DIFFERENTIABLE PARTITION FUNCTION CALCULATION FOR RNA

Marco Matthies*
Centre for Bioinformatics
University of Hamburg
Hamburg, Germany

Ryan Krueger*
Department of Applied Mathematics
Harvard University
Cambridge, MA

Andrew Torda
Centre for Bioinformatics
University of Hamburg
Hamburg, Germany

Max Ward
Department of Computer Science and Software Engineering
The University of Western Australia
Perth, Australia
max.ward@uwa.edu.au

ABSTRACT

Ribonucleic acid (RNA) is an essential molecule in a wide range of biological functions. In 1990, McCaskill introduced a dynamic programming algorithm for computing the partition function of an RNA sequence. This forward model is widely used for understanding the thermodynamic properties of a given RNA. In this work, we introduce a generalization of McCaskill's algorithm that is well-defined over continuous inputs and is differentiable. This allows us to tackle the inverse folding problem—designing a sequence with desired equilibrium thermodynamic properties—directly using gradient optimization. This has applications to creating RNA-based drugs such as mRNA vaccines. Furthermore, it allows McCaskill's foundational algorithm to be incorporated into machine learning pipelines directly since we have made it end-to-end differentiable. This work highlights how principles from differentiable programming can be translated to existing physical models to develop powerful tools for machine learning. We provide a concrete example by implementing an effective and interpretable RNA design algorithm.

Keywords Inverse folding · differentiable programming · statistical mechanics · RNA

1 Introduction

Ribonucleic acid (RNA) is fundamentally important to biology [1] and carries out many of the processes of life alongside DNA and protein such as transcription and translation [2], catalyzing reactions [3], gene regulation [4], and much more. Many of these functions require specific structures [5], so effort has gone into computational methods for predicting structures [6], which are now widely used [7, 8]. These are based on the nearest neighbor model [9, 10, 11, 12], which was found empirically through numerous experiments and predicts the thermodynamics of RNA secondary structures. A fundamental part of these methods is the computation of the *partition function*.

There exists a dynamic programming algorithm for computing the partition function of an RNA sequence under the nearest neighbor model [13]. This algorithm is widely used for computing various statistical properties of an RNA in thermodynamic equilibrium such as individual base-pair probabilities, the probability of folding into a given secondary structure, and heat capacity [14].

This approach is in stark contrast to recently developed deep learning methods for studying proteins (e.g., AlphaFold) [15, 16]. First, deep learning models of protein structure and function do not typically operate directly on thermodynamic quantities or explicitly predict ensemble statistics. Second, unlike deep learning models, the calculation of the partition

*These authors contributed equally to this work

function of an RNA sequence is only well-defined for a discrete sequence and therefore is not differentiable. This limits downstream applications of the model. For example, end-to-end differentiable models of protein structure enable new methods for inverse folding. The inverse folding problem (often also called the design problem) inverts the structure prediction problem. Instead of predicting the structure given the sequence, we must find a sequence that folds into a target structure. A differentiable folding algorithm can be inverted by starting from any sequence and optimizing the sequence using gradients propagated through the forward model [17, 18]. An end-to-end differentiable RNA partition function would enable this method for inverse RNA folding.

An end-to-end differentiable RNA partition function is important for other tasks too. It extends a classical and widely used algorithm, which can now be incorporated into machine learning pipelines, for example as a loss function in deep learning. In addition, it can be used to optimize the parameters of the nearest neighbor model in a similar way to inverse folding, which is an important but challenging task [19].

In this work, we adapt the classical dynamic programming algorithm for computing an RNA's partition function by making it end-to-end differentiable. To achieve this, the algorithm must (i) operate on a continuous input and (ii) comprise of differentiable operations. To satisfy (i), the input to our algorithm is a *probabilistic* sequence (i.e. a probability distribution of sequences) and we introduce new thermodynamic quantities for describing the partition function of such a sequence. Crucially, we then modify the existing algorithm to compute this partition function by calculating intermediate Boltzmann-weighted sums that are conditioned on the base identities of the subsequence in question. We satisfy (ii) by implementing our method in JAX [20].

To demonstrate the effectiveness of our method, we use it for inverse folding. The partition function can be used to calculate the probability of a structure. The probability of a structure is known to be a good guide for inverse folding [21]. Gradients with respect to the sequence distribution can be computed, which lets us optimize the probability by gradient descent. Success in increasing the probability indicates that our method is correct, and that the gradients we calculate are meaningful.

2 Methods

In this section, we present the definition of a partition function for a probability distribution of sequences and our algorithm for computing this quantity in a differentiable fashion. We then present a small set of inverse folding experiments performed to test our approach.

