RNA Macrostates and Computation Tools

Michael Flynn

A Thesis
submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Arts with Honors
in Physics

WILLIAMS COLLEGE
Williamstown, MA

Acknowledgments

I owe gratitude to a great many people. Thank you to my thesis advisor Daniel Aalberts for accepting me as your thesis student and advising me through the long process of writing a thesis. With your help and encouragement I felt like I was able to make an impact in the field as an undergraduate. I'd like to thank the faculty of the Physics department of Williams College who have instructed me including Bill Wootters, Kevin Jones, David Tucker-Smith, Ward Lopes, Michael Seifert, Frederick Strauch, and Charlie Doret for their patience, support and incredibly lucid explanations. I'd also like to thank the Computer Science faculty, including Tom Murtagh, Jeannie Albrect, Brent Heeringa, Morgan McGuire, Duane Bailey, Bill Lenhart, Stephen Freund, and Brent Yorgey for their patience as well, and for making some of my favorite classes at Williams.

I'd like to thank Michael Zuker and David Mathews for their correspondance and help getting started in the field of RNA secondary structure research. Thank you to Nick Markham for writing a very clear Ph. D. thesis.

I'd like to thank my friends including Maoli Vizcaino for keeping me company while I toil, Tony Blanco, Qadir Forbes, JL Etienne and Dan Evangelakos for their support, good coversations, and friendship.

Most of all I'd like to thank my parents and my family for always being there. No matter happens, I know I can come home to open arms.

Contents

List of Figures

Chapter 1

Introduction

1.1 RNA Secondary Structure Prediction

In the nucleus of a cell, RNA is constantly being synthesized by transcribing sections of DNA into mobile chains built from the nucleic acids adenine, gaunine, cytosine, and uracil. These RNAs serve several functions inside of the cell, including:

mRNA: 'messenger RNA' which serve as blueprints for proteins,

tRNA: 'transfer RNA' which bond to and transport amino acids to the ribosome to be formed into proteins,

rRNA: 'ribosomal RNA' which make up ribosomes,

microRNA which bind to mRNA and modify translation or degradation rates,

ncRNAs 'non-coding RNAs' which are not involved in the manufacturing of proteins, instead being used as tools of the cell for tasks including the regulation of gene expression, or

others such as snoRNAs which recognize splice cites.

The last group, ncRNAs, comprises the majority of RNAs synthesized and have mostly unknown functions. It is unlikely that they just float around the cell uselessly, rather they are tools the cell uses to build and run itself. It is widely believed that the function of a strand of RNA is directly related to its structure. While DNA is composed of 2 separate strands base-paired to make a double helix, RNA is most often found as a single stranded backbone that makes base pairs with other parts of itself. There are considered to be 3 main levels of structure of RNA. The *primary structure* is the sequence of nucleic acids that make up the strand of RNA i.e. 'GACCUUGGGGCCCC...'. The *secondary structure* is how these bases form base pairs (see Figure 1-2), most often of the Watson and Crick variety ('G-C', 'A-U', although sometimes 'G-U' is possible as well). The *tertiary structure* is how the structure bends on a larger scale as the stems and loops formed by the secondary structure interact with each other.

Primary structure can be readily observed by modern sequencing technology, however it is the secondary structure that determines the shape of the molecule, which is the most important when considering interactions with other biological molecules. The full specification of a secondary structure state includes a list of every base pair. Finding the secondary structure of an RNA molecule is a different task than finding the primary structure of RNA because it samples many states in thermal equilibrium. For an individual RNA molecule there are many valid pairings, in fact for a sequence of length n nucleotides there are $O(1.8^n)$ secondary structures [8].

[Todo: RNA is outer-planar graph, no pseudoknots]

To describe the structural ensemble, we turn to statistical mechanics. We approximate the RNA molecule as an isolated system in contact with a thermal resevoir that is the cell, with each secondary structure as a state of that system. In such a system the probability of any state s is its Boltzmann factor divided by the partition

function:

$$P(s) = \frac{1}{Z} e^{-\beta E(s)}, \tag{1.1}$$

where $\beta = \frac{1}{RT}$, R is the gas constant, T is the temperature, and

$$Z = \sum_{s} e^{-\beta E(s)}. \tag{1.2}$$

Initially researchers were satisfied with presenting the MFE (minimum free energy) state as the state the molecule assumes in nature, after all this state is the most probable. However we will see that, perhaps counterintuitively, the MFE structure is not very probable (see Figure 2-2). Lokking at the whole or sample portion of theBoltzmann distribution, are more modern approaches for predicting the secondary structure ensemble found in nature.

1.2 The Energy Model of RNA

Computing the partition function and probabilities is impossible without an energy model for RNA, $E(s)$. Setting the energy of the single-stranded (no pair) state as $E = 0$, the energy model must accurately estimate an energy for a folded secondary structure. The model parameters we will use come from measuring ΔG of pairing complementary sequences by measuring the relative concentrations of single-stranded states to double-stranded states in solution (see UV melting section).

In early secondary structure prediction algorithms [16], energy bonuses were assigned for any pair. The Nussinov & Jacobson algorithm reduced to finding the legal folding with the most base pairs. After, it was discovered exprimentally that G-C pairs are more stable than A-U pairs. Because of this, in further iterations the energy would be determined by counting hydrogen bonds of cannonically paired bases, assigning each -1 kcal/mol of free energy. This would mean that GC pairs are given

-3 kcals/mol, AU and GU pairs are both given -2. Additionally, certain loops were given energy penalties for forming. The Zuker and Steigler algorithm, developed for this model, minimized the free energy [20]. Both this algorithm and the previous one had elementary dynamic programming solutions. They were useful models to use as a baseline, however, even the second iteration was not very accurate, on average only 20.5% of known base pairs are correctly predicted. Later energy models would use it as a control for the hypothesis that they increased secondary structure prediction accuracy [13].

Indeed, much improvement was made over the hydrogen bond model by expanding it to include what is now called the Nearest-Neighbor model. Experiments made it clear that energy of an RNA folding is not just linearly dependent on the bonds that are made. There are significant interaction effects between neighboring bases and bonds, this is called 'sequence dependence', and there are polymer physics based free energy terms that scale logarithmically with the length of a loop (this comes from the entropy loss to make a loop). Zuker realized the we can represent secondary structure in terms of base pairs or in terms of loops (including base pair stacks) which are in the Nearest-Neighbor model. We divide our structure into its loops and compute the energy of each loop, with a separate energy model for each.

In Figure 1-2, these loops are described. In brief, there is a different category and energy model for each type of loop with different numbers of paired and unpaired bases. For example a stack loop has 2 pairs and no unpaired bases. An internal loop has 2 pairs and intervening unpaired bases. A hairpin has 1 pair and any amount of unpaired bases. Finally a multi-loop has at least 3 pairs and unpaired bases between them.

There is a 5th type of loop, called a pseudoknot. RNA secondary structure is normally assumed to be an outer-planar graph. A pseudoknot consists of 2 "crossing" pairs. This breaks the standard topology of the loop diagrams and makes compu-

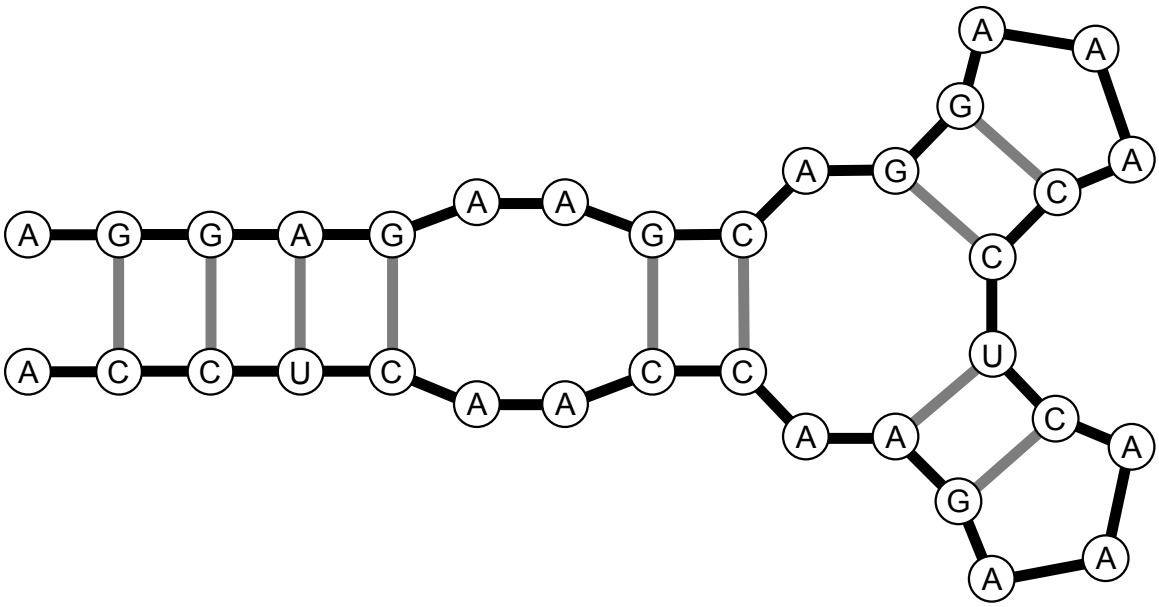


Figure 1-1: This is one possible folding out of the $O(1.8^n)$ secondary structures of the sequence 'AGGAGAAGCAGGAAACCUCAAAGAACCAACUCCA'.

---

tation of the partition function provably NP-hard. For this reason, pseudoknots are excluded from consideration. This is not ideal because pseudoknots do appear in nature, notably in ribosomal RNA and at a rate of up to 12% in Group I and Group II Introns [13]. However, some of the results of this reseach might make the computation of a certain restricted set of pseduoknots much more efficient.

1.2.1 Loop Parameters

The strand 'GGGAAACCC', will predictably form a stem-loop structure. We can decompose the energy as such:

$$\Delta G\left(\begin{array}{c}\text{G G G A}\\ \text{C C C A}\end{array}\right) = \Delta G_S\left(\begin{array}{c}\text{G G}\\ \text{C C}\end{array}\right) + \Delta G_S\left(\begin{array}{c}\text{G G}\\ \text{C C}\end{array}\right) + \Delta G_H\left(\begin{array}{c}\text{G A}\\ \text{C A}\end{array}\right) \tag{1.3}$$

Where ΔG_S is the function for free energy of a stack loop and ΔG_H is the function

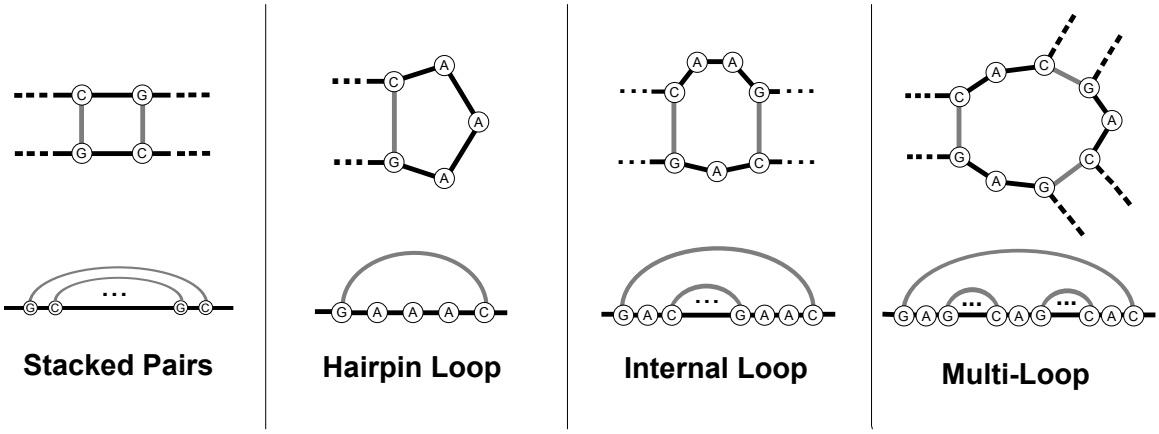


Figure 1-2: The 4 types of loops, black connections are bonds in the RNA backbone, grey connections are hydrogen bonds between bases. The types are: *Stacked Pairs*, adjacent pairs of bonded bases, *Hairpin Loops*, one bonding pair closing off a turn in the RNA backbone, *Internal Loops*, which can range from bulges to long loops, connecting to pairs with 2 chains of unpaired bases, and *Multi-Loops*, which connect 3 or more pairs. The diagram displays the loops with how they will appear in the folded structure pictured above, and below is how the same loops will be layed out many diagrams of RNA structure, such as RNAbows [1].

---

that gives you the energy of a hairpin loop.

For stack loops, the legal pairings restrict the possible loop types. The free energy of all possible stack loops were measured, each with a seperate parameter, i.e.

$$\Delta G_S \left(\begin{matrix} G & G \\ C & C \end{matrix}\right), \ \Delta G_S \left(\begin{matrix} G & A \\ C & U \end{matrix}\right), \ \Delta G_S \left(\begin{matrix} A & A \\ U & U \end{matrix}\right), \ \ldots \tag{1.4}$$

where each are parameters (the βs) to a linear regression model we fit to ΔG of 'GGGAAACCC' and similar sequences (see section: UV Melting Experiments). This is what is done in practice.

Small internal loops and hairpins were parameterized individually but transition to general forms for large loops to keep the model finite. They are as follows:

Hairpin Loop Loops with 3 and 4 unpaired bases are kept in special tables of triloop and tetraloop parameters, respectively. Each possible tetraloop and triloop

has its own energy determined by experiment. Beyond that, the general model is

$$\begin{aligned}\Delta G_H(n > 3) = \ & \Delta G_{init}(n) + \Delta G_{HStack}(\text{Initializing Stack}) \\ & + \Delta G(\text{bonuses}).\end{aligned} \tag{1.5}$$

As you can see there is an initialization term and a stack term that are both fitted via linear regression. The bonus term is for various special loops that have been experimentally found to be more stable. It's outside the scope of this thesis to get into them, but they are specified in the paper by Mathews et al [13].

Internal Loop Much like stack loops, internal loops are given individual parameters for 1×1, 1×2, 2×2, internal loops, where $n \times m$ denotes one arm of the loop having n unpaired bases and the other having m. This results from extensive studies on the ΔG's of these internal loops. For the rest of internal loops there's a model of the form:

$$\begin{aligned}\Delta G_{int}(n \times m) = \Delta G_{init}(n + m) + & \Delta G_{asymmetry}(|n - m|) \\ & + \Delta G(\text{bonuses}),\end{aligned} \tag{1.6}$$

where each term on the right is a regression parameter and the bonus term is similar to the hairpin bonus term for loops composed and ended with certain special kinds of bases determined experimentally to be more stable.

Multi-Loops Multi-Loops are harder to create experimental strands for and because of this there are no individual parameters for multi loops. In fact, for modeling mutli-loops we regress to a more simple linear model of the form:

$$\Delta G_{multi} = a_1 + b_1 n + c_1 h + \Delta G_{dangle}, \tag{1.7}$$

where a_1 is a penalty for starting a mutl-loop, n is the number of unpaired bases in the loop, b_1 is an energy penalty per unpaired base, h is the number of pairs in the structure, c_1 is the energy bonus per pair, and ΔG_{dangle} is a term similar to stack loop terms which include the energy of the unit composed of a pair and its 2 adjacent unpaired bases, if it has them.

1.2.2 UV Melting experiments

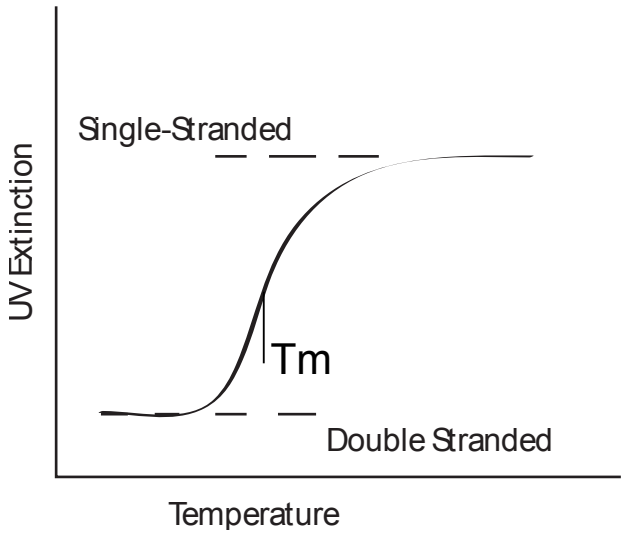


Figure 1-3: An example of the output of a UV melting experiment. We should expect to see 2 levels of extinction in the graph, one corresopoding to where single-strandedness is the equilibrium condition of the strand and the other where double strandedness is the equilibrium. As the temperature increases, the strand absorbs more energy from the environment allowing it to escape the double-stranded state and so there is a transition interval where as the temperature increases the UV extinction goes from the double-stranded extinction to the single stranded extinction.

