammar for deriving/representing just the RNA structure (in dot-bracket notation). We call these the *secondary-structure grammar*. (2) We can expand the secondary-structure grammar to an *RNA grammar* by replacing all rules with $\bullet$ by 4 rules, where $\bullet$ is replaced by $\left[\begin{smallmatrix}\texttt{A}\\ \bullet\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{C}\\ \bullet\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{G}\\ \bullet\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}\texttt{U}\\ \bullet\end{smallmatrix}\right]$, respectively, and all rules with a pair of `(` and `)` by 6 rules, where `(`/`)` is replaced by $\left[\begin{smallmatrix}\texttt{A}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{U}\\ \texttt{)}\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{C}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{G}\\ \texttt{)}\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{G}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{C}\\ \texttt{)}\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{G}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{U}\\ \texttt{)}\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}\texttt{U}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{A}\\ \texttt{)}\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}\texttt{U}\\ \texttt{(}\end{smallmatrix}\right]/\left[\begin{smallmatrix}\texttt{G}\\ \texttt{)}\end{smallmatrix}\right]$, respectively. (Or, for handling datasets with non-canonical base pairs, even generate all 4×4 combinations of bases instead of just those 6). For brevity, we will use the short notation for the secondary-structure grammar, but we actually work with the expanded RNA grammar for compression.

(3) The third type, the *prediction grammar*, has the same structure as the RNA grammar, but when using it in a parser, we only look at the first entry of each pair. Thereby we can use it to compute a (most likely) derivation for a given RNA sequence; the pairs matched with the bases in the sequence automatically give us the RNA structure corresponding to this derivation. We can thus use a Viterbi parse to obtain the most likely structure for a given sequence using the prediction grammar.

Rule-probability models. When using an RNA grammar G for compressing an RNA sequence and structure pair w, we first determine a derivation for w in G. For the encoding of w given grammar G, we now need to specify probabilities for the rules. As in [12], we focus on two options here: (a) a *static* rule-probability model, where we determine probabilities from counting how often each rule is used on a training dataset, and (b) an *adaptive* rule-probability model, where we keep running counts of rule occurrences (starting at 1, i.e., a uniform prior) in the already encoded prefix of the derivation. For the actual binary encoding, we employ arithmetic coding [15] to store the next rule. Note that the left-hand side is always known; initially it is the start symbol and then, inductively, the leftmost nonterminal in the current sentential form. More details and a worked example are given in [12, §3].

3. Stochastic RNA Normal Form for Grammars

We consider SCFGs in a specific normal form, the *Stochastic RNA Form (SRF)*. It takes inspiration from existing expert SCFG designs used for RNA structure prediction [1, 8, 10]. Our normal form assumes a total order on the nonterminals. To simplify notation in this section, assume without loss of generality that $N = \{A_1, \ldots, A_k\}$; we define $A_i < A_j$ if and only if $i < j$.

A SCFG $G = (N, T, R, A_k, P)$ is in *Stochastic RNA Form* if each of its rules has one of the following forms:

(i) $A_i \to A_j A_l$	(ii) $A_i \to \bullet$	(iii) $A_i \to (A_j)$	(iv) $A_i \to A_j$ and $j < i$

The ordering constraint on type (iv) rules ensures that there is a finite number of derivations for every word and that we can retain a total order on subproblems in parsing (see below). Apart from making parsing more efficient, the Stochastic RNA Form makes it trivial to ensure that grammars produce valid RNA structures. It therefore massively reduces the search space in our exhaustive search by excluding many invalid grammars.

The Stochastic RNA Form is chosen to be as expressive as possible, fixing only features that all stable RNA structures share. One tacit assumption we impose on RNA structures is that they have no empty hairpin loops, i.e., no subword "`( )`". Such a bond is indeed impossible due to physical limitations; it does get reported in databases, but rarely so (and likely erroneously).

