Boston College

Doctoral Thesis

On the use of Coarse Grained Thermodynamic Landscapes to Efficiently Estimate Kinetic Pathways for RNA Molecules

---

Author:
Evan Senter

Supervisor:
Dr. Peter Clote

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Clote Lab
Department of Biology

Thursday 23rd July, 2015

Declaration of Authorship

I, Evan Senter, declare that this thesis titled, 'On the use of Coarse Grained Thermodynamic Landscapes to Efficiently Estimate Kinetic Pathways for RNA Molecules' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

BOSTON COLLEGE

Abstract

Dr. Peter Clote
Department of Biology

Doctor of Philosophy

On the use of Coarse Grained Thermodynamic Landscapes to Efficiently Estimate Kinetic Pathways for RNA Molecules

by Evan Senter

Abstract

Acknowledgements

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduced in 1958, the central dogma of biology has been an excellent model for the biological flow of information, much as Newtonian classical mechanics stood the test of time for over 200 years. But just as Einstein's revolutionary principle of relativity have upended our understanding of space in a way unheard of since Copernicus, recent research has gone to confirm that for all our scientific progress, the cell still holds fundamental mysteries, and even the central dogma isn't sacred.

1.1 Thesis Organization

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

Chapter 2

Background

Chapter 3

Ribofinder

3.1 Introduction

In this chapter, we present the `Ribofinder` program—a pipeline to facilitate the detection of putative guanine riboswitches across genomic data. The `Ribofinder` tool operates in three stages. First we use `Infernal` and `TransTermHP` to detect putative aptamers and expression platforms, two distinct components of riboswitches described in section 3.2. After coalescing this data into a pool of candidate riboswitches, we use `RNAfold` with constraints based on experimental data to compute the two distinct structural conformations—'gene on' and 'gene off'. In the third and final stage, we leverage `FoldAlign` to measure the similarity between our candidate pool and a canonical guanine riboswitch well studied in the literature, the xpt G-box riboswitch from Bacillus subtilis.

3.1.1 Organization

This chapter is organized in the following fashion. After providing background on the structural components of a riboswitch alongside their biological significance, we outline the deficiencies in the 'state of the art' software when as it relates specifically to riboswitch detection. We then move on to outline the three stages of `Ribofinder`: candidate selection, structural prediction, and candidate curation. Having described the approach of the software, we move on to present our findings in using `Ribofinder` to detect guanine riboswitches across the bacterial RefSeq database. Finally, we provide

brief commentary on possible extensions of the algorithm to locate other flavors of riboswitches, of which adenine-sensitive aptamers are a straightforward extension.

3.2 Background

Riboswitches are regulatory mRNA elements that modulate gene expression via structural changes induced by the direct sensing of a small-molecule metabolite. Most often found in bacteria, riboswitches regulate diverse pathways including the metabolism and transport of purines, methionine, and thiamin amongst others. The structure of a riboswitch includes an aptamer domain—involved in the direct sensing of the small-molecule—and an expression platform whose structure changes upon the aptamer binding the metabolite. Because of the discriminatory nature of metabolite sensing, groups have had great success in finding representative examples of aptamers across a diverse collection of bacterial species; RFam 12.0 currently contains 26 different families of aptamers involved in different metabolic pathways. Whereas there exists strong sequence and structural similarity within the aptamer of a riboswitch family, the expression platform is highly variable, and thus challenging to capture using traditional SCFG-based approaches. For this reason databases such as RFam only contain the aptamer portion of the riboswitch, and there exists no database providing sequences including expression platforms, necessary for capturing the 'on' and 'off' conformations of this regulatory element. We have developed a new pipeline—called `Ribofinder`—which can detect putative riboswitches including their expression platforms and likely conformational structures across a wide collection of genomic sequences.

3.3 The `Ribofinder` pipeline

At the time of our retrieval (Tuesday 25$^{\text{th}}$ November, 2014 at 09:14), the RefSeq database hosted by NCBI comprised 5,121 complete bacterial genomes with corresponding genomic annotations. In order to both detect putative full riboswitches across this collection of data as well as filter the candidates down to a number tractable for experimental validation, we developed a novel pipeline which takes a three-tiered approach to candidate selection. Our approach is to *a*) identify a pool of candidate riboswitches across genomic data; *b*) perform a coarse-grained filtering of the candidate pool based on structural characteristics; and finally *c*) fine-grained curation of the candidates based on a collection of measures and pairwise similarity.

In the following discussion, we describe the application of `Ribofinder` to identify unannotated G-box purine riboswitches; guanine-sensing cis-regulatory elements which modulate the expression of genes involved in purine biosynthesis.

3.3.1 Step 1: Candidate selection

The RefSeq data we used for analysis contains 5,121 annotated bacterial genomes across 2,732 different organisms, totaling over $9.5 * 10^9$ bases. We used the program `Infernal` to determine the coordinates of putative aptamer structures within the RefSeq genomes, and `TransTermHP` to locate candidate rho-independent transcription terminators.