At previously stated, loop regions are given energies as parameters to linear regression models of free energy change in predictably folding strands. For example, the strand 'GGGAAACCC' folds predictably into a structure with all the G's paired to the C's and a 3-A hairpin turn (because G's pair very strongly to C's and neither are complementary to A) as seen in Equation 1.3. David Turner executed the following experimental process in the 90s: large amounts of identical strands are synthesized

and put into solution and heated. A two-state assumption is made, either the RNA is folded in a 'double-stranded' state or unfolded in a 'single-stranded' state. As the solution heats, there is enough ambient energy to put all the strands in the unfolded, no-bonds state. RNA is an organic, aromatic molecule that absobes light in the UV spectrum in different amounts depending on whether it is in a folded state or unfolded state, so the UV absobtion is fit to a curve that then tells us about the relative concentrations of the single-stranded vs. double-stranded state, which in turn tells us about the free energy change between the two at a given temperature. This free energy change is extracted and treated as a function of the loop variables, which are then fitted to a linear model over experiments on many different such strands [18].

For example, if we wanted to compute the free energy change of a strand and fit it to its loop parameters:

$$\Delta G\left(\text{GGGA/AC/CCA}\right) = \Delta G_S\left(\text{GG/CC}\right) + \Delta G_S\left(\text{GG/CC}\right) + \Delta G_H\left(\text{GA/C/A}\right) \tag{1.8}$$

We synthesize large amounts of the strand 'GGGAAACCC', put them in solution and heat them and record their UV extinction. Observing the curve to be like something in Figure 1-3. The melting temperature T_m is defined to be where the concentrations of single stranded and double stranded molecules are equal, and this is taken to be the inflection point of the graph in Figure 1-3. From there Van't Hoff analysis is performed, where the concentration, C_T, is varied and this follows a model of the form:

$$\frac{1}{T_m} = \frac{R}{\Delta H}\log C_T + \frac{\Delta S}{\Delta H}. \tag{1.9}$$

This equation comes from the relation $\Delta G = -RT\log K_{eq}$ and plotting $1/T_m$ against $\log C_T$ and fitting a linear model gives us the parameters ΔS and ΔH from the slope

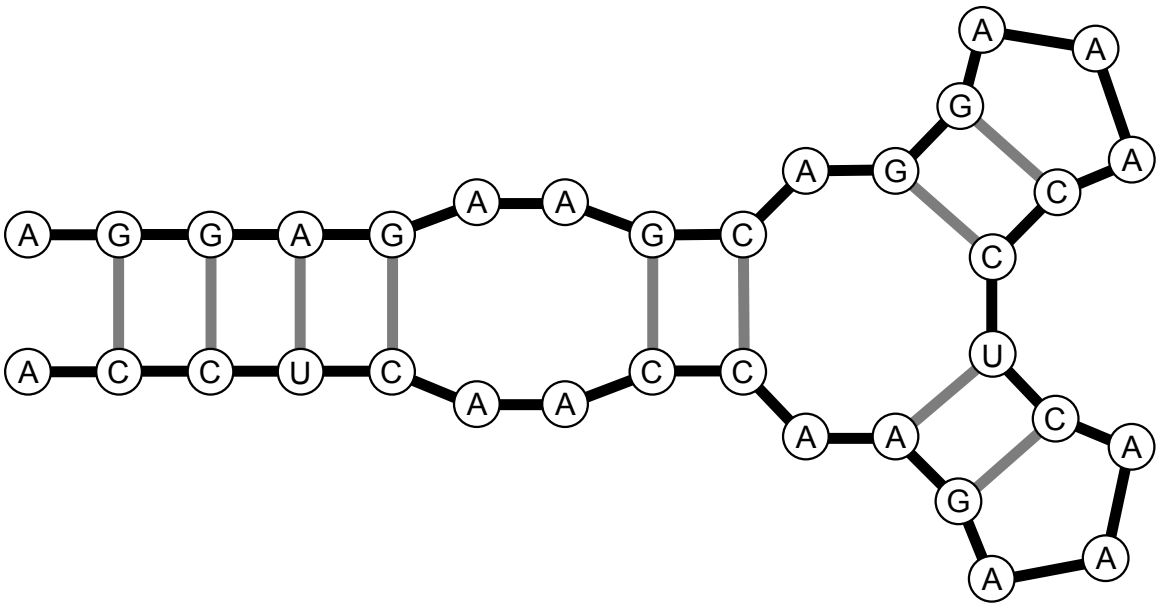


Figure 1-4: This is one possible folding of the sequence 'AGGAGAAGCAGGAAAC-CUCAAAGAACCAACUCCA'. The same picture as Figure 1-1.

---

and intercept and therefore ΔG through the relation

$$\Delta G = \Delta H - T\Delta S. \tag{1.10}$$

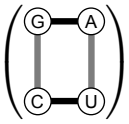
Repeating this process over several strands, Turner fit the ΔG's to a linear model based on the many parameters described in the Loop Parameters section. From there, when we want to compute the energy of a folding, all we have to do is seperate it into loops and sum the energies of each based off the parameters of Turner's model.

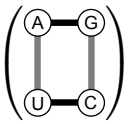
1.2.3 Example Energy Computation

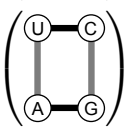
Here's an example of the energy computation of a secondary structure, pictured in Figure 1-4. There are 3 stack loop of the type:

$$\Delta G \left(\begin{matrix} G & G \\ C & C \end{matrix} \right) = -3.26 \text{ kcal/mol}, \tag{1.11}$$

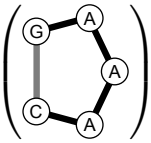
and one of each:

$$\Delta G \left(\text{  } \right) = -2.35 \text{ kcal/mol}, \tag{1.12}$$

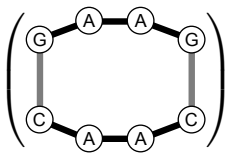
$$\Delta G \left(\text{  } \right) = -2.35 \text{ kcal/mol}, \tag{1.13}$$

$$\Delta G \left(\text{  } \right) = -2.08 \text{ kcal/mol}. \tag{1.14}$$

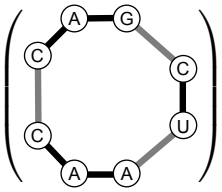
These are all lookups from linear regression parameters. Besides the stacks, there are 2 identical hairpins, an internal loop, an external loop, and a multi-loop. For the hairpins the energy can be looked up in a special parameter table designed specifically for hairpins with 3 unpaired bases called triloops:

$$\Delta G \left(\text{  } \right) = 5.8 \text{ kcal/mol}. \tag{1.15}$$

For the internal loop, there is a direct lookup for the 2×2 mismatch in a table:

$$\Delta G \left(\text{  } \right) = 0.5 \text{ kcal/mol}. \tag{1.16}$$

The multibranch loop has 3 pairs and 2 unpaired bases. The energy is therefore:

$$\Delta G \left(\text{  } \right) = 3.4 + 0 * 3 + 0.4 * 2 = 4.2 \text{ kcal/mol}. \tag{1.17}$$

If we sum up the contributions, we get that $\Delta G = -6.36$ kcal/mol. The energy of the MFE state can be computed using software, and I find it to be -3.06 with just

the first stem and a large hairpin. [TODO: figure out what is wrong, make ct file, find energy].

1.3 Critique of the Energy Model

The free energy model of RNA is not perfect. It ignores how tertiary structure could influence the structure's stability. It is complex, there are linear models, non-linear models, lookup tables, and bonuses. Complexity is a very undesirable quality for a model, it makes it hard to implement and hard for new researchers to get started in the field. Not only that, but many parameters have very high error. For example, the initialization penalty for hairpin loops is supposed to be different for different hairpin lengths, but most of the terms aren't even 1σ away from each other. Because of this, it is not clear what advantage these extra parameters give, perhaps they could be reduced to one, to simplify the model greatly. Another problem is that the data of the original experiments has been lost, so no one can take a closer look at how well the parameters fit, and no one seems to be interested in running the experiments again. Additionally, the terms are heavily dependent on the salt concentration of the solution, and great care has to be made to imitate the conditions of the cell. It is hard to know whether the solutions from the UV melting experiment were prepared correctly.

The only thing that seems to keep the current parameters in good standing is relatively solid results in prediction, with something like 80% of base pairs correctly predicted. However, to make any progress from this point, it is essential to get the energy model correct.

If there was a list of big problems to tackle in RNA secondary structure prediction, fixing the energy model should be a top priority. The new model should be consistent and physically based and confirmed by experiments. Some groups have made progress

on creating energy models for multi-loops that may perform better than the current parameters. However, none of these changes have been implemented in the software packages that currently hold the grand majority of the secondary structure prediction market. There are reasons for this, including the fact that the software is massive, complicated, and poorly documented. Perhaps a new energy model could usher in a better software framework for RNA prediction that has:

1. Better naming conventions, with function, variable, and source file names describing precisely what they represent,

2. Consistently documented source files, perhaps with a full manual created with Doxygen,

3. Simpler energy models, with less branching and complexity.

This is not to belittle the efforts of those that implemented UNAfold and RNAStructure, after all they accomplished the great feat of implementing these complex rules in their software. Instead, it is to note that the job is not done, and the items listed above are things that need to be accomplished.

Chapter 2

Partition Function Computation and Improvements

2.1 Introduction

The partition function for a thermodynamic system of fixed volume, in contact with a heat reservior with absolute temperature T, is

$$Z = \sum_s e^{-E(s)/RT}, \tag{2.1}$$

where s denotes a particular state of the system, $E(s)$ is the energy of that state, and RT is the gas constant multiplied by the temperature, specified above. Each particular term in the sum is called that state's Boltzmann factor. The probability of a state is then said to be its Boltzmann factor divided by the partition function, or

$$P(s) = \frac{1}{Z} e^{-E(s)/RT}. \tag{2.2}$$

In our model an RNA strand is such a thermodynamic system, its secondary structure is its state, and the rest of the cell is the reservoir of heat. We assign each secondary

structure an energy according to the free energy model described in Chapter 1. According to this model, the energy of a secondary structure is the sum of the energies of its loops.

To compute the partition function we must sum up every possible secondary structure. There are many possible secondary structures, on the order of 1.8^n where n is the sequence length. However, the computation of the partition function benefits greatly from the linear nature of the energy model: it allows us to recursively define the partition function. For example, say we have computed the full partition function of a strand of n bases, define a function Q such that:

$$Q(i,j) = \text{ sum of Boltzmann factors for every structure contained between } i \text{ and } j. \tag{2.3}$$

The full partition computation can therefore be represented as the function Q acting on the bounding two bases 1 and n:

$$Q(1,n) = \sum_{s \text{ on } [1,n]} e^{-E(s)/RT}. \tag{2.4}$$

If we now extend the strand by attaching a small hairpin to the end, as pictured in Figure 2-1, there is no need to go back and recompute the energy of every single secondary structure, rather we can just multiply the already-computed partition function by the Boltzmann factor of the hairpin turn (let $E(h)$ be the hairpin energy) to get the new partition function:

$$Q(1,n+5) = \sum_{s \text{ on } [1,n]} e^{-(E(s)+E(h))/RT} = Q(1,n)e^{-E(h)/RT}. \tag{2.5}$$

This replaces an exponential-time computation (the sum), with a constant time computation (multiplying the pre-computed $Q(1,n)$ with the energy term). In general, this technique allows us to compute the partition function efficiently, working up from

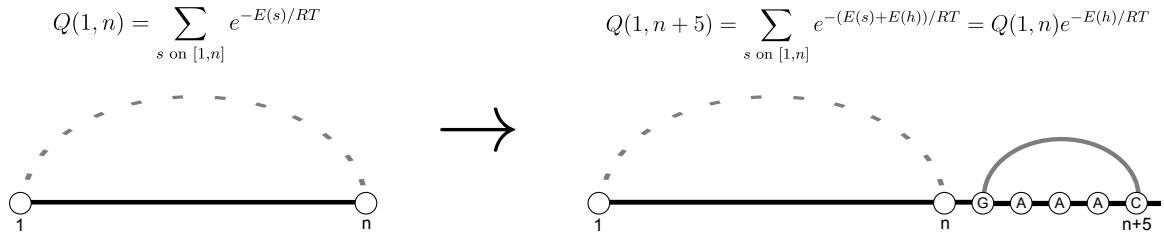


Figure 2-1: Extending the strand by adding an attached hairpin adds the energy of the hairpin to each structure. To compute the new partition function $Q(1, n+5)$, we do not need to sum the energies of all possible structures again, we can simply use the old value, $Q(1, n)$, and multiply it by the Boltzmann factor of the hairpin. This is the concept behind the dynamic programming algorithm.

$Q(1, 2)$ to $Q(1, n)$. In computer science this technique is called dynamic programming, a fancy name for the technique of storing the results of a computation in a table for later use.

The dynamic programming algorithm for computing the partition function of an RNA strands has several versions, depending on how in depth you go with the energy model. The simplest algorithms allow for only structures with nested base pairs. Nested means that the structure can be represented with parenthesis, for example $(((..((((....))))(((..)))..)))$, or without crossing pairs in expanded loop diagrams such as the ones in Figure 2-1. Note that this kind of structure excludes pseudoknots. If you ignore psuedoknots, which is standard in the field, and if you make an approximation that internal loops will never exceed a certain length, the fastest algorithm runs in $O(n^3)$. We believe that we can streamline this computation even more, taking advantage of the fact that empirically, the number of probable base pairs of a strand of length n seems to grow like n, not n^2. This is the same result we used to speed up the stochastic traceback algorithm and potentially a partition function algorithm that includes pseudoknots. The algorithm will be presented in its simplest form, with more complicated versions available in the appendices.

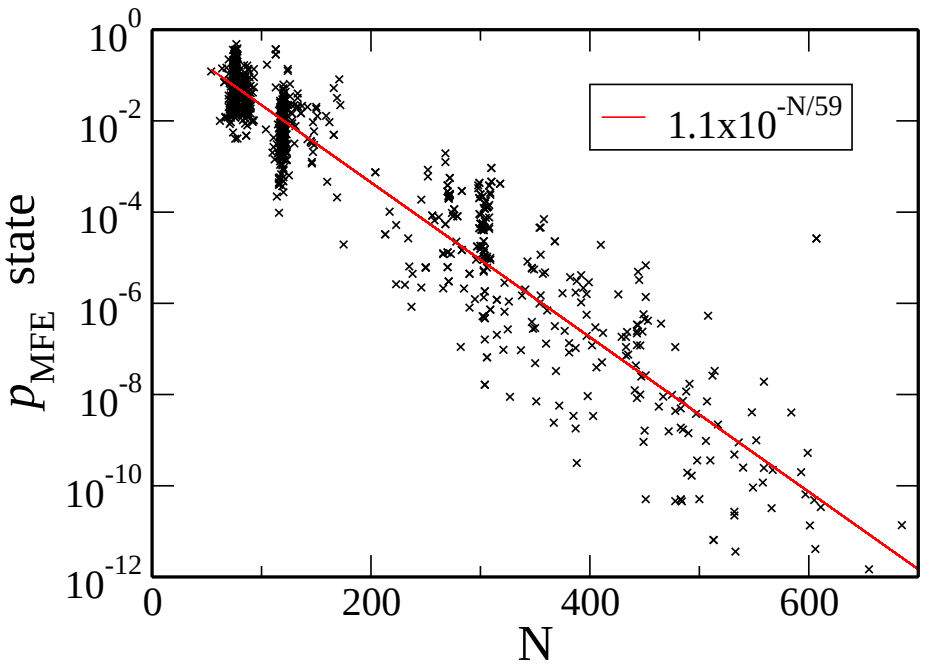


Figure 2-2: A plot of probability of the MFE state as computed $P(MFE) = e^{-E_{MFE}/RT}/Z$ for sequences of many different lengths. For sequences of reasonable length, even though it is the most probable state, the MFE probability approaches floating point error. This forces us to approach probabilities differently, using sums of probabilities of many similar states instead of just one state [**?**].