Given a grammar in SRF, we can adapt the probabilistic CYK parser [4] to our grammars as follows: For that, we denote by $V_A(w[i..j))$, for $A \in N$ and $0 \leq i < j \leq n = |w|$ the probability of the most likely derivation of $w[i..j)$ when starting with A. We then have $V(w) = V_S(w[0..n))$ for S the start symbol of G and obtain the recursive equations following the allowed rule types:

$$V_A(w[i..i+1)) \;=\; \begin{cases} P(A \to w[i]) & \text{if } A \to w[i] \in R \text{ (ii)} \\ 0 & \text{otherwise} \end{cases}$$

$$V_A(w[i..j)) \;=\; \max \begin{cases} \max\limits_{\substack{k \in [i+1..j) \\ A \to BC \in R}} P(A \to BC)\, V_B(w[i..k))\, V_C(w[k..j)) & \text{(i)} \\ \max\limits_{A \to w[i] B w[j-1] \in R} P(A \to w[i]\, B\, w[j-1])\, V_B(w[i+1..j-1)) & \text{(iii)} \\ \max\limits_{A \to B \in R} P(A \to B)\, V_B(w[i..j)) & \text{(iv)} \end{cases}$$

The ordering constraint on type (iv) rules implies that the subproblems $V_A(w[i..j))$ can be totally ordered by (ℓ, A) for $\ell = j - i$ the length of the produced subword and A the nonterminal, allowing for an efficient bottom-up dynamic-programming parser.

4. Methodology and Results

4.1. Exhaustive exploration

We first report on the results of an exhaustive exploration of all small stochastic context-free grammars in Stochastic RNA Form. The goal is to shed light on the following questions: *Does RNA favor a specific shape of grammars? If so, which? How different is the compression ability of different grammars of the same size? Are the human-expert chosen grammars best possible for their size?*

For this, we implemented an exhaustive generation algorithm that can iterate over all possible grammars in Stochastic RNA Form by constructing for a given number of nonterminals

#NTs	#rules	# SRF grammars	#parsing gr.	(%)	best bits-per-base
1	3	1	1	100%	3.6241
2	1	15	0	0%	—
2	2	105	0	0%	—
2	3	455	1	0.2%	3.6890
2	4	1365	28	2%	3.1424
2	5	3003	201	7%	2.9969
2	6	5005	783	16%	2.9762
2	7	6435	1831	28%	2.9927
2	8	6435	2801	44%	3.0088
2	9	5005	2953	59%	3.0516
2	10	3003	2198	73%	3.0668
2	11	1365	1158	85%	3.0856
2	12	455	424	93%	3.1229
2	13	105	103	98%	3.3456
2	14	15	15	100%	3.5364
2	15	1	1	100%	3.8076
3	1	42	0	0%	—
3	2	861	0	0%	—
3	3	11 480	1	0.00001%	3.6891
3	4	111 930	71	0.00001%	3.1424
3	5	850 668	2015	0.002%	2.9866
3	6	5 245 786	33 170	0.006%	2.6620
3	7	26 978 328	377 522	0.013%	2.5495
3	8	118 030 185	3 212 691	0.027%	2.5582

Table 1: Overview of the overall number of grammars of certain sizes and the number of grammars that are able to parse all RNA in our benchmark dataset. The best bits-per-base gives an indicative compression performance (adaptive rule-prob model on the 10% sample of "benchmark"). Note that the number of possible rules for 2 nonterminals is 15 and for 3 nonterminals is 42.

all possible rules in SRF. Then we can iterate over all possible subsets, or all subsets of a given size to generate all possible SRF grammar with these parameters. Note that for k nonterminals, the total number of possible SRF rules is k^3 type (i) rules, k type (ii) rules, k^2 type (iii) rules and $\binom{k}{2}$ type (iv) rules.

The number of grammars is large ($> 2^{k^3}$) even for moderate k; however many grammars do not even allow to parse all RNA structures (see Table 1) and can be discarded quickly. For that, we use a tiny dataset "parsable" of short RNA structures; any grammar that fails to parse this dataset is skipped. For the remaining grammars, we determine the normalized compressed size, i.e., the number of bits in the compressed representation divided by the number of bases of the RNA; both using adaptive and static rule-probability models. The dataset we use is the "benchmark" dataset from Dowell and Eddy [1]. As an efficient first filter we reduce it to a randomly chosen 10% subsample; we determined in preliminary experiments that this predicts the bits-per-base value on the entire dataset to within ±1%. We hence determine the best grammars from this 10% subsample and then evaluate the most successful grammars on the full benchmark dataset.