2.1 Partition Functions for Probabilistic Sequences

The partition function of a system is defined as

$$Z = \sum_{s \in S} e^{-\beta E_s} \tag{1}$$

where S is the set of all microstates, β is the inverse temperature, and E_s is the total energy of the system in microstate s.

In the context of RNA folding, the set of possible microstates S for a given sequence is defined as all possible secondary structures under the model. A secondary structure is typically described as a properly nested set of parentheses where indexes are permitted to be unpaired [6, 13]. Formally, for a sequence of length n, a secondary structure is a set of pairs $(i, j) \in \{1 \dots n\}^2$ such that $i < j$ and there are no two pairs (i, j) and (k, l) where $i < k < j < l$ and there are no pairs with shared indexes.

For a fixed RNA sequence, the partition function described by Equation 1 corresponds to the sum of Boltzmann weights over all secondary structures. We refer to this fixed-sequence, variable-structure partition function as the *structure partition function*, denoted Z_{struct}. For a sequence q, the structure partition function is defined as

$$Z_{\text{struct},q} = \sum_{s \in S} e^{-\beta E_{q,s}} \tag{2}$$

where $E_{q,s}$ denotes the energy of sequence q with structure s under the nearest neighbor model. This is what McCaskill's algorithm computes [13].

In this work, we introduce two new thermodynamic quantities: the *sequence partition function* and the *structure-sequence partition function* denoted Z_{seq} and Z_{ss}, respectively. Consider a probability distribution of sequences of

length n,

$$x \in [0,1]^{4\times n}, \text{ where } \sum_{i=1}^{4} x_{i,j} = 1 \tag{3}$$

where the value at each of the four indices at a given position j corresponds to the probability of the the nucleotide at that position having identity A, C, G, or U. For a given discrete sequence $q \in Q$ where Q is the set of all 4^n sequences of length n, the probability of sampling q from x is the product of all corresponding base identities at each position. Intuitively, this can be thought of as a probabilistic, or "fuzzy" sequence. Crucially, a probabilistic sequence is continuous and can be analysed using a gradient.

For a given secondary structure s, we define the sequence partition function as the weighted sum of Boltzmann weights over all sequences $q \in Q$ assuming structure s:

$$Z_{\text{seq},s}(x) = \sum_{q\in Q} x_q \cdot e^{-\beta E_{q,s}} \tag{4}$$

where x_q is the probability of sampling sequence q from x. Similarly, we define the structure-sequence partition function:

$$Z_{\text{ss}}(x) = \sum_{s\in S}\sum_{q\in Q} x_q \cdot e^{-\beta E_{q,s}} \tag{5}$$

$$= \sum_{s\in S} Z_{\text{seq},s}(x) \tag{6}$$

See Section 2.3 for an example of how this partition function can be interpreted in the context of thermodynamic properties.

2.2 Calculation of Structure-Sequence Partition Function

We modify McCaskill's algorithm [13] so that it is differentiable. This involves two major conceptual changes. First, the input must be made continuous, and is therefore described by a probability distribution (see Equation 3). We describe new recursions to work with a probability distribution. Second, all the operations need to be differentiable. We re-implemented the energy functions to be differentiable, and the recursions to contain only differentiable operations. This was achieved using JAX [20].

We implemented the nearest neighbor energy model with three main energy functions [9]. The model gives each loop a free energy contribution and the sum of contributions is the free energy of the structure. A loop is a region enclosed by base pairs and each kind of loop has a different energy function. See Figure 1 for an example. We modify the energy functions to work with explicit base identities to support a probabilistic sequence.

The ONE-LOOP(b_i, b_j, i, j) function gives the free energy contribution of a loop closed by a single pair i, j, also called a hairpin loop. The parameters b_i and b_j refer to the bases (A, C, G, or U) at positions i and j.

Next, we have the TWO-LOOP$(b_i, b_j, b_k, b_l, i, j, k, l)$ function, which gives the free energy of a loop formed by the two pairs i, j and k, l. This captures stacks, bulge, and internal loops in the nearest neighbor model. The parameters b_i, b_j, b_k, b_l describe the base identities of the corresponding locations.

Finally, the model uses an affine approximation of loops closed by 3 or more pairs. These are called multiloops. A multiloop closed by P pairs and containing U unpaired nucleotides is assigned a free energy contribution of $M_i + M_p \cdot P + M_u \cdot U$.

We have left out several complexities in the energy model for brevity. These include terminal mismatches, dangling ends, helix end penalties, special hairpins, and special internal loops. These amount to tedious case-work and the core ideas of the algorithm are clearer without them. Details can be found in our code.