2.2 Motivation

There exists an efficient algorithm to find the minimum free energy structure of a sequence. The fundamental algorithm for the MFE was developed by Zuker and Steigler in 1981 [20]. However, perhaps counterintuitively, the MFE structure is not always the structure that is found in nature, even though natural systems tend to their lowest energy state. It is less counterintuitive after seeing Figure 2-2. The probability of any individual state is bounded from above by the MFE, which approaches zero very quickly as the length of the sequence increases. The encourages us to treat the probabilities of secondary structures given to us by Boltzmann statistics as probability densities instead of actual probabilities. The structure that is found in nature is most likely going to be from the most probable macrostate, where we define macrostate as following:

Macrostate: A set of RNA secondary structures sufficiently close to a local energy minimum structure s_{min}.

The "sufficiently close" in the definition is not widely agreed upon, and has spawned several different techniques of defining and computing RNA macrostates (see Clustering chapter).

This logic was presented by Ye Ding, Chi Yu Chan, and Charles E. Lawrence in their 2005 paper "RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble" [3]. What they found that, as suspected, the MFE state was not always a member of the most probable macrostate. Using a method based on centroids of statistically sampled states, they found that their method made 30% fewer prediction errors, improving positive predictive value by 46.5% and sensitivity by 21.7%. Figure 2-3 is an illustration of this concept.

In all the clustering algoritms, since we must know Boltzmann statistics of different states, we must compute the partition function. In Aalberts and Jannen's *Visualizing RNA base-pairing probabilities with RNAbow diagrams*, the PF method clusters using partition function data, requiring the partition function to be recomputed several times. If the partition function takes on the order of hours or days to compute, this can make partition function clustering a major bottleneck in whatever process RNA secondary structure prediction is being used in. However in these situations it is also true that the partition function is recomputed with almost the same conditions, just certain pairs restricted.

We've been able to show via experiment that the partition function only admits roughly $O(n)$ pairs with probabilities above thresholds around the machine precision limit. After the partition function is computed once, subsequent runs may ignore $O(n^2)$ pairs that have no sufficient probability. It suggests a general strategy to speed up the partition function, namely developing heuristics to eliminate improbable pairs in the initial computation. This motivates a method of computing the partition

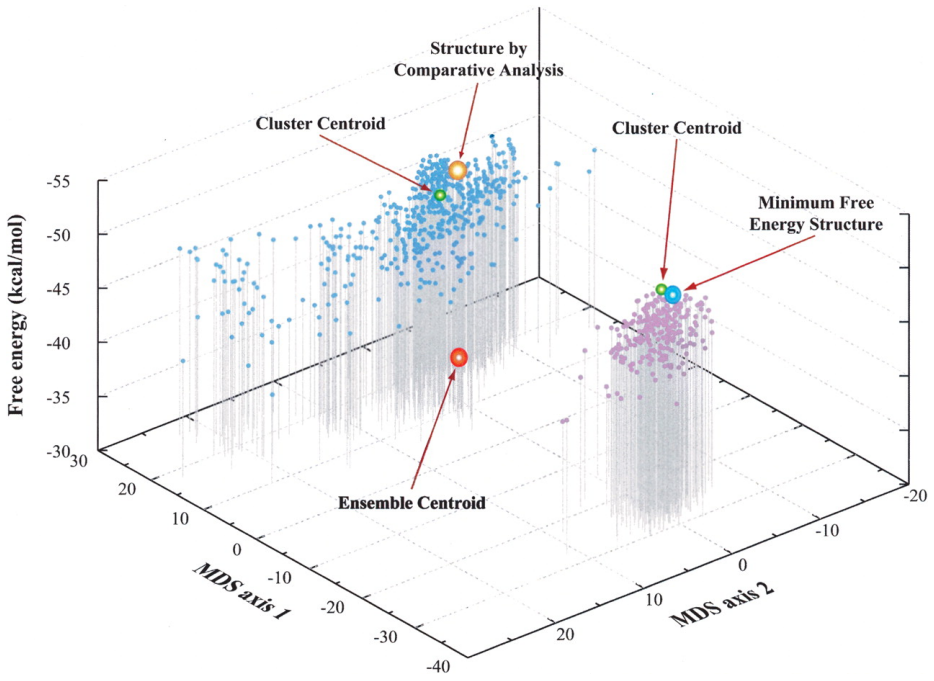


Figure 2-3: An RNA strand with mutliple macrostates, found by clustering analysis and plotted along 2 principle components. The MFE state is not in the most probably macrostate, neither is the ensemble centroid, but the macrostate centroid gives good predictions of secondary structure. The two axes are created by doing principle component analysis on the base-pair matrix. Figure from Ding, Chu, & Lawrence (2005).

---

function using a known-pairs heueristic to prune away unneccesary computation, and that is what we have implemented.

2.3 The Computation

The standard way of computing the partition function involves filling out a table where the (i, j)th member represents the partition function for the substrand from base i to base j, $Q(i, j)$. Because the energy model for RNA is linear, $Q(i, j)$ can be expressed as a function of nearby members of this table. This function is called the recurrence relation for the partition function of RNA. Because the free energy model is so complicated and has gone through many iterations, different RNA folding software packages implement different versions of the recurrence relation, and they

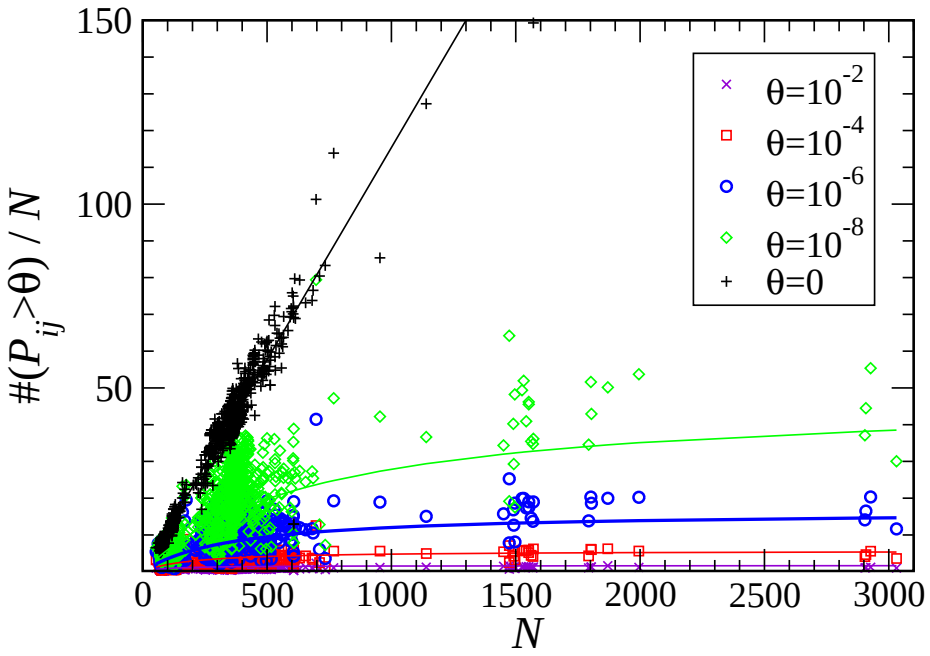


Figure 2-4: A plot of 'number of pairs with probability greater than colored threshhold' vs length of sequence. If there is no threshold, the number of pairs possible goes up like n^2, but even with small thresholds such as requiring a pair to have probability greater than 10^{-8}, the number of possible pairs goes flat as n increases. Therefore, a great amount of efficiency can be gained by restricting the pairs under consideration, while not sacrificing much accuracy since the thresholds are so low.

---

vary widely in complexity. This version is as simple as possible, there is an energy penalty for starting a multiloop, a, and an energy penalty for keeping a base unpaired, b, otherwise we hold to the standard energy functions.

McCaskill introduced the partition function recurrence relation for RNA in his classic 1990 paper *The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure* [14]. Here I will follow the clearer presentation by Dirks and Pierce [5].

Starting at the outermost layer, we compute $Q(i, j)$ by adding 2 cases together, either the strand is empty on that interval or it has a rightmost pair (d, e) (see Figure 2-5). If it has a rightmost pair, the partition function must be summed over every combination of structures on the substrand $(i, d-1)$ as well as every structure that

Figure 2-5: The recurrence relation for $Q(i,j)$. When we compute $Q(i,j)$ we only need to consider two cases, and add them together: either the strand will be empty of pairs, or there must be a rightmost pair followed plus whatever else is left, which can be represented as $Q(i,d-1)$. The partition function "underneath" the rightmost pair is $Q^b(d,e)$, which is the partition function with d and e constrained to be paired. The mathematical formula corresponding to this diagram is in equation 2.7.

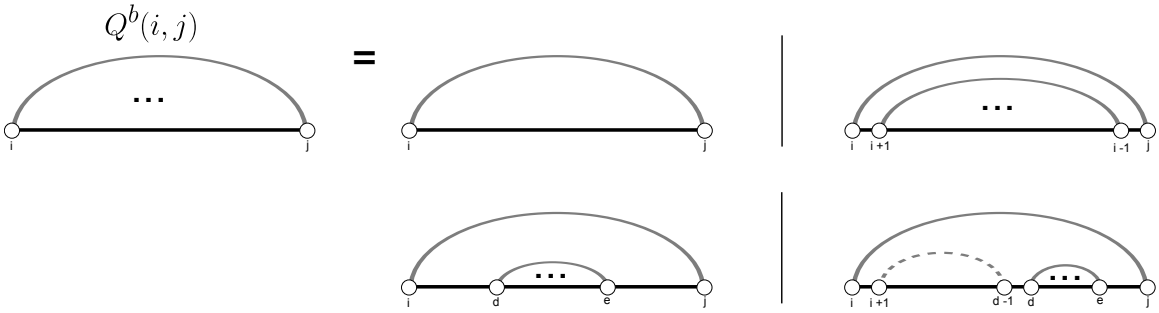


Figure 2-6: The recurrence relation for $Q^b(i,j)$. We consider 4 cases, corresponding to each of the type of loop that could form under the (i,j) pair, and add them together. The loop could form a hairpin, a stack loop, an internal loop, or a multi loop. The mathematical formula corresponding to this diagram is in equation 2.8.

---

could form underneath the pair (d,e). Define:

$$Q^b(i,j) = \text{sum of Boltzmann factors for every structure contained underneath assumed pair } (i,j). \tag{2.6}$$

Q^b differs from Q requiring that i and j be paired. Recalling Figure 2-1, we can get every combination of such states by multiplying $Q(i,d-1)$ by $Q^b(d,e)$ and an energy penalty for leaving $(e+1,j)$ empty. Of course, it must be summed over all possible indices d and e:

$$Q(i,j) = \overbrace{e^{-b(j-i+1)/RT}}^{\text{empty}} + \overbrace{\sum_{\substack{d,e \\ i \le d < e \le j}} Q(i,d-1)Q^b(d,e)e^{-b(j-e)/RT}}^{\text{rightmost pair}}. \tag{2.7}$$

To compute $Q^b(i,j)$ we sum over the 4 cases of loops that could form underneath the pair (i,j). This could mean a hairpin loop, with energy function E_h, a stack loop, with E_s, an interior loop, with E_i, or a multiloop with a, the multiloop penalty. For each new pair created, we must multiply in Q^b for that section as well, and for every open region created, Q (see Figure 2-6).

$$\begin{aligned} Q^b(i,j) = &\overbrace{e^{-E_h(i,j)/RT}}^{\text{hairpin}} + \overbrace{e^{-E_s(i,j)/RT} Q^b(i+1,j-1)}^{\text{stack}} \\ &+ \overbrace{\sum_{\substack{d,e \\ i<d<e<j}} e^{-E_i(i,d,e,j)/RT} Q^b(d,e)}^{\text{internal}} \\ &+ \overbrace{e^{-a/RT} \sum_{\substack{d,e \\ i<d<e<j}} Q(i+1,d-1) Q^b(d,e) e^{-b(j-e-1)/RT}}^{\text{multiloop}}. \end{aligned} \tag{2.8}$$

Assuming that we've already computed and stored $Q(d,e)$ and $Q^b(d,e)$ for all (d,e) such that $i \leq d < e \leq j$, each of those functions is just a table lookup. Therefore the only compute-bounding term is the sum over d and e, which makes computing $Q(i,j)$ an $O(n^2)$ computation, a big speed up over enumerating all $O(1.8^n)$ states. The same goes for $Q^b(i,j)$. One catch is that in order to compute $Q(i,j)$ or $Q^b(i,j)$, equation 2.8 requires computing all $O(n^2)$ $Q(d,e)$'s and $Q^b(d,e)$'s on the inside, so the entire algorithm is actually $O(n^4)$ but Lyngsø brought it down to $O(n^3)$ by restricting the lengths of internal loops [9]. This thesis presents improvements that bring the algorithm down to $O(n^2)$, provided a good heuristic, in the next section.

2.4 Efficient Partition Function

Let us assume that we have a huerstic of the form of a function

$$K : \text{Base Index} \rightarrow [\text{Base Index}], \tag{2.9}$$

that takes the index of a base as input and outputs a list of other bases it can pair to, where the length of this list is always very short and can be considered constant length in relation to the sequence length n. This is not an unreasonable assumption in certain cases see Figure 2-4. For example if we have already computed the partition function and wish to define macrostates around local minima as we will do in Chapter 4, the small number of pairs that are in the macrostate can speed computing the basin partition function. Another possibility could be restricting the base pairs based on sequence alignment. We could also just restrict non-Watson-Crick pairs to start. Importantly, we have seen empirically that the list of pairs above a probability theshold for each base will be very short.

The problem terms of each computation (equations 2.7 and 2.8) are the sums over d and e. In order to get the algorithm down to $O(n^2)$ each $Q(i,j)$ and $Q^b(i,j)$ must take constant time. Efficiency gains have already been seen by McCaskill by limiting the internal loop length, introducing the constraint that $(d-i)+(j-e) < L$ for some constant L. This way, an n^2 computation is limited to $O(L^2)$ which is a constant (although it could be a big one if $L = 30$). That leaves only the multi-loop term to make a constant time computation. Using the heuristic, this is possible. Let us define a new function (and table) $Q^m(i,j)$ which is defined as (this time explicitly summing d and e):

$$Q^m(i,j) = \sum_{d=i}^{j-1} \sum_{e=d+1}^{j} Q(i,d-1)Q^b(d,e)e^{-b(j-e)/RT}. \tag{2.10}$$

We can take the out all the terms which contain $e = j$ to split the formula into two parts (note that I can limit the first term's d sum to only go to $j-2$ because d must

always be less than e):

$$\begin{aligned} Q^m(i,j) = &\sum_{d=i}^{j-2} \sum_{e=d+1}^{j-1} Q(i,d-1)Q^b(d,e)e^{-b(j-e)/RT} \\ &+ \sum_{d=i}^{j} Q(i,d-1)Q^b(d,j). \end{aligned} \tag{2.11}$$

The first term in this new equation is simply $Q^m(i,j-1)e^{-b/RT}$. In the second term, it is always implied that d pairs with j, so we can substitute the summation index for our heuristic for j, $K(j)$, without losing anything. Therefore, our formula becomes:

$$Q^m(i,j) = Q^m(i,j-1)e^{-b/RT} + \sum_{k \in K(j)} Q(i,k-1)Q^b(k,j). \tag{2.12}$$

Now, we can examine the efficiency of computing $Q^m(i,j)$. The first term is a constant factor and the second is a sum over only a constant number of indices by our empirical assumptions about the heuristic. Therefore the computation of Q^m can be treated as a constant time computation.

Now the computation for $Q(i,j)$ becomes:

$$Q(i,j) = e^{-b(j-i+1)/RT} + Q^m(i,j), \tag{2.13}$$

and for $Q^b(i,j)$:

$$\begin{aligned} Q^b(i,j) = &e^{-E_h(i,j)/RT} + e^{-E_s(i,j)/RT}Q^b(i+1,j-1) \\ &+ \sum_{\substack{d,e \\ i<d<e<j \\ (d-i)+(j-e)<L}} e^{-E_i(i,d,e,j)/RT}Q^b(d,e) \\ &+ e^{-a/RT}Q^m(i+1,d-1). \end{aligned} \tag{2.14}$$

So assisted by this heuristic we are able to reduce all table updates to constant

time computations. This means the algorithm is only limited by the number of (i, j)'s which grows with $O(n^2)$.