4.2. Distribution of compression ability

Using each possible small grammar (all grammars with 2 nonterminals, and all grammars with 3 nonterminals and at most 6 rules) to compress the 10% sample of the benchmark dataset,

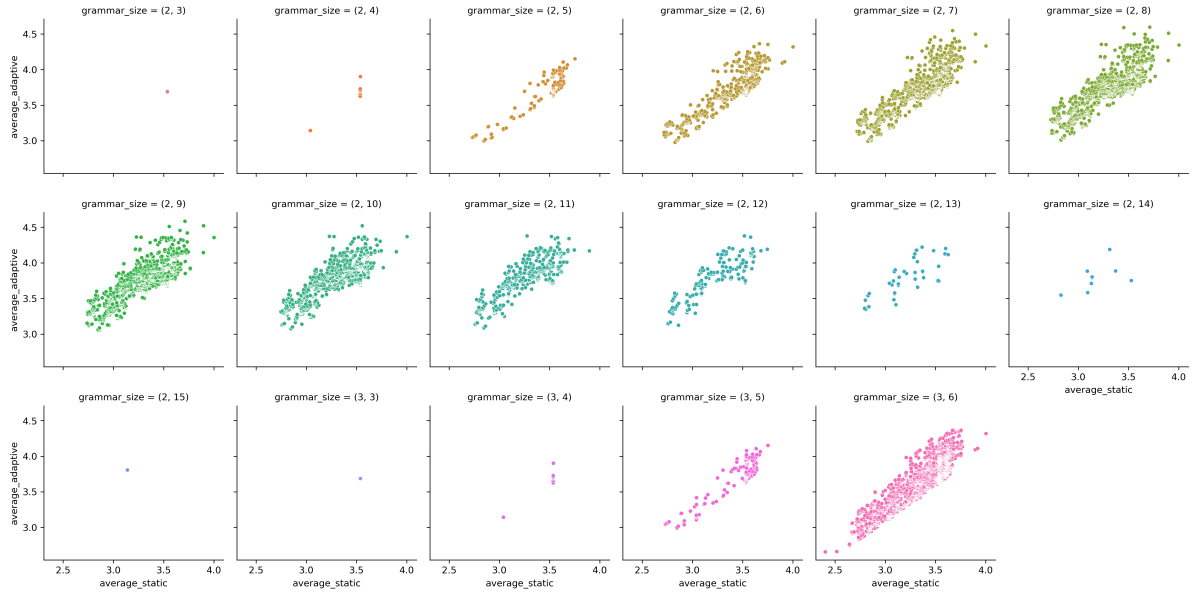

Figure 2: Normalized average compressed size (in bits per base) for all grammars of the given size (#NTs (Nonterminals), #rules) on 10% sample of the "benchmark" dataset from [1]. Each dot is one grammar; the x-coordinate is using the static rule-probability model, with rule counts on the same dataset; the y-coordinate uses the adaptive rule-probability model.

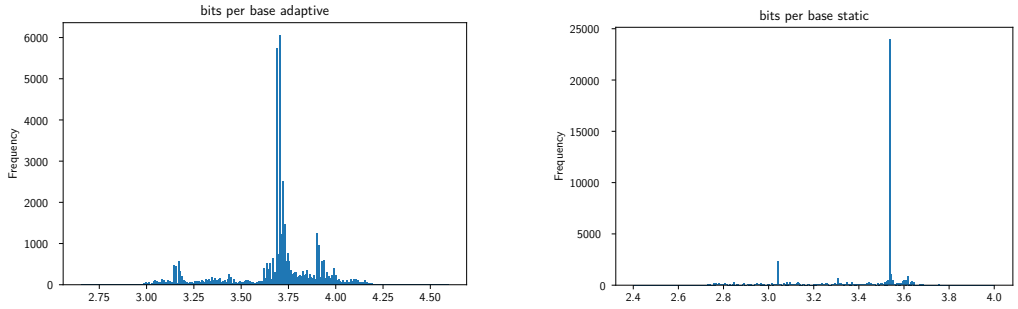

Figure 3: Histogram of the normalized average compressed size (in bits per base) for all grammars from Figure 2 on the 10% subsample of the benchmark dataset from Dowell and Eddy [1] using the adaptive (left) resp. static (right) rule-probability model.