For convenience, we define a Boltzmann function,

$$\mathcal{B}(E) = e^{-\beta E} \tag{7}$$

which scales a given energy term E.

The structure-sequence partition function is computed by dynamic programming described by three functions. The first computes the external loop. Define $\mathcal{E}(i)$ as the partition function of the fragment of nucleotide $[i, n)$ assuming the

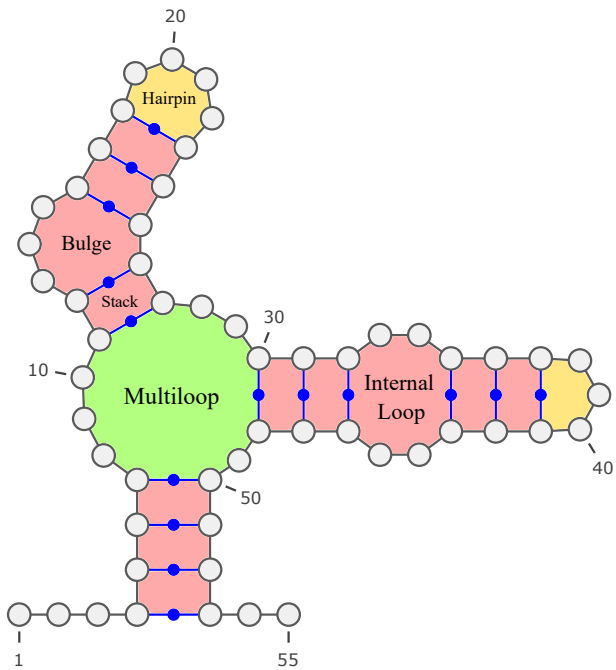

Figure 1: An example RNA structure showing the nearest neighbor model. One-loops are depicted in yellow, two-loops in red, and multi-loops in green. Examples of specific nearest neighbor model motifs are given. This image was generated with the help of VARNA [22].

nucleotides are indexed from 0 to $n-1$. The base case is $\mathcal{E}(n) = 1$. Define $\mathcal{P}(b_i, b_j, i, j)$ as the partition function of the fragment $[i, j]$ conditioned on i being base b_i and j being base b_j and assuming i, j form a pair. The base cases are $\mathcal{P}(b_i, b_j, i, j) = 0$ when $i \geq j$. Define $\mathcal{M}(p, i, j)$ as the partition function of the fragment $[i, j]$ assuming it is in a multiloop, and the fragment contains at least p pairs closing the multiloop. The base cases are $\mathcal{M}(0, i, j) = 1$ when $i > j$ and $\mathcal{M}(p > 0, i, j) = 0$ also when $i > j$.

We can compute $\mathcal{E}$ by summing over the cases where i is not paired, and where i is paired.

$$\mathcal{E}(i) = \mathcal{E}(i+1) + \sum_{\substack{b_i, b_j \in A, \\ i<j<n}} \mathcal{P}(b_i, b_j, i, j) \cdot x_{b_i,i} \cdot x_{b_j,j} \tag{8}$$

Recall that x is the probabilistic sequence and $x_{b_i,i}$ denotes the probability that the sequence is base b_i at location i. These probabilities must be multiplied through. Note that there are four combinations each for b_i and b_j that we sum over corresponding to the alphabet of nucleotides $A = \{\text{A, C, G, and U}\}$. In combination, we also sum over all valid right endpoints j for a pair i, j.

Observe that the partition function itself is defined by $\mathcal{E}(0)$.

$$\mathcal{P}(b_i, b_j, i, j) = \sum \begin{cases} \mathcal{B}(\textsc{one-loop}(b_i, b_j, i, j)) \\ \mathcal{P}(b_k, b_l, k, l) \cdot x_{b_k,k} \cdot x_{b_l,l} \\ \quad \cdot \mathcal{B}(\textsc{two-loop}(b_i, b_j, b_k, b_l, i, j, k, l)) \\ \quad \forall b_k, b_l \in A, i < k < l < j \\ \mathcal{M}(2, i+1, j-1) \cdot \mathcal{B}(M_i) \cdot \mathcal{B}(M_p) \end{cases} \tag{9}$$

The recursions for $\mathcal{P}$ sum over three types of case. The first is the one-loop closed by just the pair i, j. The second is two-loops closed by the two pairs, i, j and some k, l that is fully contained inside i, j. Note that in the second case we must consider all options for k, l and the bases b_k, b_l at those locations. Furthermore, that we only multiply by the probabilities $x_{b_k,k}$ and $x_{b_l,l}$ since $\mathcal{P}(b_i, b_j, i, j)$ is conditional on b_i and b_j. The third and final case is for multiloops

closed by i, j. Observe that at least 2 additional pairs are required to make a multiloop, which is why $\mathcal{M}(2, i+1, j-1)$ is used.