2.5 Results

Calculating the partition function is still difficult, but it can be improved in the asymptotic case. The results show that the new computation has a large constant factor in the beginning of the algorithm. This could be related to the way the internal loop sum is calculated, if $L = 30$, the constant attached the the loop sum is $L^2 = 900$. This constant term will only stop dominating as the number of bases gets much larger than 900, as we can see by the plot. If we reduce L, we should also see a reduction in probability barrier, we get further reductions in runtime.

2.6 Conclusion

The partition function computation is hard and is a limiting factor in RNA structure prediction. However, given a heuristic to predict the base pairs, we can make the algorithm run much faster. These improvements can be used in applications such as clustering based on our nestedness measure, using the partition function. These improvements also open up several possible branches for further investigation. Can a heuristic be generated before the partition function is computed, so that the pairs can be restricted and the computation done in much faster time than the $O(n^3)$ algorithm? Could this concept be applied to the pseudoknot algorithm to possibly speedup the computation for that as well? Applications of this sort could have tremendous impact on RNA secondary structure prediction, opening up incredible avenues of potential.

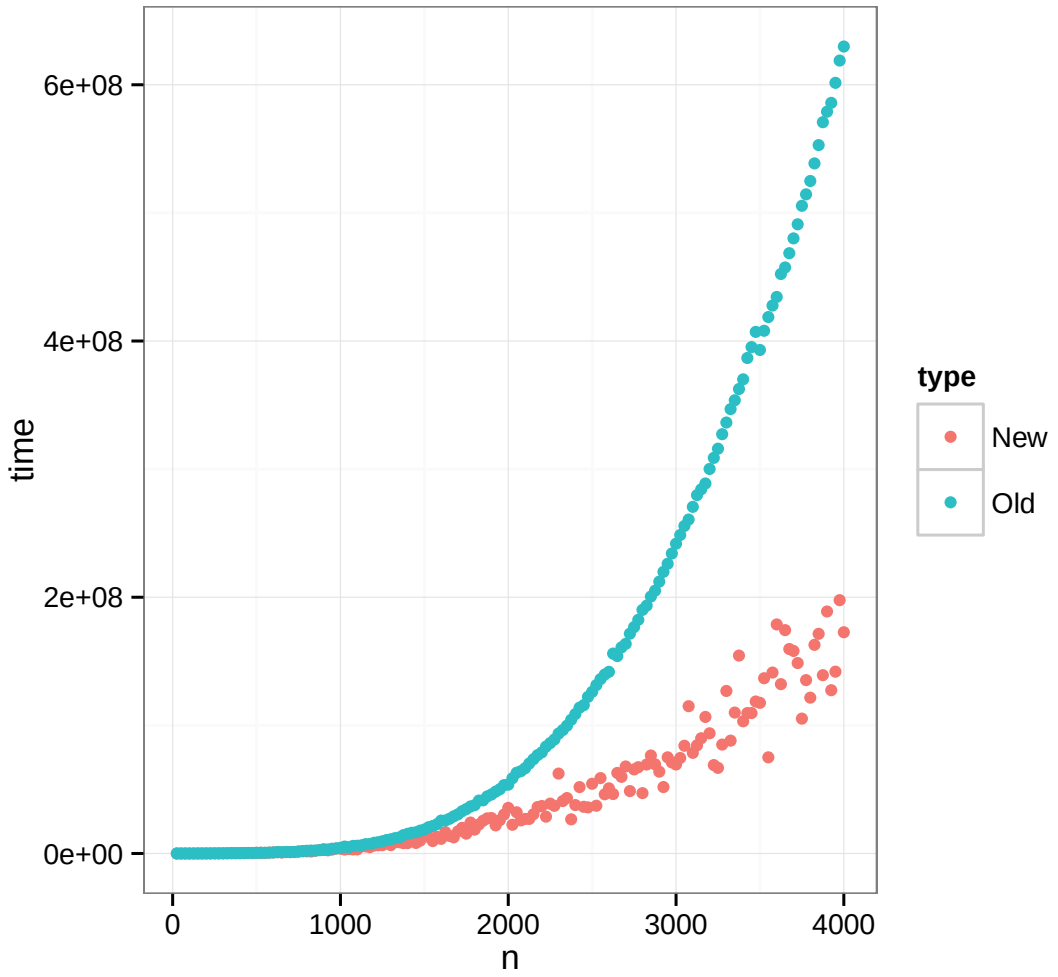


Figure 2-7: Computation time vs length for random sequences, the old $O(n^3)$ partition function run against the new $O(n^2)$ partition function. Not quite the results we were after, however, we can always set the max internal loop length L to a lower value and increase the probability threshold θ.

Chapter 3

Stochastic Traceback Algorithm and Improvements

3.1 Introduction

The stochastic traceback algorithm was introduced by Ye Ding and Charles Lawrence [4] as a means to explore the energy landscape of RNA by sampling structures according to their Boltzmann probabilities. This was important because the minimum free energy structure was very sensitive to errors in the parameters of the free energy model, and although algorithms existed for generating suboptimal structures, they either sampled a very limited set of states [19], or had exponential runtime and the output did not have the same distribution as the physical ensemble of states [**?**]. Structures sampled according to this algorithm can be clustered into macrostates, whos properties can be examined.

The method first computes the partition function, then it "traces back" over the contents of the tables calculated during that algorithm. Specifically, the tables $Q(i, j)$, $Q^b(i, j)$, etc. now contain information about the conditional probabilities of bases pairing. Starting from a bare, undetermined strand at the top of the recurrence

relations, the stochastic traceback samples pairs between bases until a structure is formed such that its probability of it being created by the algorithm is equal to its boltzmann weight $e^{-E(s)/RT}/Z$.

The general principle of the backwards trace is that, presented with several possibilities for the structure along a sequence from i to j, the sampling probability for a case is the contribution to the partition function by that case's partition function. For example, since the partition function $Q(i,j)$ is defined:

$$Q(i,j) = \overbrace{e^{-b(j-i+1)/RT}}^{\text{empty}} + \overbrace{\sum_{\substack{d,e \\ i \le d < e \le j}} Q(i,d-1)Q^b(d,e)e^{-b(j-e)/RT}}^{\text{rightmost pair}}, \tag{3.1}$$

during a stochastic traceback the probability of sampling a pairless strand, and the probability of drawing a rightmost pair (d,e) and recursing from there are:

$$P(\text{empty}|i,j) = \frac{e^{-b(j-i+1)/RT}}{Q(i,j)} \tag{3.2}$$

$$P(\text{pair } (d,e)|i,j) = \frac{Q(i,d-1)Q^b(d,e)e^{-b(j-e)/RT}}{Q(i,j)}. \tag{3.3}$$

Notice that when summed together for every possible case of (d,e), the numerator becomes the definition of $Q(i,j)$, so these probabilities are naturally normalized. Likewise, since $Q^b(i,j)$ is defined as

$$\begin{aligned} Q^b(i,j) = &\overbrace{e^{-E_h(i,j)/RT}}^{\text{hairpin}} + \overbrace{e^{-E_s(i,j)/RT}Q^b(i+1,j-1)}^{\text{stack}} \\ &+ \overbrace{\sum_{\substack{d,e \\ i<d<e<j}} e^{-E_i(i,d,e,j)/RT}Q^b(d,e)}^{\text{internal}} \\ &+ \overbrace{e^{-a/RT}\sum_{\substack{d,e \\ i<d<e<j}} Q(i+1,d-1)Q^b(d,e)e^{-b(j-e-1)/RT}}^{\text{multiloop}}. \end{aligned} \tag{3.4}$$

The probability of sampling a hairpin, stack loop, internal loop, and multiloop are:

$$P(\text{hairpin}|i,j) = \frac{e^{-E_h(i,j)/RT}}{Q^b(i,j)} \tag{3.5}$$

$$P(\text{stack}|i,j) = \frac{e^{-E_s(i,j)/RT}Q^b(i+1,j-1)}{Q^b(i,j)} \tag{3.6}$$

$$P(\text{internal } (d,e)|i,j) = \frac{e^{-E_i(i,d,e,j)/RT}Q^b(d,e)}{Q^b(i,j)} \tag{3.7}$$

$$P(\text{multi } (d,e)|i,j) = \frac{e^{-a/RT}Q(i+1,d-1)Q^b(d,e)e^{-b(j-e-1)/RT}}{Q^b(i,j)} \tag{3.8}$$

Notice for the same reason as before, all the probabilities for the Q^b recursion sum to one. In general, the stochastic traceback can be readily understood by examining the recurrence relation figures from chapter 2: Figure 2-5 and Figure 2-6. We start at the Q recursion for the strand, we may pick a pair and add it to the structure, but in general, for every pair added we recurse down to Q^b on that region to determine the structure below that pair. For every structure left undetermined, we use the probabilities from Q to determine that structure and so on.

The algorithm's implementation uses two stack data structures. A stack can be thought of as a literal "stack", like papers stacked on a desk, except instead of paper their are items of data. There are two basic operations, one to put an item on the top of the stack, and another to retrieve an item off the top. These are called "push" and "pop" operations, respectively, in Computer Science. The data items we will be pushing on to the first stack, A, are of the form $\{(i,j),b\}$ where i and j are indexes along the strand and b is either $true$ if we have determined that i and j are paired, or $false$ otherwise. The second stack, B, is where we'll collect pairs and unpaired bases for one sample.

The algorithm displayed in Figure 3-2 is what gets the job done. The probability of the structure is equal to the product of the probabilities that determined its pairs and empty regions, but if you follow the recursion all the way down you see that this

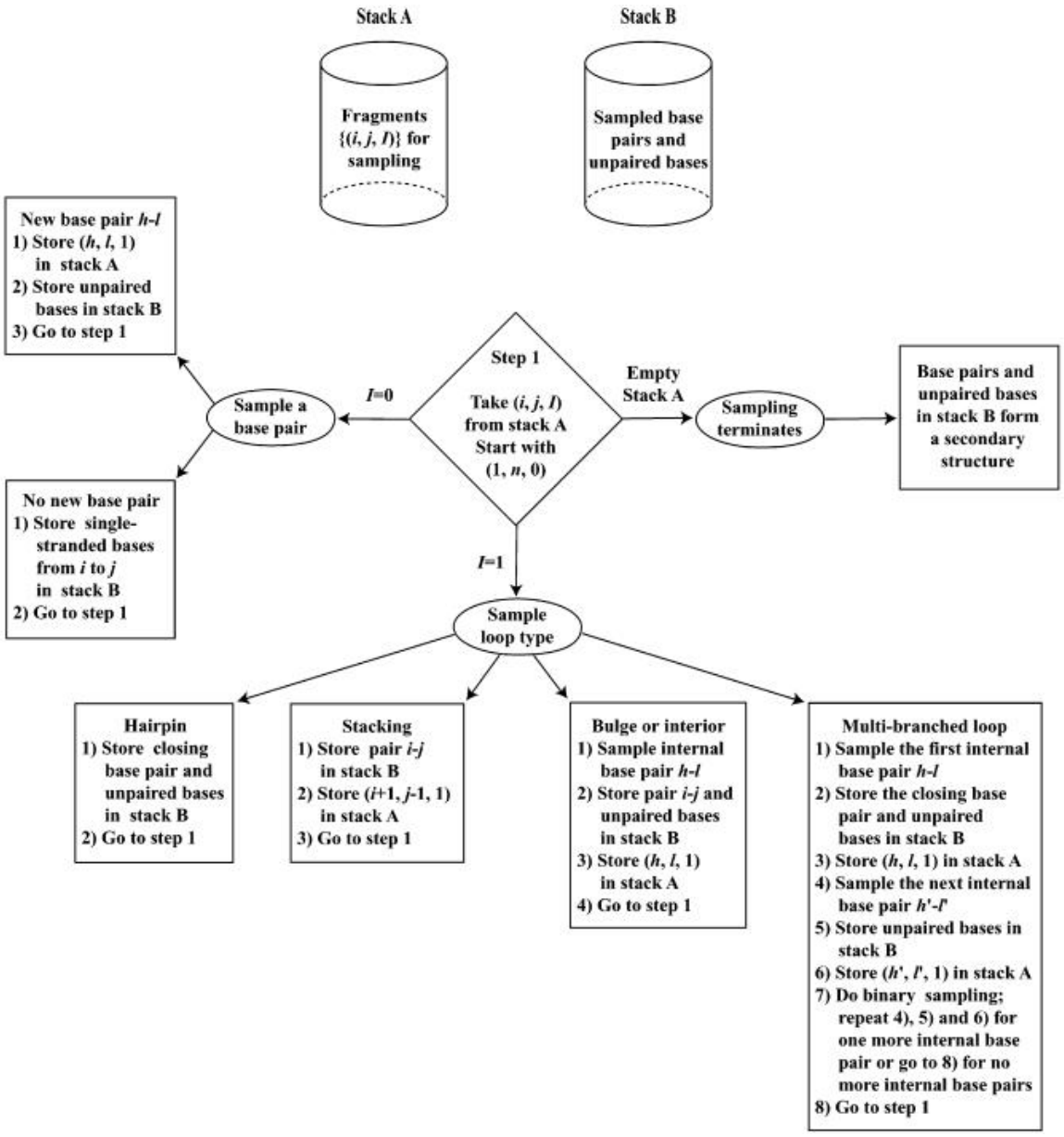


Figure 3-1: Flowchart of the algorithm from Ding and Lawrence's paper [4]. Basically: if we are on a $Q(i, j)$ recursion, sample either an empty section or a rightmost pair. For Q^b choose a loop type and corresponding pairs if applicable. After both cases recurse down appropriately so that Q^b is drawn for any pair created and Q for any open region.

The initialization of the algorithm is to push $\{(1, n), false\}$ onto the stack. From there the algorithm repeats the following steps:

1. **Pop an element, $\{i, j, b\}$ off stack A.**
2. **Case b is $false$:**
   (a) **($Q(i, j)$ recursion) Pick empty or pair (k, l) with probabilities listed above. If empty, push all pairs on $[i, j]$ inclusive onto B as unpaired bases, if not push $\{(d, e), true\}$ and $\{(i, d-1), false\}$ onto A.**
3. **Case b is $true$:**
   (a) **($Q^b(i, j)$ recursion) Choose what type of loop (i, j) is from the probabilities listed above.**
   (b) **Push the appropriate elements onto the stack for that loop type, see Figure 3-1.**
4. **If stack A is empty, the pairs and unpaired bases in stack B become a sampled structure. Reinitialize for additional samples.**

Figure 3-2: The stochastic traceback algorithm is implemented by pushing the recursive elements back onto the stack to sample a full structure.

---

results in just the boltzmann factor for the structure:

$$P(s) = \prod_{\text{cases}} \frac{P(\text{case})}{Q(\text{case})} = \frac{1}{Z} e^{-E(s)/RT}. \tag{3.9}$$

3.2 Motivation

In the past 10 years, the stochastic traceback algorithm has become an increasingly central part of RNA secondary structure prediction algorithms [12]. This is because they present many advantages over the minimum free energy prediction. It can be shown that the minimum free energy state, even though it is the most probable state, can still have astronomically unlikely probabilities on average for typical strands of reasonable length (Figure 2-2). The more important concept in understanding the physical behavior of an RNA strand is therefore the overall shape of the energy landscape. Althogh the probability of any individual structure might be infinitesimally

small, there can be shown to be relatively few large basins containing clusters of similar foldings. The consensus structures and the difference between the consensus structures of these basins define the function of the RNA molecule.

The way the stochastic algorithms probe that is by providing structures to group into these basins, and since the stochastic traceback algorithm samples states with the exact probability defined by the partition function, we know that the macrobehavior of these samples match what we would probably see in reality. There is one catch and that is statistical error. However, the error can be reduced and the landscape can be further explored the more stochastic samples we make.

The need to sample large numbers of secondary structures makes a speedup very convenient, and that is what motivates our current expedition.