$G^*_{2,6}$
$A_0 \to \bullet$
$A_0 \to A_1 A_0$
$A_0 \to A_1 A_1$
$A_1 \to A_0 A_1$
$A_1 \to (A_1)$
$A_1 \to A_0$

$G^*_{2,5}$
$A_0 \to \bullet$
$A_1 \to A_1 A_1$
$A_1 \to A_0 A_1$
$A_1 \to (A1)$
$A_1 \to \bullet$

$G^*_{3,6}$
$A_0 \to \bullet$
$A_0 \to (A_2)$
$A_1 \to A_0$
$A_2 \to (A_2)$
$A_2 \to A_1$

$G^*_{3,7}$
$A_0 \to \bullet$
$A_0 \to (A_0)$
$A_0 \to A_1 A_2$
$A_1 \to \bullet$
$A_1 \to A_0$
$A_2 \to A_1$
$A_2 \to A_1 A_2$

$G^\dagger_{6,10}$
$A_0 \to (A_5)$
$A_1 \to \bullet$
$A_1 \to A_0$
$A_2 \to A_3 A_5$
$A_2 \to A_4 A_3$
$A_4 \to A_1$
$A_4 \to A_4 A_1$

$A_5 \to A_0$
$A_5 \to A_0 A_2$
$A_5 \to A_4$

Figure 4: Newly identified grammars; $G^*_{k,r}$ is the best grammar with k NTs (Nonterminals) and r rules from exhaustive search; $G^\dagger_{k,r}$ is the best grammar we found with random search.

we obtain the normalized compressed size (in bits per base) using the adaptive and static rule-probability models. Figure 2 shows scatter plots of these results, split by grammar size. There is a clear correlation of the two measures, meaning that grammars mostly live on a single scale from better to worse compression approximately reflecting both models.

Moreover, the vast majority of grammars give rather poor compression. This is even more visible in Figure 3 which shows the distribution of bits-per-base for all grammars up to 3 nonterminals and 6 rules (same data as in Figure 2). We expect the normalized compressed size to be at least 2 bit per base, since the primary structure of RNA is not known to be substantially compressible (and thus needs roughly 2 bits per character), and we need some additional information to encode the structure on top of that. The vast majority of grammars just realize a compressed size around $\lg(4 \cdot 3) \approx 3.58$, which corresponds to storing each of the pairs of terminals (4 bases, 3 structure symbols) independently with uniform likelihood. Note that this is (approximately) the compression achieved by the trivial grammar with a single nonterminal and the three rules $A \to \texttt{(}A\texttt{)} \mid \bullet \mid AA$. It seems hence indeed the case that most grammars are not able to pick up the structure of RNA at all, and only a very small number of grammars achieve substantially better compression than almost all other grammars. Figure 4 shows some of these.

4.3. Random search

Despite an increasing fraction of grammars not parsing all RNA and most grammars only giving trivial compression, compression quality does increase with (moderate) increase in grammar size (see Table 1 and Figure 2). It is therefore desirable to be able to search among larger grammars than accessible via exhaustive exploration. As a simple first step towards that, we implemented a random grammar explorer that repeatedly generates random grammars (from the grammars of a given size) and keeps track of the top m grammars ever encountered. Again, to obtain any meaningful efficiency, we first check grammars for the ability to parse a tiny dataset of artificial RNA. Among those who parse that correctly, we evaluation their compression ability on a small dataset of short RNA and those who perform on this at least as good as the worst of the current top m grammars, we evaluate on the benchmark dataset. The top m grammars are always chosen based on the benchmark dataset. (We confirmed on the small grammars where we have exhausted all grammars that this process indeed identified the overall best performing grammars.) We ran this random exploration for grammars with 3–10 nonterminals and different numbers of rules, together exploring several billions of grammars. The best grammar found in that process was $G^{\dagger}_{6,10}$, see Figure 4 (right). Note that $G^{\dagger}_{6,10}$ contains 2 nonterminals that are dead ends for any derivation: A_3 has no rules, and hence cannot ever be replaced. A_2 only has rules involving A_3 and hence can likewise never be resolved to terminals. Removing those rules (and nonterminals) gives ${G^{\dagger}_{6,10}}'$.

4.4. Comparison with expert-curated grammars

We compare the results from newly found grammars with the expert-curated RNA grammars collected in the literature in Table 2; the grammars can be found in Figure 4 resp. the appendix of [12]. Although by a narrow margin, the best grammar for adaptive rule probabilities known is our new grammar ${G^{\dagger}_{6,10}}'$, clearly demonstrating that human-expert grammars are not necessarily best possible!

Two expert grammars are in the range where exhaustive exploration is still feasible and both are *not* the best possible grammar, even in their (flyweight) category: $G_{L'}$ by Liu et al. [8] is surpassed by a fair margin (2.95 vs 3.12 bits per base) by other grammars with 2 nonterminals.