$$\mathcal{M}(p,i,j) = \sum \begin{cases} \mathcal{M}(p,i+1,j) \cdot \mathcal{B}(M_u) \\ \mathcal{P}(b_i,b_k,i,k) \cdot \mathcal{M}(\max(0,p-1),k+1,j) \\ \quad \cdot \mathcal{B}(M_p) \cdot x_{b_i,i} \cdot x_{b_k,k} \\ \quad \forall b_i, b_k \in A, i < k \leq j \end{cases} \tag{10}$$

The recursions for $\mathcal{M}$ sum over two cases similar to those for $\mathcal{E}$. The first is when position i is not paired. The second is when i is in a pair i, k. In the second case, we must sum over all options for b_i, b_k and k.

Our implementation targets parity with ViennaRNA 2.5.0 [7]. ViennaRNA is the most widely used RNA folding software pacakge. By default, ViennaRNA double counts dangling ends and terminal mismatches to approximate coaxial stacking. We leave out a detailed discussion of how we implemented this, since it involves subtle casework. However, it is worth pointing out that this significantly complicates the recursions for multiloops, $\mathcal{M}$, and leads to high memory usage. Ironically, the behaviour of ViennaRNA is an approximation designed to make their recursions simpler, but when they are redesigned to the structure-sequence partition function, it makes them more complicated. We expect that the memory usage and speed will improve under a more correct (non-approximate) folding model.

The complexity of our algorithm is the same as McCaskill's [13] requiring $O(n^4)$ time. It could be optimized to $O(n^3)$ using speedup by Lyngsø [23]. Also, JAX allowed us to run our algorithm in parallel on the GPU. Given sufficient cores, our algorithm runs in $O(n)$ time, since all table values for a given row indexed by i can be computed in parallel so long as the table values for rows with larger indexes have already been computed, and there are only n rows in total.

2.3 Test Case: Inverse Folding

A classical result of statistical mechanics is that the probability of a microstate $s \in S$, where S is the set of all possible microstates of a system, is proportional to its Boltzmann weight,

$$p_s \propto e^{-\beta E_s} \tag{11}$$

where β is the inverse temperature and E_s is the energy of s. Since the total probability of all microstates is equal to 1,

$$p_s = \frac{e^{-\beta E_s}}{Z} \tag{12}$$

So, the probability of a discrete RNA sequence q folding into some structure $s \in S$ is

$$p_s(q) = \frac{e^{-\beta E_{q,s}}}{Z_{\text{struct},q}} \tag{13}$$

Given the thermodynamic quantities introduced in Section 2.1, we can define an analog of Equation 13 for a probabilistic sequence x:

$$p_s(x) = \frac{Z_{\text{seq},s}(x)}{Z_{\text{ss}}(x)} \tag{14}$$

Intuitively, this can be interpreted as the probability that a sequence sampled from the distribution x will fold into secondary structure s. Note that in the case that x is a one-hot encoded vector, Equation 14 reduces to Equation 13.

To validate our method, we perform a small set of inverse folding experiments on structures from the Eterna100 [24], a well known benchmark for inverse RNA folding. We consider all structures from the Eterna100 of length at most 50 given computational limitations of our implementation. For each structure s, we apply simple gradient descent over the probabilistic sequence x to maximize $p_s(x)$. In practice, we define our loss function as $\mathcal{L}_s(x) = -\log(p_s(x))$. To ensure that the normalization condition in Equation 3 is satisfied for each gradient update, we apply the softmax operator to the sequence logits at each step. All probabilistic sequences were initialized with uniformly distributed base identities at each position. Each design experiment was conducted using 200 iterations of gradient descent with the RMSProp optimizer [25] and a learning rate of 0.1.

The result of this optimization is, by construction, a distribution over sequences and not a single discrete sequence. In our examples, the softmax of sequence logits converged to a one-hot encoded sequence. Though not used in this work, convergence to a discrete sequence can be enforced via the method developed by Roney and Ovchinnikov for inverse protein design with AlphaFold [17].