3.3 Adding Efficiency

Taking advantage of the empirical fact that the number of probable base pairs for an RNA strand tend to grow very slowly, we can restrict our traceback to only explore bases that we know can pair with one another. This is as simple as replacing the old recurrence relations with the new ones. For example during a $Q(i,j)$ state of the recursion there would be probabilities of:

$$P(\text{empty}|i,j) = \frac{e^{-b(j-i+1)/RT}}{Q(i,j)} \tag{3.10}$$

$$P(\text{goto } Q^m|i,j) = \frac{Q^m(i,j)}{Q(i,j)}. \tag{3.11}$$

For a Q^m state:

$$P(\text{continue}|i,j) = \frac{Q^m(i,j-1)e^{-b/RT}}{Q^m(i,j)} \tag{3.12}$$

$$P(\text{pair } (k,j)|i,j) = \frac{Q(i,k-1)Q^b(k,j)}{Q^m(i,j)}. \tag{3.13}$$

And finally for $Q^b(i,j)$:

$$P(\text{hairpin}|i,j) = \frac{e^{-E_h(i,j)/RT}}{Q^b(i,j)} \tag{3.14}$$

$$P(\text{stack}|i,j) = \frac{e^{-E_s(i,j)/RT}Q^b(i+1,j-1)}{Q^b(i,j)} \tag{3.15}$$

$$P(\text{internal } (d,e)|i,j) = \frac{e^{-E_i(i,d,e,j)/RT}Q^b(d,e)}{Q^b(i,j)} \tag{3.16}$$

$$P(\text{goto } Q^m|i,j) = \frac{e^{-a/RT}Q^m(i,j)}{Q^b(i,j)}. \tag{3.17}$$

3.4 Results

As one can see from Figure 3-3, the speedup is enormous. For randomly sampled sequences up to lengths in the thousands, the old stochastic timing grows quadratically, while the new method flatlines below it. The log-log plot in Figure 3-4 shows that the algorithm is not quite $O(n)$, this is probably because there is a probability to recurse down different cases of Q^m, however it is still quite below the old slope. A good question to ask would be, how do we know that this new algorithm is outputting structures with the correct probabilities. Verification plot Figure 3-5 attempts to answer that question.

What we would expect to see from these plots, is that for a given base, we would expect to see it pair with other bases with probabilities given by the partition function as one can see. Of course there is sampling error, so each bin represents a sampling from a Bernoulli distribution. For n^2 samples, we would expect [Todo: find out what error we expect] error. The number of samples that violate the bounds, do not deviate much from what we would expect from doing n^2 experiments, so I think we can confidently say that the new algorithm is making the correct computation.

3.5 Conclusion

The stochastic traceback algorithm can be sped up to allow large amounts of sampling such that the limiting computation factor is memory, not time. The applications of this improvement are that statistical algorithms like Ding & Lawrence [**?**] [3] can minimize their statistical error. Nestor [**?**] benefits greatly from these improvements because more structures mean more branches to the tree. In general the algorithm is useful because it removes unnecessary computation that was done before.

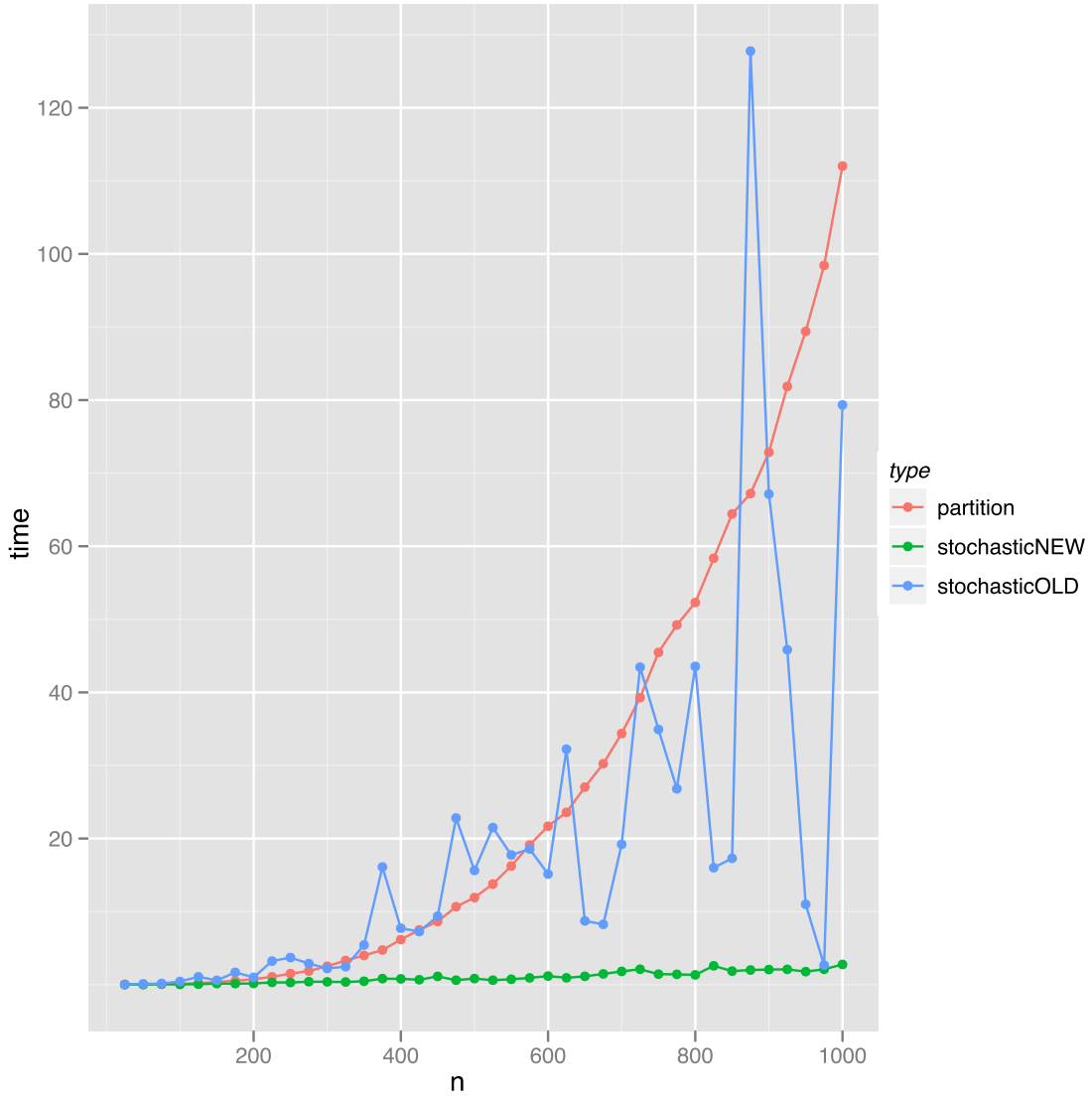


Figure 3-3: A comparison of the computation time between the partition function computation, and stochastic tracebacks of 1000 samples with the Old algorithm and the new algorithm. The new algorithm is much faster.

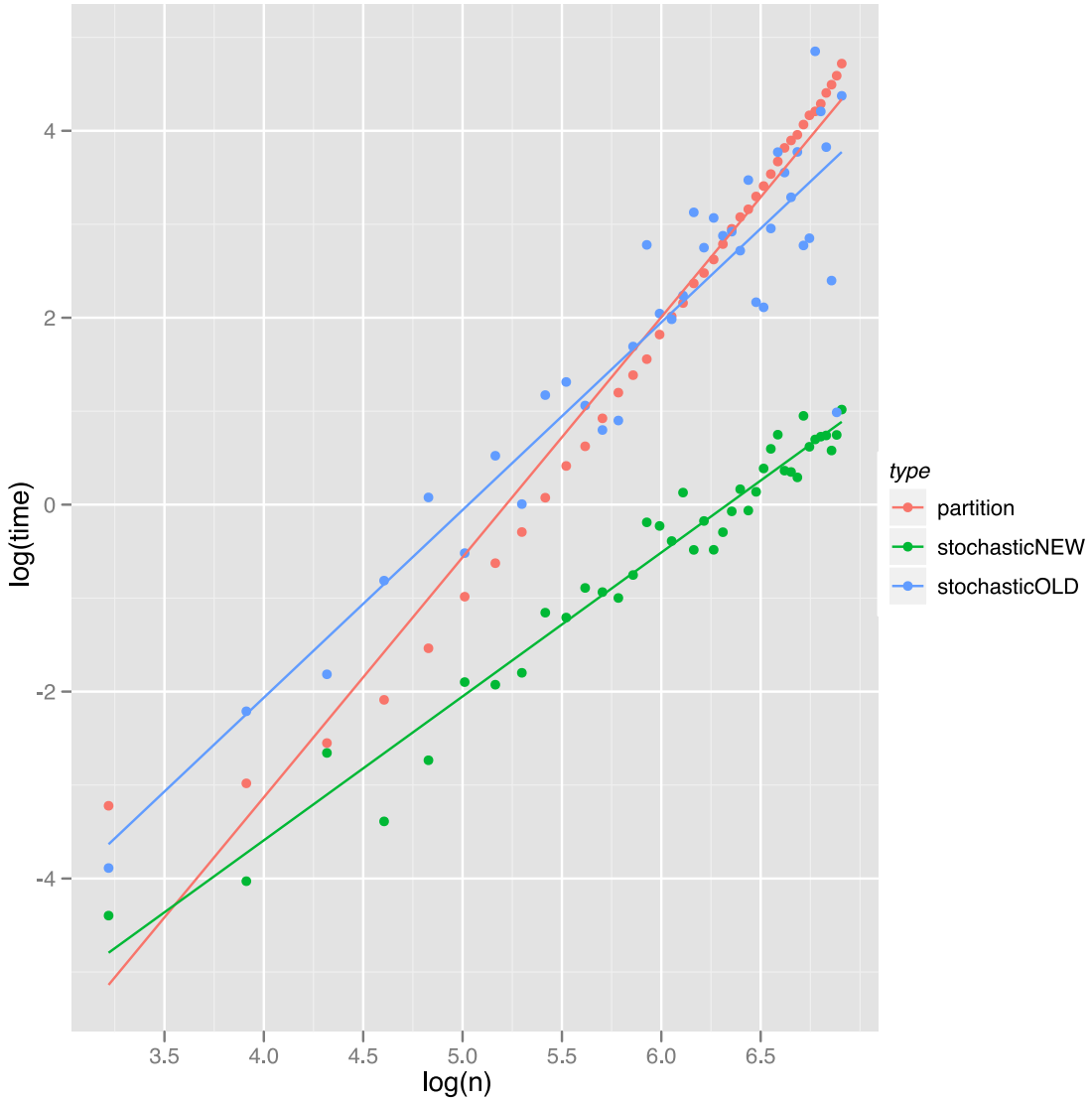


Figure 3-4: Log-Log plots of the timings in Figure 3-3. The new stochastic algorithm slope is slightly higher than 1, but not much.

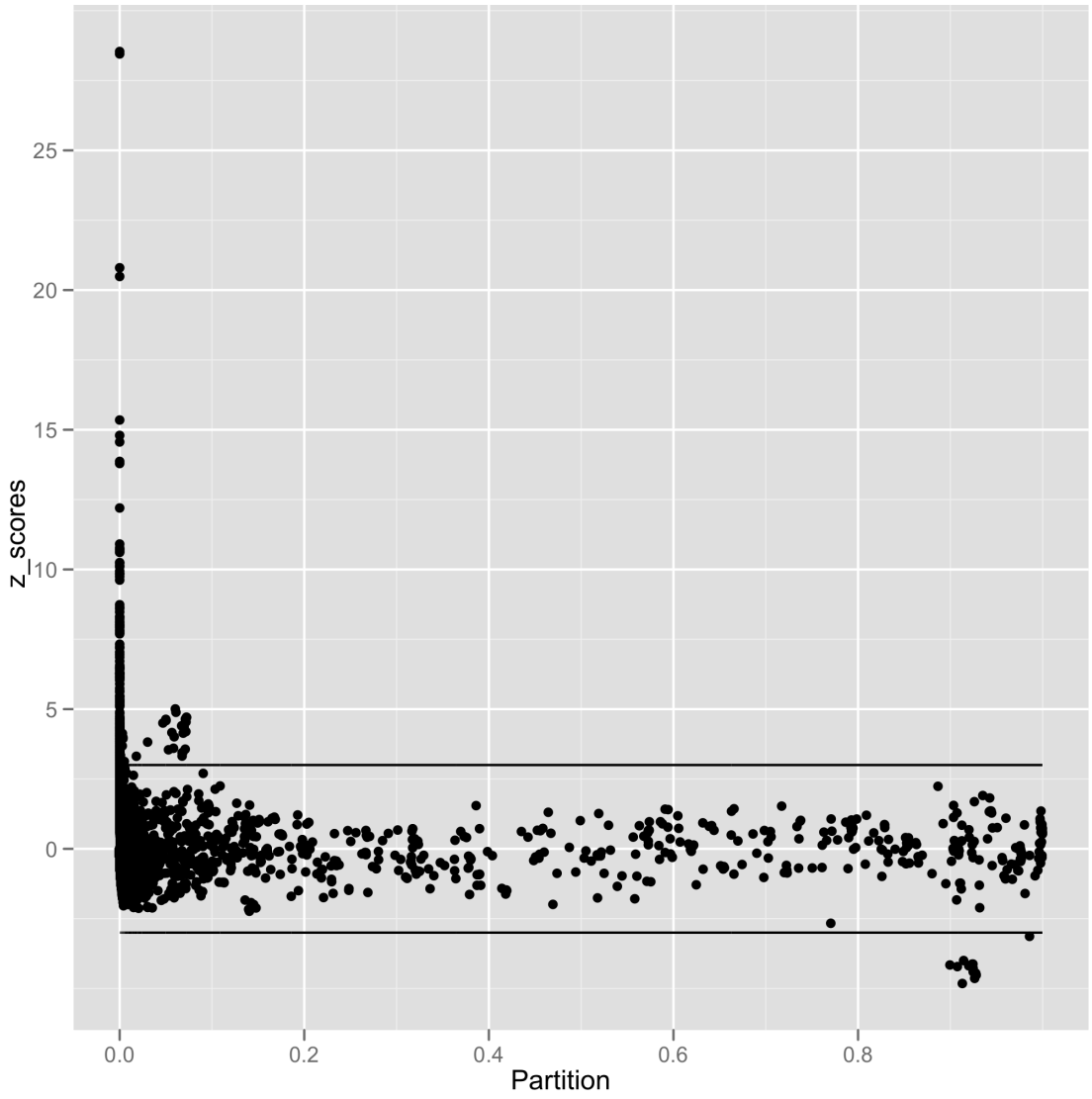


Figure 3-5: A plot of the difference between the partition function probability and the stochastic traceback probability of a base pair according to the new algorithm. According to the number of states, there should be 30 states above 0 probability that cross the horizontal lines, there are less than 30.

Chapter 4

Macrostates

As discussed in Chapater 2, there is signficant motivation to not think of RNA as a typical energy minimizing system. A ball rolls to the bottom of a valley, if it is smooth. However, the energy landscape of RNA is shallow and bumpy: the deepest point (MFE) can be around 10^{-10} times the total partition function, with many local minima for example in Figure 4-1. In 2005, Ding and Lawrence showed that the Boltzmann distribution for RNA naturally parititions into distinct clusters [3] as seen in Figure 2-3. Therefore, it is better to think of RNA as a thermodynamic system in equilibrium between several different macrostates, defined:

Macrostate: a set of states of RNA containing similar structural qualities, or within some metric of energy distance from each other.

The defintion of Macrostate is left vague, because there currently is no widespread agreement on how to identify and classify them. If the set of states defining a macrostate is S, then the probability of a macrostate is defined as

$$P(S) = \sum_{s \in S} P(s). \tag{4.1}$$

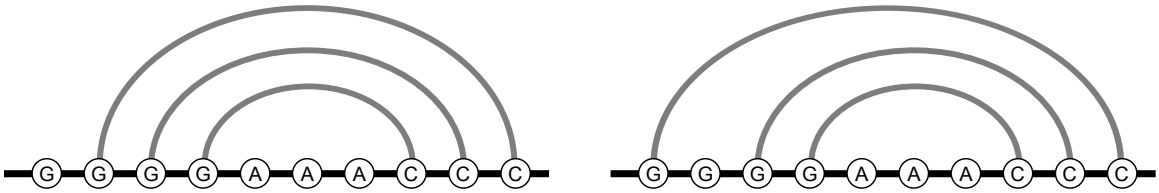


Figure 4-1: Here are two local minima of the same strand, "GGGGAAACCC". To get from one to the other, the outermost bond must be broken, raising the energy. There are multitudes of such local minima, and this contradicts any idea of a "smooth" decent to the global minimum.

This can be thought of as integrating "infinitesimals" ($P(s) \ll Z$) over an energy "basin".

In the past 10 years, several groups have started to explore this concept. There are two approaches, in general, to define basins and classify structures into them. The first class of methods defines the basins from the top-down: given a number of stochastically sampled structures, we divide them into groups based on some kind of distance metric. These methods tend to be very similar to typical clustering algorithms used in computer science and data analysis.