Grammar	adaptive	static	#NTs	#rules	grammar size
grammar-1NT	3.6241	3.4731	1	3	3
$G_{L'}$ (Liu et al.)	3.1229	3.0201	2	4	18
$G^*_{2,5}$ (new)	2.9699	2.8200	2	5	19
$G^*_{2,6}$ (new)	2.9494	2.8011	2	6	20
G_5 (Dowell, Eddy)	2.8368	2.7423	3	6	32
G_3 (Dowell, Eddy)	2.5804	2.4549	5	11	69
G_6 (Dowell, Eddy)	2.4957	2.3687	5	9	61
G_4 (Dowell, Eddy)	2.7138	2.5974	6	11	78
G_1 (Dowell, Eddy)	3.0779	2.8956	6	13	87
G_2 (Dowell, Eddy)	2.9723	2.5525	18	296	1742
G_7 (Dowell, Eddy)	2.6343	2.3333	38	321	2883
G_8 (Dowell, Eddy)	2.7213	2.4561	39	322	2926
G_S (Nebel, Scheid)	2.5045	2.2876	108	244	3396
$G^*_{3,6}$ (new)	2.6329	2.3797	3	6	32
$G^*_{3,7}$ (new)	2.5287	2.3878	3	7	34
$G^\dagger_{6,10}$ (new)	2.5091	2.3835	6	10	73
$G^\dagger_{6,10}{}'$ (new)	2.4902	2.3835	4	7	45

Table 2: Normalized compressed size (bits per base) for some newly found small grammars and the grammars from [1, 8, 10]. All results are for the "benchmark" dataset from [1]. "grammar size" is the #bits needed to encode the grammar using the following scheme: encode k (#NTs) in Elias gamma code, then r (# rules) in binary and finally the size-r subset of rules using an enumeration of all size-r subsets.

It is best possible, though if we insist on exactly 4 rules. For G_5 from Dowell and Eddy [1], we find a better grammar of the same size, again with a substantial improvement of compression ability.

5. Conclusion

We formulated the competition for the best joint RNA compression grammar and gave first improvements on the best known grammars. Unfortunately, the combinatorial explosion of possible grammars and the fact that most grammars do not compress RNA meaningfully turns the problem into the search for a needle in a haystack. Further progress in future work will likely have to come from heuristics, potentially including learning systems.

The fact that the expert grammars are (in several ways) not best possible indicates that there are still unexplored patterns in RNA structures to be understood. The contest to find the best grammar for adaptive rule probabilities in particular has the potential to deepen our understanding of RNA structures and might lead to interesting compression methods on the path towards optimal grammars for RNA compression and prediction.

References

[1] 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.

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

[3] J. Goodman. *Parsing Inside-Out*. PhD thesis, 1998.

[4] J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999.

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

[6] 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.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 2nd edition, 2001.

[8] 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.

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

[10] 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.

[11] J. Oncina. The cocke-younger-kasami algorithm for cyclic strings. *Proceedings of 13th International Conference on Pattern Recognition*, 1996.

[12] E. Onokpasa, S. Wild, and P. W. H. Wong. RNA secondary structures: from ab initio prediction to better compression, and back. In *Data Compression Conference (DCC)*, pages 278–287, 2023.

[13] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, Boston, 2006.

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

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

Appendix

A. New Grammars

$G^*_{2,6}$

$A_0 \to \bullet$
$A_0 \to A_1 A_0$
$A_0 \to A_1 A_1$
$A_1 \to A_0 A_1$
$A_1 \to (A_1)$
$A_1 \to A_0$

$G^*_{2,5}$

$A_0 \to \bullet$
$A_1 \to A_1 A_1$
$A_1 \to A_0 A_1$
$A_1 \to (A1)$
$A_1 \to \bullet$

$G^*_{3,6}$

$A_0 \to \bullet$
$A_0 \to (A_2)$
$A_1 \to A_0$
$A_2 \to (A_2)$
$A_2 \to A_1$

$G^*_{3,7}$

$A_0 \to \bullet$
$A_0 \to (A_0)$
$A_0 \to A_1 A_2$
$A_1 \to \bullet$
$A_1 \to A_0$
$A_2 \to A_1$
$A_2 \to A_1 A_2$

$G^\dagger_{6,10}$

$A_0 \to (A_5)$
$A_1 \to \bullet$
$A_1 \to A_0$
$A_2 \to A_3 A_5$
$A_2 \to A_4 A_3$
$A_4 \to A_1$
$A_4 \to A_4 A_1$
$A_5 \to A_0$

$A_5 \to A_0 A_2$
$A_5 \to A_4$