Puzzle ID	Initial	Optimized	Answer 1	Answer 2
1	0.017	0.976	0.402	0.909
3	$\approx 10^{-13}$	0.988	0.420	0.798
8	0.206	0.984	0.545	0.596
10	$\approx 10^{-21}$	0.962	0.530	0.716
11	$\approx 10^{-11}$	0.941	0.449	0.562
15	$\approx 10^{-20}$	0.002	0.403	0.540
20	$\approx 10^{-14}$	0.209	0.244	0.588
23	$\approx 10^{-8}$	0.563	0.005	0.021
26	$\approx 10^{-7}$	0.987	0.235	0.241
30	$\approx 10^{-9}$	0.980	0.094	0.162
33	$\approx 10^{-27}$	0.726	0.676	0.594
40	$\approx 10^{-16}$	0.990	0.835	0.794
41	$\approx 10^{-23}$	0.021	0.001	$\approx 10^{-6}$
47	$\approx 10^{-31}$	0.381	0.002	0.007
57	$\approx 10^{-35}$	$\approx 10^{-8}$	$\approx 10^{-12}$	$\approx 10^{-14}$
65	$\approx 10^{-21}$	0.101	0.133	0.136
66	$\approx 10^{-25}$	0.003	0.001	$\approx 10^{-4}$
88	$\approx 10^{-19}$	0.011	0.014	0.013

Table 1: The results of the inverse folding procedure described in Section 2.3 on all Eterna100 structures of length at most 50. Each structure is identified by its ID in the Eterna100 dataset. The "Initial" column corresponds to the probability that a sequence sampled from the initial sequence distribution will fold into the desired structure. The "Optimized" column corresponds to the same probability for the optimized sequence distribution after 200 gradient updates. "Answer 1" and "Answer 2" correspond to the probabilities of the two Eterna100 reference answers folding into the target structure.

3 Results

We sought to validate the utility of the gradients calculated from the procedure described in Section 2.2 for numerical optimization. To do so, we conducted inverse folding experiments per the method described in Section 2.3 on all 18 structures of length at most 50 from the Eterna100. The results of all 18 optimizations are presented in Table 1. The post-compilation runtime of a single gradient step ranged from $0.06 - 2.67$ seconds on an 80 GB NVIDIA A100 GPU.

In all cases, the initial probability distribution of sequences was improved by gradient descent, as measured by the probability of sampling a sequence that folds into the target structure. In most cases, the probability improved by many orders of magnitude. In 14 of the 18 cases, the final probability distributions (all of which converged to a discrete sequence) outperform both reference answers provided in the Eterna100 dataset. In the remaining four cases (i.e., puzzles 15, 20, 65, and 88), visual inspection of the final probability distributions suggests that the optimization procedure arrived at local minima. Additionally, two of these four cases (i.e. puzzles 57 and 88) are known to be unsolvable using the Vienna 2 parameters [26].

In all of our experiments, normalized sequence entropy drops quickly over the first 30 steps and sequence distributions settle down to a preferred base at each position. In many cases though we see subsequent smaller changes to base identities and corresponding improvements to target probability after many steps of optimization and even after phases of seemingly little change. In Figure 2 we visualize the successful optimization run for puzzle 10 ("Frog Foot"). After 120 optimization steps the U14-A19 positions, which are base-paired in the target structure, change to a C-G base pair via a U-G intermediate, resulting in an an additional small improvement to target probability. Other changes at positions unpaired in the target structure are the intermittent A to U switch at position 6 during steps 25–175 and the U to A change at position 10 around step 175.

In optimization runs of intermediate or low success as measured by final target probability, as shown in Figures 3 and 4 for puzzles 47 and 41, we still see improvements happening over the course of the optimization run, albeit from a low base of target probability, indicating that the optimization run might be stuck in a local minimum.

The results show that our method was successful in computing meaningful gradients. In addition, straightforward gradient optimization was surprisingly effective for the inverse folding problem.