Another approach is to start at local minima and climb up the energy barriers between minima. These methods can be used to accurately compute the energies of the transition states between local minima and these can tell you the kinetics of the structure. This method was devloped by the makers of ViennaRNA and continues to be a main area of research [10] [17] .

In general, finding macrostates, estimating their size and where they overlap will tell you the probability of that macrostate and the transition probabilities between them. This information provides a much greater understanding of the function of an RNA strand than standard minimum free energy analysis.

4.1 Sfold

The author's of the stochastic traceback paper **??** Ding and Lawrence explored clustering the results of their algorithm and found that it not only improved the prediction accuracy **??**, but indicated that there should be a revolutionary change of perspective in terms of the mechanics of RNA. Specifically they found that 45% of sequences they studied did not have the MFE in the biggest, most probable cluster. If the MFE is not in the most probable cluster, any predictions based on the MFE for that strand are going to be inaccurate. They confirmed this and found that the MFE predictions perform the worst in these kinds of strands. Another important finding was that in 45% of sequences do not have a macrostate with greater than 70% probability, but that most sequences do not have more than 3 or 4 clusters. If this is true, this means that there must not be a single dominant structure for that RNA strand, but rather that it is in equilibrium between macrostates, and these different states and the transitions between them have critical biological roles.

Ultimately the change in perspective to clustering analysis improved the results of their prediction. They compared their results to secondary structures that have been determined by comparative sequence analysis and looked at the results for the MFE, the biggest cluster, and the cluster that performed the best. The biggest cluster centroid and the best cluster centroid beformed better than the MFE in 74.1% and 91.4% of sequences, respectively, where performance was measured by their base pair distance metric from the comparative sequence structure (see Sfold Methods) [3] [12]. The centroids also had less errors in measures such as the positive predictive value (PPV), defined as the rate of true positives, improving to up to 30% in the best cluster centroid. One interesting finding they found was that the natural sequence found by comparative sequence analysis was not necessarily in the biggest cluster. This could be because the strand is not necessarily in thermal equilibrium, rather assembles itself in a macrostate that is easily accessible from the unfolded state (low energy barrier)

The algorithm starts with structures all in the same cluster and repeats the following sequence:

1. **Find the structure s_m with highest average distance $D(s_m, s_n)$ for all s_n in the same cluster and create a new cluster containing just s_m.**
2. **For each structure s_n that is closer to s_n than the rest of the structures on average, add it to the cluster containing s_m.**
3. **For the 2 new clusters created by this algorithm, unless stopping criterion has been met, repeat this algorithm on each.**

Stopping criteria: CH-index

$$CH(k) = \frac{B(k)/(k-1)}{W(k)/(n_{total}-k)}$$

is maximized, where k is the number of clusters, $B(k)$ is between-cluster sum of squares, $W(k)$ is average in-cluster sum of squares, and n_{total} is the total number of structures.

Figure 4-2: The DIANA algorithm for dividing data into clusters based on a distance measure D. Used by Sfold to divide structures into macrostates.

---

but that has not as much access to the most Boltzmann probable macrostate (high energy barrier). This puts even more emphasis on knowing the transition energies between states.

4.1.1 Sfold Methods

Sfold clusters RNA structures that have been stochastically sampled according to their Boltzmann probabilities using the DIANA clustering algorithm, using a base pair metric as the distance. Defining I_{ij}^m to be a binary value that is 1 if secondary structure s_m contains the base pair (i, j) and 0 otherwise, the metric is defined as

$$D(s_m, s_n) = \sum_{j,k} (I_{jk}^m - I_{jk}^n)^2, \tag{4.2}$$

or roughly adding 1 for every base pair that s_m and s_n do not share. Using this distance metric, the DIANA clustering goes as described in Figure 4-2, budding off structures by finding the most different structure, and tries to maximize the ratio of between average between-cluster distance and average in-cluster distance.

The cluster centroids, s_s, are found within each cluster created by DIANA analysis by creating an upper-triangle matrix I^s_{jk} that solves the linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_m D(s_s, s_m) \; = \sum_{m,i,j} (I^m_{ij} - I^s_{ij})^2 \\ \text{subject to} \quad & I^s_{ij} + I^s_{kl} \leq 1 \qquad \forall (i < k \;\&\; k < j) \\ & I^s_{ij} \in \{0, 1\} \qquad \forall i, j. \end{aligned} \tag{4.3}$$

[TODO: should I delete and just say "solve a linear program"?] This produces a valid structure that is as close as possible to the other structures in the cluster.

4.1.2 Drawbacks

While the base-pairs distance metric is useful and efficient from a computer science standpoint, a better distance measure would be based on the energy cost to get from one structure to the other. Then distances could be related to other thermodynamic variables, such as time constants of transitions, and would be a more physically based method of determining energy states. While base-pair differences are related to the energy needed to go from one structure from another, they do not directly correspond and so can lead to some slight discrepencies. In their paper *Comparing RNA secondary structures using a relaxed base-pair score*, Agius, Bennet, and Zuker examine how it is possible to construct 3 secondary structures for the strand where the last 2 are the same base-pair distance from the first but one looks similar and the other is very different. They go on to develop a different distance metric they call *relaxed-base-pair* [2]. Aalberts and Jannen developed another way of measuring

distance, explained in the Nestor section.

One other drawback is that there is no good way of measuring the transition state between two macrostates. The way the clustering algorithm works allows for no "in between". The PF clustering method, a slight modification of the Nestor algorithm, allows both a better distance metric and an ability to measure transtition states. Another method, which deserves mentioning in my opinion, was developed by the team of ViennaRNA, which they called Barriers.

4.2 Vienna RNA: Barriers

Barriers is a tool designed by those behind the software package ViennaRNA to compute macrostates by starting at individual local minima and climbing up the free energy basins by making and breaking pairs [6] [17]. Macrostates are thereby defined as the set of states with the same local minima. This climbing is done by using a physically accurate stochastic simulation, and this allowed them to compute the transition probabilities between macrostates by examining the probability of transitioning from a state with one local minima to a state with another local minima. Here I will be following the results of their most recent work, *Efficient exploration of discrete energy landscapes* by Mann and Klemm (2011) and *Memory-efficient RNA energy landscape exploration* by Mann *et al* (2014) [10] [11], where they develop a clear and precise formalism.

4.2.1 Algorithm

The algorithm uses the Metropolis-Hastings framework for simulating the thermal process of RNA using a markov chain [15] [7]. Specifically, they define the energy landscape of RNA to be a triple (X, E, M) where

- X is a finite set of states,

The algorithm starts by "seeding" n minima from X using Ding and Lawrence's stochastic traceback algorithm and then a gradient decent using M. From there, for each state x:

1. **Find basin $b = B(x)$ (trivial on first step).**
2. **Update the thermodynamic variables (see equations 4.6 and 4.8):**
   (a) $Z = Z + w(x)$
   (b) $Z_b = Z_b + w(x)$
   (c) **For each** $y \in M(x)$**:** $Z_{\{b,B(y)\}} = Z_{\{b,B(y)\}} + \min\left(\frac{w(y)|M(x)|}{|M(y)|}, w(x)\right)$
3. **Find next state y by sampling uniformly from $M(x)$ and accepting $x = y$ according to probability**

$$p_{x \to y} = \min\left(\frac{w(y)|M(x)|}{w(x)|M(y)|}, 1\right),$$

   (equation 4.4), otherwise x stays the same.
4. **Go to step 1, unless sufficiently satisfied.**

Results: Probability of any macrostate b is Z_b/Z, transition probability from state b to c is $Z_{\{b,c\}}/Z_b$.

Figure 4-3: The algorithm by Flamm *et al* as it appears in the most recent papers. Estimating macrostate probabilities and transition probabilities by direct stochastic simulation. Notation: $w(x) = e^{-E(x)/RT}$, $B(x)$ is the local minima found from x by doing gradient descent on M.

- $E : X \to R$ is an energy function on X (the Turner model),
- $M : X \to \mathcal{P}(X)$ is a neighborhood function that assigns to each state $x \in X$ the set of states that could be reached from x by breaking and making a single pair and including x itself. $\mathcal{P}(X)$ is the power set of X.

The algorithm is described as in Figure 4-3. Local minima are found from a stochastic sample of the Boltzmann distribution. These minima are then used as the starting points of markov chains generated by the Metropolis algorithm. Specifically given a structure x, a structure y is chosen by uniformly sampling from $M(x)$, and that

structure is accepted as the next state of the chain with probability

$$p_{x\to y} = \min\left(\frac{P(y)T(y\to x)}{P(x)T(x\to y)}, 1\right) = \min\left(\frac{e^{-E(y)/RT}|M(x)|}{e^{-E(x)/RT}|M(y)|}, 1\right), \tag{4.4}$$

where $P(x)$ is the standard Boltzmann probability of the state x, $e^{-E(x)/RT}/Z$, and $T(x \to y)$ is the probability of generating the transition from x to y, which is $1/|M(x)|$ because we uniformly chose from $M(x)$. If the new state is not accepted, x is kept as the next state in the chain. Equation 4.4 fulfills the requirements of "detailed balance" which ensures that the states of the markov chain will converge to the distribution $P(x)$ according to Metropolis *et al* and Hastings [15] [7].

For each state sampled in this manner we can compute it's local minima and compute the thermodynamic variables of that basin on the fly. Let $B \subset \mathcal{P}(X)$ be the set of all macrostates on the free energy landscape. For an individual basin $b \in B$, we can define the probability of the basin as:

$$P(b) = \sum_{x\in b} \frac{e^{-E(x)/RT}}{Z} = \frac{Z_b}{Z}, \tag{4.5}$$

where

$$Z_b = \sum_{x\in b} e^{-E(x)/RT} \tag{4.6}$$

is a quantity we can continuously update as the sample more states x. We can additionally now define a quantity $P_b(x) = e^{-E(x)/RT}/Z_b$ which is the probability of state x within basin b. To compute the transition probabilities between two basins b and c, $q_{b\to c}$ and $q_{c\to b}$, we simply use a weighted sum of the probabilities of transitioning

from states in one to the other (using shorthand $w(x) = e^{-E(x)/RT}$):

$$\begin{aligned} q_{b\to c} &= \sum_{x\in b}\left(P_b(x)\sum_{y\in M(x)\cap c} p_{x\to y}\right) \\ &= \sum_{x,y}\frac{w(x)}{Z_b}\min\left(\frac{w(y)|M(x)|}{w(x)|M(y)|},1\right) \\ &= Z_b^{-1}\sum_{x,y}\min\left(\frac{w(y)|M(x)|}{|M(y)|},w(x)\right) \\ &= Z_b^{-1}Z_{\{b,c\}}, \end{aligned} \tag{4.7}$$

defining a new quantity

$$Z_{\{b,c\}} = \sum_{x\in b}\sum_{y\in M(x)\cap c}\min\left(\frac{w(y)|M(x)|}{|M(y)|},w(x)\right). \tag{4.8}$$

Notice that both the quantities Z_b and $Z_{\{b,c\}}$ can be updated with each new state x, as done in Figure 4-3. At the end of the computation, the probabilities of each macrostate b can be computed by the term Z_b/Z, and the transition probability from b to c is $Z_{\{b,c\}}/Z_b$.

4.2.2 Coarse Graining

One of the drawbacks of Barriers is that there are a great many local minima in the landscape of RNA, as shown in Figure 4-1. To get good estimates of the thermodynamic variables for each of the basins, the algorithm will exibit a very long convergence time since very many states need to be sampled. One way to combat the multiplicity of local minima is to use on of the results of Ding and Lawrence, which is that they observed at maximum 3 or 4 major clusters in the landscape. Similar states with low energy barriers ($q_{b\to c}$ is very high) can be merged into one macrostate, the new thermodynamic parameters can just be added ($Z_{a=b\cup c} = Z_b + Z_c$, $Z_{\{a,d\}} = Z_b Z_{\{b,d\}} + Z_c Z_{\{c,d\}}$) since each macrostate is disjoint by definition, this is called

coarse-graining and was implemented in Barriers as well.

4.3 Nestor and RNAbows

There are seperate clustering methods developed in published and unpublished work by Aalberts and Jannen based on the concept of non-nestedness [1] [**?**]. Non-nestedness is a measure of how much crossing there is between pairs of structures. Non-nestedness is a physically based measure, since to get from one state to another, they will have to break at least all non-nested pairs. Clustering analysis, similar to the analysis done in Sfold can be done with this measure of distance.

To be precise, non-nestedness can be defined either in a sampled ensemble or on partition function data. The partition function version was developed as part of this thesis and will be explored in the next section. For the former, large numbers of sequences are sampled using the stochastic traceback algorithm. The probability of each pair P_{ij} is computed by counting the occurence in each strand in the ensemble. Then the non-nestedness between two ensembles μ and ν is defined as:

$$\Psi^{\mu\nu} = \sum_{k,l} \sum_{m,n \times (k,l)} P^{\mu}_{kl} P^{\nu}_{mn}, \tag{4.9}$$

where $m, n \times (k, l)$ denotes the pairs (m, n) that *cross* (k, l).

4.3.1 Algorithm

The goal of the algorithm is to find the most non-nested pair in the ensemble, or (k, l) such that

$$\max_{k,l} \sum_{m,n \times (k,l)} P^{\mu}_{kl} P^{\nu}_{mn} \tag{4.10}$$

and then split the ensemble into an ensemble that includes that pair and an ensemble that forbids that pair. This splitting is continued recursively until chosen stopping

rules have been fulfilled. A standard stopping rule, one that is currently used in the (unpublished) software is to set a number of output clusters desired, preferrably a power of 2.

The probability of each output cluster is simply the number of sampled structures contained within it divided by the total number of sampled structures. Unfortunately, similar to Sfold, there is no way to estimate the transition states for the divided clusters. However the PF version of Nestor can measure transition states as well as give exact probabilities for macrostates, something that Nestor cannot do because of statistical error. In this way the PF method is an improvement over the stochastic method of Nestor.

4.3.2 Visualization

A nice method of visualizing and comparing two different subensembles is implemented in RNAbows. RNAbows takes in two different ensembles and calculates pairs info about each. Each subensemble is given a color, and circular arcs are drawn in between pairs of bases, where the color and alpha is weighted by their probability. Bases that are shared between the two ensembles are colored black. An example of an RNAbow is in Figure 4-4.

4.4 PF method

The PF method is really just a small modification of the Nestor algorithm. Instead of using a sampled ensemble of states, we compute the partition function and its associated matrices $Q(i,j)$ and $Q^b(i,j)$. The probability of a pair can be computed from a pretty complicated formula given in McCaskill's paper [14] [TODO: find the

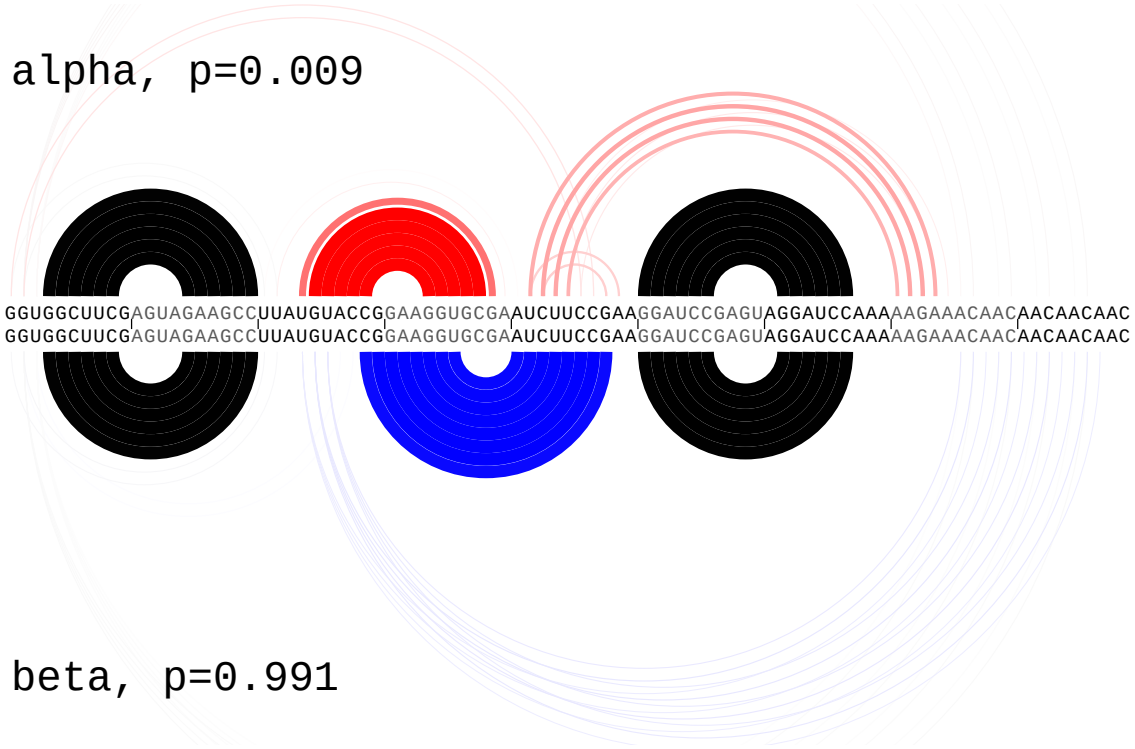


Figure 4-4: An RNAbow of the two main clusters of a strand of RNA. The two colors denote the two differnt clusters, and how solid the colors of the arcs are indicates how probable that base pairing is. The bottom sequence dominates the top sequence with 99% Boltzmann probability. However, if the structure starts in the red state, it might require some energy to get to the blue state. The two ensembles are most non-nested in the middle region, and in order to get from one to the other one must at least break every pair where the red heavy stem crosses the blue heavy stem.

---

better formula and substitute]:

$$P(h,l) = \frac{Q(1,h-1)Q^b(h,l)Q(l+1,n)}{Q(1,n)} + \text{ ... overhead terms} \tag{4.11}$$

From here we can find the most non-nested pair in this ensemble. Once we have obtained this pair (h,l) we can compute an ensemble with all the pairs crossing (h,l) restricted. From there we have an ensemble μ, which contains the structures nested with (h,l). To compute the other ensemble ν, we "flip" the probabilities of μ with the formula

$$P_{i,j} = p^{\mu} P^{\mu}_{ij} + p^{\nu} P^{\nu}_{ij}, \tag{4.12}$$

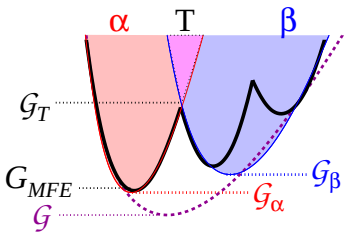


Figure 4-5: What we try to isolate in Nestor. The largest basins are identified by identifying the most non-nestedpair. These basins might contain several ViennaRNA-Barriers-type basins. The top down approach lets us look at the most important basin/clusters efficiently.

---

where p^μ is the macrostate probability of μ and P_{ij}^μ is the probability of the pair (i, j) occuring withing macrostate μ. Noting that we can approximate $p^\nu = 1 - p^\nu$, we have that

$$P_{ij}^\nu \approx \frac{P_{ij} - p^\mu P_{ij}^\mu}{1 - p^\mu}. \tag{4.13}$$

These would be an equality if μ and ν were completely disjoint, however we know there is an overlap region between them that corresponds to the transition state. To estimate this transition state, we can recompute the partition function for the ensemble under the assumption that pairs with very small estimated P_{ij}^ν will not occur, so we restrict these pairs. This new partition function computation gives us the exact P_{ij}^νs. From here, we know the probability of each macrostate is simply the partition function for that macrostate divided by the original partition function Z_μ/Z. We can also estimate the transition probability by the amount of overlap between the two states:

$$Z_{\{\mu,\nu\}} = Z_\mu + Z_\nu - Z, \tag{4.14}$$

$$P(\mu \to T) = \frac{Z_{\{\mu,\nu\}}}{Z_\mu}, \tag{4.15}$$

$$P(\nu \to T) = \frac{Z_{\{\mu,\nu\}}}{Z_\mu}, \tag{4.16}$$

where T is the transition state. Notice that the transition states are defined slightly differently than Barriers because we have defined that macrostates slightly differently. These probabilities can be used to estimate the energy gap needed to be crossed in order to get from one state to another, and can also be used as an automatic stopping rule to be used instead of just explicity stating the number of splits.

4.5 Testing Nestor PF predictions using SHAPE experiments

It would be very good to verify all these predictions against real world experiments. This is not as straightforward as it sounds, as the Williams students who take Applications of Quantum Mechanics learn for example, a lot of care must be taken to ensure that a single photon experiment actually has one photon going through at a time. For a theoretical biologist this task is exascerbated by the environment in which the most important measurements must take place: inside living things. For example, the partition function of an rna strand describes its statistical properties in thermal equilibrium. However, the living cell is certainly not in thermal equilibrium. Furthermore, most measurements of RNA structure necessarily take RNA out of its natural environment in order to examine it.

Selective 2'-hydroxyl acylation analyzed by primer extention (SHAPE) experiments have shown great potential in showing highly accurate profiles of base-pairing in diverse solvent environments [**?**]. A colloquial description of these experiments would be as follows: polymers are prepared that bind to RNA at unpaired regions and an enzyme is added that cuts the the polymers to lengths that are proportional to the index of the base that they are bound to. These polymers are then strained using gel electrophoresis which causes different length polymers to seperate from one another. The counts of polymers at different lengths reveal which bases are unpaired

along the secondary structure of the strand. This information can be used to inform standard secondary structure algorithms, and was implemented by Katherine Deigan, David Mathews, and Kevin Weeks in the RNAStructure package as an additional energy term and saw accuracy improve up to 96-100% [**?**].

4.5.1 Mutate and Map experiments

The experimental methods used to test our theoretical results were those recently established by Das Lab at Stanford called "mutate and map" experiments [**?**]. The group recognized that RNA exists in equilibrium between macrostates and attempted to experimentally examine these states by disrupting pairs through mutation. Specifically, for a sequence n nucleotides long, Das Lab produced $n+1$ sequences, sequence 0 being the normal sequence, 1 being the sequence with base 1 flipped to its opposite pair, 2 the sequence with base 2 flipped to its opposite pair, all the way up to sequence n with base n flipped to its opposite pair. Das Lab then performed shape analysis on each of these strands and found which bases were paired and unpaired. An example of a mutate and map experimental output is in Figure 4-6.

A goal of this analysis is to compare the output of a mutate and map experiment with the output of Nestor PF. There are a couple obstacles in the way of this: reactivities, the output of the experiments, are not normalized on a [0,1] scale, in fact the reactivity is distributed somewhat like a power law distribution $1/r$. Another is that there is no direct way to infer clusters from the mutate and map experiments. To solve these analysis challenges, we developed a normalization strategy for reactivity and a method of cluster analysis on the data.

The specific analysis focused on was on a single sequence, the Hobartner Bistable Switch.

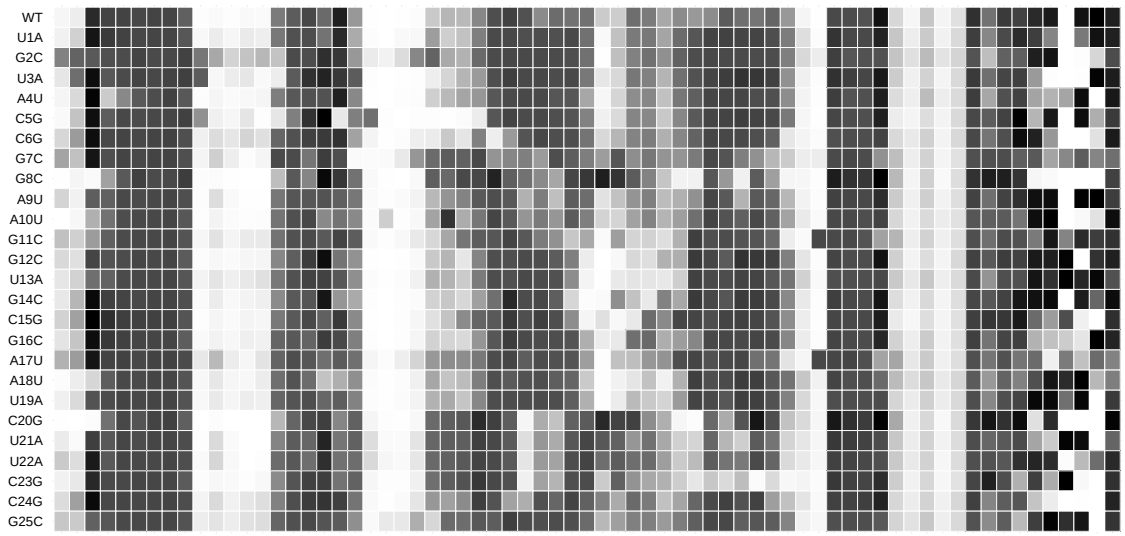


Figure 4-6: Mutate and map experimental outcome. Disruption can be seen in rows G7C, G8C, and C20G through C23G.

4.5.2 Normalization Strategy

The central assumption made by our normalization strategy is that the distribution of reactivities follows the same ranking as partition function probabilities of pairing. Explicitly, we assume that, for bases i and j, if in terms of partition probability of pairing $P(i) > P(j)$, then reactivities $r(i) > r(j)$ and vice versa. We also have a theoretical distribution for the probabilities, $P(i)$. Because of this, we chose to normalize the reactivities by rank sorting them and mapping them onto rank sorted partition function probabilities (see Figure **??**.

4.5.3 Cluster Data Analysis

The mutate and map data from the bistable switch was modeled as "superpositions" of two macrostates to get the reactivities of each mutatant. Specifically, if the two macrostates are μ and ν, they each have probabilities P_i^μ and P_i^ν for each base i being paired, and then each mutant has a certain probability for each macrostate p_m^μ and

p_n^ν. Putting the model together looks something like:

$$\begin{bmatrix} p_1^\mu & p_1^\nu \\ p_2^\mu & p_2^\nu \\ \vdots & \vdots \\ p_m^\mu & p_m^\nu \end{bmatrix} \begin{bmatrix} P_1^\mu & P_2^\mu & \dots & P_n^\mu \\ P_1^\nu & P_2^\nu & \dots & P_n^\nu \end{bmatrix} + E = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix}, \tag{4.17}$$

where the r_{mn}'s are the mapped reactivities and E is the model error matrix. The parameters, the p_m^μ's and the P_i^μ's, are fit to this model using a boxed gradient descent from the R *optim* package [**?**].

4.5.4 Results

For both reactivity fit and the partition function theory, we classify pairs above .25 probability as predicted, and not predicted otherwise. Using this analysis I found the around 55% of base pairs were properly matched between the SHAPE analysis and Nestor predictions. This does not seem like much because the prediction algorithms when compared to crystal structures get better accuracy, of up to 70%. However, this is the first time this analysis has been done, so it is pretty amazing we were able to get any pairs at all. See Figures 4-7 and 4-8 to see how well they match up.

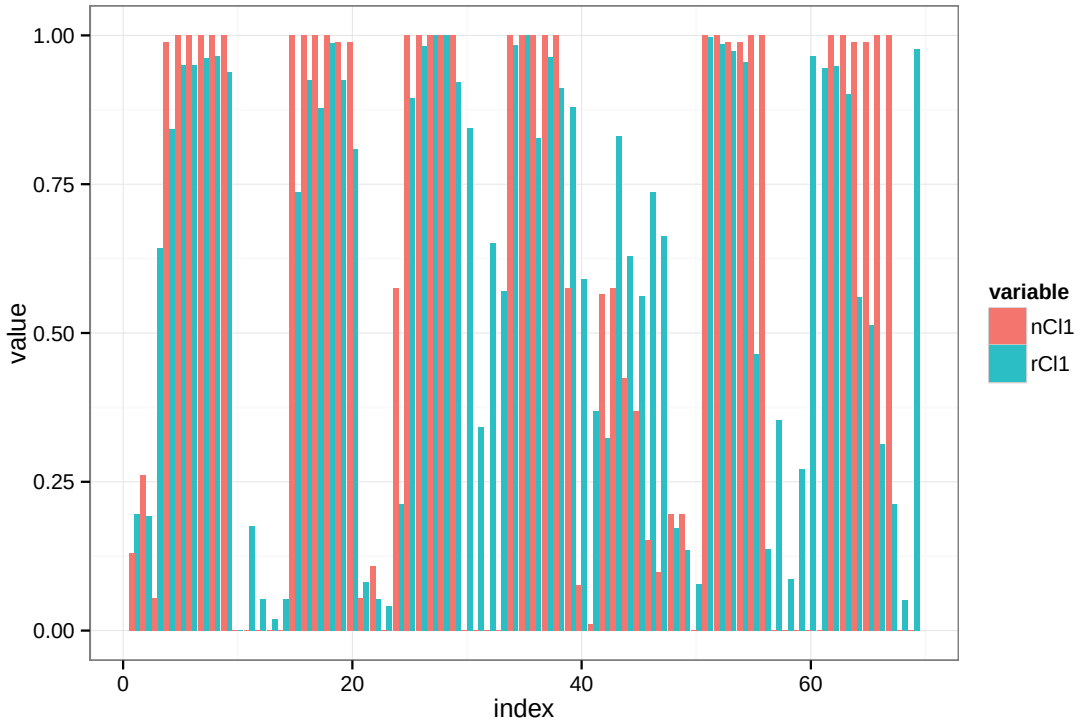


Figure 4-7: The results of comparing the first cluster results of nestor "nCl1" against the cluster analysis on SHAPE data "rCL1". The x axis is base index and the y axis is probability.

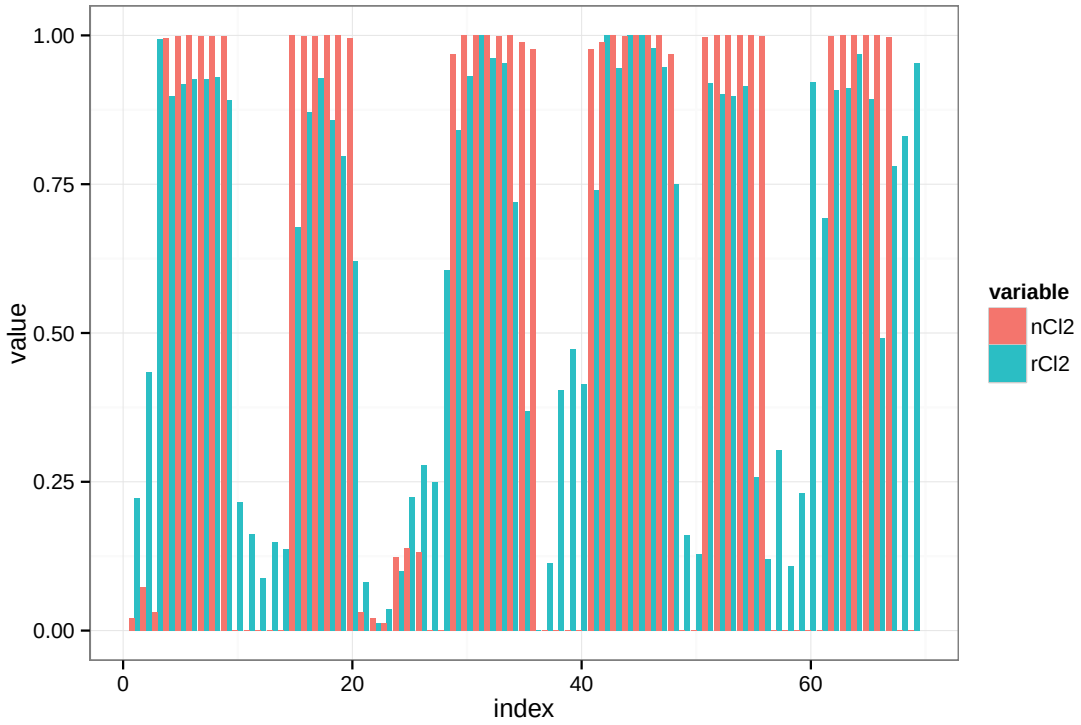


Figure 4-8: The results of comparing the second cluster results of nestor "nCl2" against the cluster analysis on SHAPE data "rCL2". The x axis is base index and the y axis is probability.

Chapter 5

Conclusion

My hopes for this thesis is that it will be of use to someone interested in doing RNA structure research in the future. Predicting RNA structure is a very interesting and difficult problem to solve, but it also contains a variety of open problems that someone new to the field can jump in and solve. I was very lucky to have been able to jump in and take some of the low hanging fruit. In addition, I think my research has opened up many more opportunities for future researchers to grab more low hanging fruit.