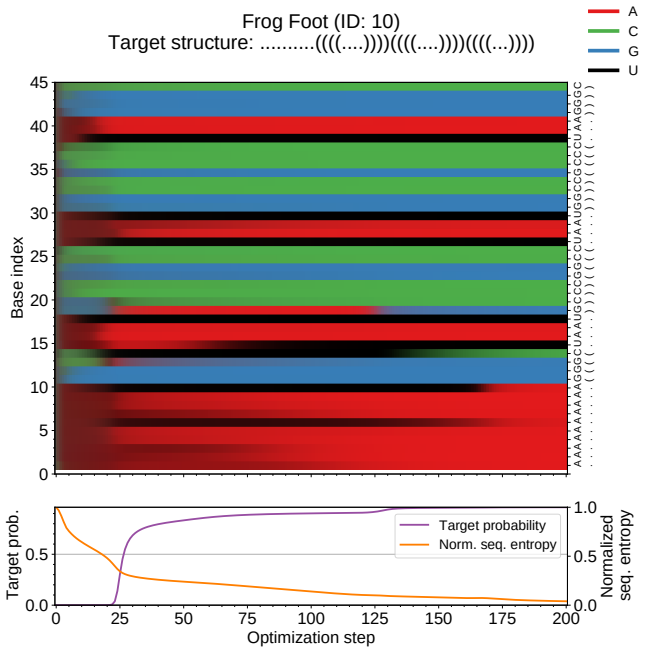

Figure 2: Overview of sequence optimization run for puzzle 10 ("Frog Foot"). Evolution of sequence probability distributions (above), probability of target structure and normalized sequence entropy (below) over the course of an optimization run. The sequence probability distributions at each site are projected into RGB-space and represented as colors. On the right hand side of the sequence probability distributions is the sequence of most likely bases as well as the minimum free energy (MFE) structure predicted for this sequence. This figure shows a successful optimization run with a final target probability of 0.962, with base identity changes happening throughout the optimization run.

4 Discussion

The goal of this work was to develop a differentiable means of computing the partition function for an RNA sequence. To this end, we introduced a new differentiable generalization of McCaskill's algorithm that accepts probabilistic sequences as input (see Section 2.2). Ideally, our algorithm should be practically useful as well as theoretically sound. There is no theoretical guarantee that the gradients computed via this procedure will be useful for numerical optimization. The results presented in Section 3 provide empirical evidence for this. While there are many relevant loss functions of the partition function over which one could optimize, inverse folding is a straightforward and practical demonstration of our method. Moreover, the arrival at local minima via our simple optimization procedure is an expected property of gradient descent and suggests that combining our method with additional design strategies (e.g., an adaptive walk, multiple initializations) could lead to a breakthrough in this domain.

Given the gradients of the structure-sequence partition function with respect to sequence probability, it is not immediately apparent that gradient descent should always collapse the sequence probability distributions to a one-hot encoded sequence, yet this is what we mostly observed in our optimization runs. While this is certainly a desirable property, it could be advantageous to stop an optimization run earlier in order to sample from the resulting sequence distribution, thereby increasing the number of candidate solutions one can get from one run.

The rapid drop of normalized sequence entropy during the initial stages of optimization corresponds to a fixing of base identities at many positions that are base-paired in the target structure. These positions often do not change their base identity throughout an optimization run and bias the further search to a local minimum in sequence space. This could be improved by performing smaller gradient updates in the initial phase of sequence optimization, before the choice for a specific minimum has been made.

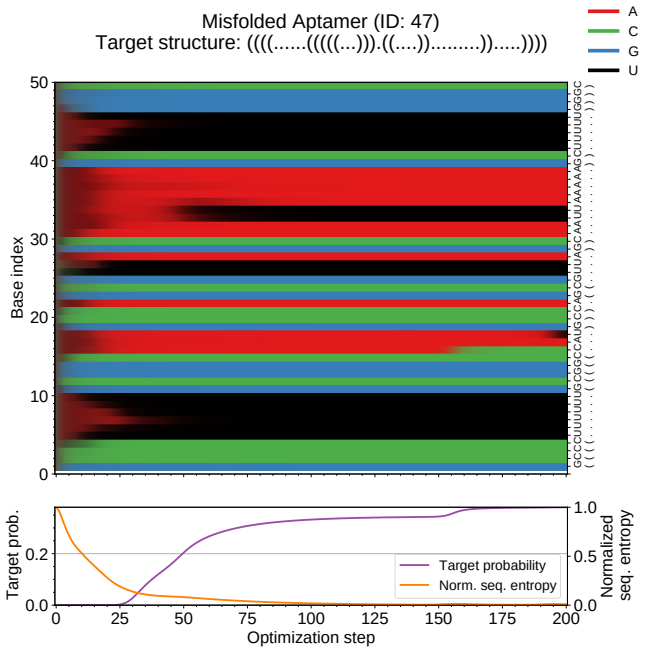

Figure 3: Overview of sequence optimization run for puzzle 47 ("Misfolded Aptamer"). This figure shows an optimization run with a final target probability of 0.381. The designed sequence of most likely bases has the correct minimum free energy structure, but the probability is only 0.381, indicating the possibility for better sequences to exist. Normalized sequence entropy drops off very quickly in the first 50 steps, and between steps 50–150 seemingly little happens. During steps 150–200, there are two base transitions in positions unpaired in the target structure, giving a small increase in target probability.

5 Future Work and Limitations

The primary limitation of this work is the memory cost of the current implementation. As discussed in Section 2.2, there are many subtle design choices in the recursions (e.g. the treatment of dangling ends and mismatches). In future work, we plan on implementing more memory-efficient variants of these recursions to scale our method to longer sequences. We expect this will be possible given that the approximation for dangling ends, terminal mismatches, and coaxial stacking used by ViennaRNA leads to poor memory usage when adapted to probabilistic sequences. By using a non-approximate model, we could actually use less memory.

Given a means to calculate gradients for sequence lengths of practical importance, there are many applications of our method for the inverse folding of RNA sequences with desired thermodynamic properties. For example, extensions of our method for designing sequences with desired secondary structure could be used for the design of mRNA molecules, such as those used in mRNA vaccines, with specified sequence and structure constraints [27]. Additionally, since we optimize over probabilities, our method could also be used to optimize for more complex equilibrium properties such as bistability. It is also straightforward to optimize for thermodynamic properties other than probabilities of equilibrium structures; for example, one could modify the procedure described in Section 2.3 to optimize for sequences with a desired heat capacity.

Instead of using a fixed energy model and optimizing over the sequence to minimize some function of the partition function, one could equally fix a set of sequences and optimize over the parameters of the energy model to maximize agreement with experimental data [19]. One straightforward application would be a transparent and reproducible reparameterization of the nearest-neighbor model. Equivalently, one could efficiently fine-tune the nearest-neighbor model to chemically modified RNAs, *in vivo* data, or a particular RNA family.

Lastly, we envision our method serving as a loss function in deep learning. For example, Petti *et al.*'s differentiable version of the Smith-Waterman algorithm enabled jointly learning a multiple sequence alignment (MSA) and downstream parameters (e.g. AlphaFold weights) rather than treating MSA generation as a preprocessing step. Analogously,

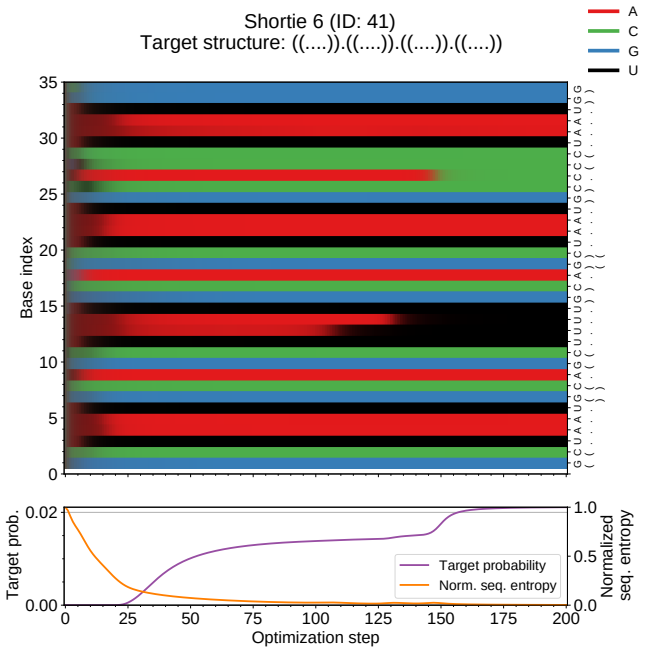

Figure 4: Overview of sequence optimization run for puzzle 41 ("Shortie 6"). This figure shows an optimization run with a final target probability of 0.021. The normalized sequence entropy drops quickly during the first 30 steps, fixing the base identity at all positions that are base-paired in the target structure. The A to C transition at position 27 around step 150 gives a small improvement to target probability.

existing deep learning approaches to RNA structure prediction often use secondary structure as a preprocessing step and could be made end-to-end differentiable via our method [28, 29].

We note that any application of our method to RNA can also be applied to DNA using the Santa-Lucia parameters for the nearest-neighbor model [30].

6 Conclusion

The partition function is the fundamental quantity in statistical mechanics and captures the statistical properties of a system in thermodynamic equilibrium. McCaskill's dynamic programming algorithm for calculating the structure partition function of an RNA sequence allows one to study such properties for a given sequence computationally. However, the reverse is not true; it is not straightforward to use McCaskill's algorithm to design sequences with desired thermodynamic properties.

In this work, we modify McCaskill's algorithm so that it is differentiable. This allows one to apply gradient descent over a probability distribution of sequences to optimize with respect to macroscopic thermodynamic quantities. We demonstrate this approach empirically by optimizing for sequences that maximize the probability of folding into a given secondary structure (see Section 3), but note that this method can be adapted to many different optimization problems (see Section 5).

This work serves as an example of the synergies between existing analytical models of physical systems and differentiable programming. The adaptation of existing models to be well-defined over continuous inputs not only serves as a powerful path towards inverse folding, but one in which the relevant quantities are expressive and readily interpretable.