One of these is combining Nestor and Barriers. Nestor is a great top down approach that focuses on the highest energy conflict between states to define the macrostates, and Barriers is a very good bottom up approach that defines physically accurate and conceptually complete thermodynamic variables. Combining the two approaches could be fruitful.

Other improvements involve extending the efficiency improvements to their natural generalization, which includes the pseudoknot algorithm. This could be the most revolutionary change, because the improvements to the standard partition function are less significant since computers are fast enough to compute the largest strands in reasonable time anyways. This is not true for pseudoknotted partition functions, and an efficient algorithm could bring it into the realm of the feasible. This is what

is discussed in the next section.

5.1 Future work: Pseudoknot Algorithm

The partition function algorithm discussed in this thesis and the literature in general has a crucial flaw that makes it incomplete. The algorithm assumes that RNA secondary structure is 'well nested', that there are no overlapping pairs (i, j) and (k, l) such that one of k or l is between i and j and the other is not.

It is not true that RNA secondary structures are well-nested in general. When a non-nested pair is present in a secondary structure, that structure is said to contain a 'pseudoknot'. Large groups of classified RNA sequences such as Group I Introns (around 6% pseudeoknotted) and RNase P (around 13% pseudoknotted) have pseudoknots in their chemically determined secondary structure, and it is in these groups that secondary structure prediction performs the worst (Matthews et all 1999). Large RNA molecules tend to have pseudoknots, which is unfortunate because the partition function computation scales much worse when pseudoknots are included.

In general, computation of pseudoknotted structures is hard. In fact, the problem has been shown to be NP-Complete. Algorithms to perform the computation for a signficiant subset of pseudoknots have been designed, most notably by Dirks and Pierce, who have specified an $O(n^8)$ algorithm and a $O(n^5)$ simplification of that algorithm, using the same approximations used to reduce the complexity of the internal loop energy computation.

I believe the heuristic approach of only iterating over allowed pairs will be able to significantly speed up the pseudoknot computation. This could be truly revolutionary, because there is no software that currrently includes pseudoknots in the computation of the partition function. Since the algorithms seem to perform the worst in high-pseudoknot structure groups, this could lead to a significant improvement in RNA

structure prediction.

However, this would require a good energy model for pseudoknots to be developed. There currently is only one pseudoknot energy model, it is in the Dirks and Peirce paper [5]. However this energy model is not physically based, and is just a linear energy model with a penalty for starting a pseudoknot and a constant bonus for adding pairs. As we have seen with the standard energy model, this approach does not capture enough interaction between terms, so a new energy model needs to be specified. This could be another low hanging fruit.

Appendix A

UNAfold Implementation of Partition Function Improvements

Note the terms Z_{ND}, $Z_{3'D}$, $Z_{5'D}$, and Z_{DD} are extra free energy terms corresponding to 'dangle energies' which are the results of an experiment later implemented in the model to improve it from the standard energy model. In addition there are AU penalty terms appended to where pairs are made, as AU and GU pairs have penalties associated with forming. These additional energy terms improve the model's predictive ability and bring the model closer to the "truth", however it unfortunately makes the partition function seem very threatening.

A.1 New Q(i,j) derivation

In UNAfold, we have that the old recurrence relations were as follows:

$$Q(i,j) = \sum_{k=i}^{j} \left(Q(i,k-1) + e^{-\frac{b(k-i)}{RT}} \right) Q^1(k,j) \tag{A.1}$$

where

$$\begin{aligned} Q^1(i,j) = \; & Q^1(i,j-1)e^{-\frac{b}{RT}} \\ & + e^{-\frac{c}{RT}} Z_{ND}(i,j)Q'(i,j) \\ & + e^{-\frac{b+c}{RT}} Z_{5'D}(i+1,j)Q'(i+1,j) \\ & + e^{-\frac{b+c}{RT}} Z_{3'D}(i,j-1)Q'(i,j-1) \\ & + e^{-\frac{2b+c}{RT}} Z_{DD}(i+1,j-1)Q'(i+1,j-1) \end{aligned} \tag{A.2}$$

We can expand the recursive definition of $Q^1(i,j)$:

$$\begin{aligned} Q^1(i,j) = \sum_{k'=i+1}^{j} e^{-\frac{b(j-k')}{RT}} \Big[\; & e^{-\frac{c}{RT}} Z_{ND}(i,k')Q'(i,k') \\ & + e^{-\frac{b+c}{RT}} Z_{5'D}(i+1,k')Q'(i+1,k') \\ & + e^{-\frac{b+c}{RT}} Z_{3'D}(i,k'-1)Q'(i,k'-1) \\ & + e^{-\frac{2b+c}{RT}} Z_{DD}(i+1,k'-1)Q'(i+1,k'-1) \Big] \end{aligned} \tag{A.3}$$

Plugging this into $Q(i,j)$ we get:

$$\begin{aligned} Q(i,j) = \sum_{k=i}^{j} \sum_{k'=k+1}^{j} \left(Q(i,k-1) + e^{-\frac{b(k-i)}{RT}} \right) e^{-\frac{b(j-k')}{RT}} \Big[\; & e^{-\frac{c}{RT}} Z_{ND}(k,k')Q'(k,k') \\ & + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,k')Q'(k+1,k') \\ & + e^{-\frac{b+c}{RT}} Z_{3'D}(k,k'-1)Q'(k,k'-1) \\ & + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,k'-1)Q'(k+1,k'-1) \Big] \end{aligned} \tag{A.4}$$

Now we'll take the jth element of the second sum and split it out (note that the jth part of the 1st sum has no elements to sum now, so we can decrement that too):

$$
\begin{aligned}
Q(i,j) = \sum_{k=i}^{j-1} \sum_{k'=k+1}^{j-1} \left(Q(i,k-1) + e^{-\frac{b(k-i)}{RT}}\right) e^{-\frac{b(j-k')}{RT}} \Bigg[& \; e^{-\frac{c}{RT}} Z_{ND}(k,k') Q'(k,k') \\
& + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,k') Q'(k+1,k') \\
& + e^{-\frac{b+c}{RT}} Z_{3'D}(k,k'-1) Q'(k,k'-1) \\
& + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,k'-1) Q'(k+1,k'-1) \Bigg] \\
+ \sum_{k=i}^{j} \left(Q(i,k-1) + e^{-\frac{b(k-i)}{RT}}\right) \Bigg[& \; e^{-\frac{c}{RT}} Z_{ND}(k,j) Q'(k,j) \\
& + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,j) Q'(k+1,j) \\
& + e^{-\frac{b+c}{RT}} Z_{3'D}(k,j-1) Q'(k,j-1) \\
& + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,j-1) Q'(k+1,j-1) \Bigg]
\end{aligned}
\tag{A.5}
$$

Notice that the double sum is simply $Q(i,j-1)e^{-b/RT}$ and the terms of the second, single sum are over the pairs with j or $j-1$. Therefore, we can use our heuristic for the pairs of j and $j-1$ to produce the following computation for $Q(i,j)$ which is much more efficient than the previous ones:

$$
\begin{aligned}
Q(i,j) = Q(i,j-1)e^{-b/RT} + \sum_{k(j)} \Bigg[& \left(Q(i,k-1) + e^{-\frac{b(k-i)}{RT}}\right) \; e^{-\frac{c}{RT}} Z_{ND}(k,j) Q'(k,j) \\
& + \left(Q(i,k-2) + e^{-\frac{b(k-i-1)}{RT}}\right) e^{-\frac{b+c}{RT}} Z_{5'D}(k,j) Q'(k,j) \Bigg] \\
+ \sum_{l(j-1)} \Bigg[& \left(Q(i,l-1) + e^{-\frac{b(l-i)}{RT}}\right) \; e^{-\frac{c}{RT}} Z_{ND}(l,j-1) Q'(l,j-1) \\
& + \left(Q(i,l-2) + e^{-\frac{b(l-i-1)}{RT}}\right) e^{-\frac{2b+c}{RT}} Z_{DD}(l,j-1) Q'(l,j-1) \Bigg]
\end{aligned}
\tag{A.6}
$$

A.2 Derivation of new Q'(i, j) formula

For $Q'(i,j)$ we start with the recursion:

$$\begin{aligned} Q'(i,j) = Z_H(i,j) &+ Z_S(i,j)Q'(i+1,j-1) + QBI(i,j) \\ &+ e^{-\frac{a+c}{RT}} Z_{ND}(j,i) \sum_{k=i+3}^{j-5} Q(i+1,k-1)Q^1(k,j-1) \\ &+ e^{-\frac{a+b+c}{RT}} Z_{3'D}(j,i) \sum_{k=i+4}^{j-5} Q(i+2,k-1)Q^1(k,j-1) \\ &+ e^{-\frac{a+b+c}{RT}} Z_{5'D}(j,i) \sum_{k=i+3}^{j-6} Q(i+1,k-1)Q^1(k,j-2) \\ &+ e^{-\frac{a+2b+c}{RT}} Z_{DD}(j,i) \sum_{k=i+4}^{j-6} Q(i+2,k-1)Q^1(k,j-2) \end{aligned} \tag{A.7}$$

The 4 for loops in this make this an expensive computation as the number of bases gets very high. However, these for loops are very similar to the partition function in structure. Indeed, we could perhaps replace each of them with a function of the form $Q^m(i,j)$ defined as

$$Q^m(i,j) = \sum_{k=i+3}^{j-5} Q(i+1,k-1)Q^1(k,j-1) \tag{A.8}$$

Which would simplify the previous sum to a constant time computation, provided we have memoized Q^m:

$$\begin{aligned} Q'(i,j) = Z_H(i,j) &+ Z_S(i,j)Q'(i+1,j-1) + QBI(i,j) \\ &+ e^{-\frac{a+c}{RT}} Z_{ND}(j,i) Q^m(i,j) \\ &+ e^{-\frac{a+b+c}{RT}} Z_{3'D}(j,i) Q^m(i+1,j) \\ &+ e^{-\frac{a+b+c}{RT}} Z_{5'D}(j,i) Q^m(i,j-1) \\ &+ e^{-\frac{a+2b+c}{RT}} Z_{DD}(j,i) Q^m(i+1,j-1) \end{aligned} \tag{A.9}$$

Now there just needs to be a way to efficiently compute Q^m. First we substitute in the expanded version of Q^1:

$$\begin{aligned} Q^m(i,j) = \sum_{k=i+3}^{j-5} \sum_{k'=k+1}^{j-1} Q(i+1,k-1) e^{-\frac{b(j-k')}{RT}} \Big[& e^{-\frac{c}{RT}} Z_{ND}(k,k') Q'(k,k') \\ & + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,k') Q'(k+1,k') \\ & + e^{-\frac{b+c}{RT}} Z_{3'D}(k,k'-1) Q'(k,k'-1) \\ & + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,k'-1) Q'(k+1,k'-1) \Big] \end{aligned} \tag{A.10}$$

Then we do as before and seperate out the jth term of the second sum. Note that there seems to be an additional sum needed to account that I've decreased the first sum's endpoint to $j-6$, but the sum ends up being from $k'=j-4$ to $k'=j-2$ and since Q' for bases less than 4 apart is 0 due to hairpin loop rules, this sum is equal

to zero.

$$
\begin{aligned}
Q^m(i,j) = \sum_{k=i+3}^{j-6} \sum_{k'=k+1}^{j-2} Q(i+1,k-1) e^{-\frac{b(j-k'-1)}{RT}} \Big[& e^{-\frac{c}{RT}} Z_{ND}(k,k') Q'(k,k') \\
& + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,k') Q'(k+1,k') \\
& + e^{-\frac{b+c}{RT}} Z_{3'D}(k,k'-1) Q'(k,k'-1) \\
& + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,k'-1) Q'(k+1,k'-1) \Big] \\
+ \sum_{k=i+3}^{j-5} Q(i+1,k-1) \Big[& e^{-\frac{c}{RT}} Z_{ND}(k,j-1) Q'(k,j-1) \\
& + e^{-\frac{b+c}{RT}} Z_{5'D}(k+1,j-1) Q'(k+1,j-1) \\
& + e^{-\frac{b+c}{RT}} Z_{3'D}(k,j-2) Q'(k,j-2) \\
& + e^{-\frac{2b+c}{RT}} Z_{DD}(k+1,j-2) Q'(k+1,j-2) \Big]
\end{aligned}
\tag{A.11}
$$

The double sum is again going to be equal to $Q^m(i,j-1)e^{-b/RT}$, and the second sum can be made much more efficient by our heuristic.

$$
\begin{aligned}
Q^m(i,j) = Q^m(i,j-1)e^{-b/RT} + \sum_{k(j-1)} \Big[& Q(i+1,k-1) e^{-\frac{c}{RT}} Z_{ND}(k,j-1) Q'(k,j-1) \\
& + Q(i+1,k-2) e^{-\frac{b+c}{RT}} Z_{5'D}(k,j-1) Q'(k,j-1) \Big] \\
+ \sum_{k(j-2)} \Big[& Q(i+1,k-1) e^{-\frac{c}{RT}} Z_{3'D}(k,j-2) Q'(k,j-2) \\
& + Q(i+1,k-2) e^{-\frac{b+c}{RT}} Z_{DD}(k,j-2) Q'(k,j-2) \Big]
\end{aligned}
\tag{A.12}
$$

Since the ks for any individual j are found to be quite limited, the final form should be much more efficient at computing the $Q'(i,j)$.

Note that for Q and in many places for Q', instead of a sum over the known k that

could possibly begin a leftmost pair, we see a double sum. One of them over k that could end a leftmost pair, and this sum is limited to a certain length below j. This is just making the same assumption that the internal loop computation makes: there are not arbitrarily long strands without base pairs, after a certain number of bases it becomes overwhelmingly more likely to make a base pair that we can virtually ignore the energy of the the cases of length beyond a certain L.

As for the seocnd sum, since the number of probable pairs for a base i has been shown empirically to be roughly constant, regardless of length, the second sum is essentially constant. What this all means is that all $O(n^2)$ computations of $Q(i, j)$'s are roughly constant time. This means that the overall algorithm is $O(n^2)$, an improvement over the previous algorithms asymptotic bound by and order of n.

Bibliography

[1] Daniel P Aalberts and William K Jannen. Visualizing rna base-pairing probabilities with rnabow diagrams. *RNA*, 19(4):475–478, 2013.

[2] Phaedra Agius, Kristin P Bennett, and Michael Zuker. Comparing rna secondary structures using a relaxed base-pair score. *RNA*, 16(5):865–878, 2010.

[3] YE Ding, Chi Yu Chan, and Charles E Lawrence. Rna secondary structure prediction by centroids in a boltzmann weighted ensemble. *Rna*, 11(8):1157–1166, 2005.

[4] Ye Ding and Charles E Lawrence. A statistical sampling algorithm for rna secondary structure prediction. *Nucleic acids research*, 31(24):7280–7301, 2003.

[5] Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of computational chemistry*, 24(13):1664–1677, 2003.

[6] Christoph Flamm, Ivo L Hofacker, Peter F Stadler, and Michael T Wolfinger. Barrier trees of degenerate landscapes. *Zeitschrift für Physikalische Chemie International journal of research in physical chemistry and chemical physics*, 216(2/2002):155, 2002.

[7] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[8] Ivo L Hofacker, Peter Schuster, and Peter F Stadler. Combinatorics of rna secondary structures. *Discrete Applied Mathematics*, 88(1):207–237, 1998.

[9] Rune B Lyngsø and Christian NS Pedersen. Rna pseudoknot prediction in energy-based models. *Journal of computational biology*, 7(3-4):409–427, 2000.

[10] Martin Mann and Konstantin Klemm. Efficient exploration of discrete energy landscapes. *Phys. Rev. E*, 83:011113, Jan 2011.

[11] Martin Mann, Marcel Kucharík, Christoph Flamm, and Michael T Wolfinger. Memory efficient rna energy landscape exploration. *Bioinformatics*, page btu337, 2014.

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

[13] 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.

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

[15] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[16] Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded rna. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.

[17] Michael T Wolfinger, W Andreas Svrcek-Seiler, Christoph Flamm, Ivo L Hofacker, and Peter F Stadler. Efficient computation of rna folding dynamics. *Journal of Physics A: Mathematical and General*, 37(17):4731, 2004.

[18] Tianbing Xia, John SantaLucia, Mark E Burkard, Ryszard Kierzek, Susan J Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of rna duplexes with watson-crick base pairs. *Biochemistry*, 37(42):14719–14735, 1998.

[19] Michael Zuker et al. On finding all suboptimal foldings of an rna molecule. *Science*, 244(4900):48–52, 1989.

[20] 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.