Software Availability

We have made two implementations of our method publicly available. First, an implementation in JAX using reverse-mode automatic differentiation: `https://github.com/rkruegs123/jax-rnafold`. Second, an implementation in Julia using forward-mode automatic differentiation: `https://github.com/marcom/DiffFoldRNA.jl/`.

References

[1] Harold S Bernhardt. The RNA world hypothesis: the worst theory of the early evolution of life (except for all the others). *Biology direct*, 7(1):1–10, 2012.

[2] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[3] Jennifer A Doudna and Thomas R Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002.

[4] Alexander Serganov and Dinshaw J Patel. Ribozymes, riboswitches and beyond: regulation of gene expression without proteins. *Nature Reviews Genetics*, 8(10):776–790, 2007.

[5] Ignacio Tinoco Jr and Carlos Bustamante. How RNA folds. *Journal of molecular biology*, 293(2):271–281, 1999.

[6] Michael Zuker and Patrick Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981.

[7] Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6(1):1–14, 2011.

[8] Jessica S Reuter and David H Mathews. RNAstructure: software for rna secondary structure prediction and analysis. *BMC bioinformatics*, 11(1):1–9, 2010.

[9] Douglas H Turner and David H Mathews. Nndb: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, 38(suppl_1):D280–D282, 2010.

[10] David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure. *Journal of molecular biology*, 288(5):911–940, 1999.

[11] David H Mathews, Matthew D Disney, Jessica L Childs, Susan J Schroeder, Michael Zuker, and Douglas H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of rna secondary structure. *Proceedings of the National Academy of Sciences*, 101(19):7287–7292, 2004.

[12] Ignacio Tinoco, Olke C Uhlenbeck, and Mark D Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:362–367, 1971.

[13] John S McCaskill. The equilibrium partition function and base pair binding probabilities for rna secondary structure. *Biopolymers: Original Research on Biomolecules*, 29(6-7):1105–1119, 1990.

[14] David H Mathews. Revolutions in RNA secondary structure prediction. *Journal of molecular biology*, 359(3):526–532, 2006.

[15] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[16] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[17] James P. Roney and Sergey Ovchinnikov. State-of-the-art estimation of protein model accuracy using alphafold. *Phys. Rev. Lett.*, 129:238101, Nov 2022.

[18] Igor Melnyk, Aurelie Lozano, Payel Das, and Vijil Chenthamarakshan. Alphafold distillation for improved inverse protein folding. *arXiv preprint arXiv:2210.03488*, 2022.

[19] Mirela Andronescu, Anne Condon, Holger H Hoos, David H Mathews, and Kevin P Murphy. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23(13):i19–i28, 2007.

[20] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[21] Max Ward, Eliot Courtney, and Elena Rivas. Fitness functions for rna structure design. *bioRxiv*, 2022.

[22] Kévin Darty, Alain Denise, and Yann Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974, 2009.

[23] Rune B Lyngsø, Michael Zuker, and Christian NS Pedersen. Internal loops in rna secondary structure prediction. In *Proceedings of the third annual international conference on Computational molecular biology*, pages 260–267, 1999.

[24] Jeff Anderson-Lee, Eli Fisker, Vineet Kosaraju, Michelle Wu, Justin Kong, Jeehyung Lee, Minjae Lee, Mathew Zada, Adrien Treuille, and Rhiju Das. Principles for predicting rna secondary structure design difficulty. *Journal of molecular biology*, 428(5):748–757, 2016.

[25] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[26] Rohan V Koodli, Boris Rudolfs, Hannah K Wayment-Steele, Eterna Structure Designers, and Rhiju Das. Redesigning the eterna100 for the vienna 2 folding engine. *bioRxiv*, pages 2021–08, 2021.

[27] Hannah K Wayment-Steele, Do Soon Kim, Christian A Choe, John J Nicol, Roger Wellington-Oguri, Andrew M Watkins, R Andres Parra Sperberg, Po-Ssu Huang, Eterna Participants, and Rhiju Das. Theoretical basis for stabilizing messenger RNA through secondary structure design. *Nucleic acids research*, 49(18):10604–10617, 2021.

[28] Robin Pearce, Gilbert S Omenn, and Yang Zhang. De novo rna tertiary structure prediction at atomic resolution using geometric potentials from deep learning. *bioRxiv*, 2022.

[29] Yang Li, Chengxin Zhang, Chenjie Feng, Peter L Freddolino, and Yang Zhang. Integrating end-to-end learning with deep geometrical potentials for ab initio rna structure prediction. *bioRxiv*, pages 2022–12, 2022.

[30] John SantaLucia Jr and Donald Hicks. The thermodynamics of dna structural motifs. *Annu. Rev. Biophys. Biomol. Struct.*, 33:415–440, 2004.