3.3.1.1 Detecting Aptamers with `Infernal`

`Infernal` uses a stochastic context-free grammar (SCFG) with a user-provided multiple sequence alignment (MSA) to efficiently scan genomic data for RNA homologs, taking into consideration both sequence and structural conservation. Using the purine aptamer MSA from RFam 12.0 (RF00167), `Infernal` (v1.1.1, default options) detects 1,537 significant hits having E-value <= 0.01. Because `Infernal` leverages the concept of a 'local end'—a large insertion or deletion in the alignment at reduced cost—it is possible for the software to return a significant hit whose aligned structure does not have the canonical three-way junction observed in all purine riboswitches. `Ribofinder` prunes these truncated `Infernal` hits by converting the alignment structure into a parse tree, and only permitting trees of sufficient complexity to contain a multiloop (described further in 3.3.2.1). The pyrimidine residue abutted next to the P1 stem in the J3-1 junction differentiates between guanine and adenine-sensing riboswitches by binding the complimentary purine ligand; for our interest in G-box riboswitches exclusively we require the presence of a cytosine at this residue. In total, using `Infernal` with these additional filters yields 1,280 G-box aptamers across 555 unique organisms (note: here and elsewhere I define a 'unique organism' as having a unique taxonomy ID).

3.3.1.2 Detecting Expression Platforms with `TransTermHP`

`TransTermHP` detects rho-independent terminators in bacterial genomes in a context-sensitive fashion by leveraging the protein annotations available in PTT data. These terminator sequences canonically have a stable hairpin loop structure immediately preceding a run of 5+ uracil residues, the combination of which causes the ribosomal machinery to stall and dissociate from the transcript. `TransTermHP` performs a genomic

scan to determine candidate loci with this motif, and returns scored hits. The scoring system considers both structural homology and the genomic contextual information available in the PTT file. Across our collection of bacterial genomes acquired from NCBI RefSeq data, `TransTermHP` identified 2,752,469 rho-independent terminators using the default filters.

Due to the spatially-mediated structural regulation of purine riboswitches, whereby ligand interaction with the aptamer domain induces local structural rearrangement in the expression platform, we paired aptamers with corresponding terminators by minimizing the genomic distance, with an upper bound of 200 nucleotides between the end of the aptamer domain and start of the terminator. This approach yields 577 candidate riboswitches, 81 of which have multiple rho-independent terminators within range of a putative aptamer produced by `Infernal`. For these, we simply pair the closest `TransTermHP` hit with the aptamer domain.

3.3.2 Step 2: Structural prediction

3.3.2.1 Notation for Representing Abstract RNA Shapes

Given an RNA sequence $\mathbf{s} = a_1, a_2, \cdots, a_n$, where positions a_i are drawn from the collection of single-letter nucleotide codes, i.e. $\{\mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C}\}$, it is possible to describe a corresponding secondary structure $\mathcal{S}$ compatible with $\mathbf{s}$ using the dot-bracket notation. In this notation, each nucleotide a_i has a corresponding state s_i, where s_i is denoted as a '.' if unpaired and a '(' [resp. ')'] if the left [resp. right] base in a basepair. Given any two basepairs (i, j) and (k, l) in $\mathcal{S}$, then $i < k < j \iff i < l < j$; pseudoknots are not permitted in the structure. A secondary structure taking this form is said to have balanced parentheses, and can additionally be represented using a context-free grammar such as:

$$S \to S\,. \mid .\,S \mid (S) \mid SS \mid \epsilon \tag{3.1}$$

The grammar from (3.1) can be used to generate a parse tree $\mathcal{T}$ for $\mathcal{S}$. The benefit of working with $\mathcal{T}$ over $\mathcal{S}$ is that the parse tree offers an abstract representation of secondary structure shape independent of sequence length, permitting us to classify and eventually constrain a large collection of sequences having variable length which are all expected to have the same abstract tree shape. This is analogous to what the Giegerich

lab refers to as their 'type 5' structural abstraction using the `RNAshapes` tool. Every node in $\mathcal{T}$ represents a helix in $\mathcal{S}$, and internally tracks the indices of both its beginning (i, j) and closing (k, l) basepair. We use a level-order naming convention to refer to helices within the parse tree, whereby a position $\mathtt{p}_1$ references the first child of the root node, $\mathtt{p}_{1,2}$ references the second child of $\mathtt{p}_1$, and generally $\mathtt{p}_{i_1, i_2, \cdots, i_n}$ refers to the ${i_n}^{\text{th}}$ child of $\mathtt{p}_{i_1, i_2, \cdots, i_{n-1}}$. To reference specific nucleotides in the context of their location relative to a helix, we use the opening and closing basepairs (i, j) and (k, l) as landmarks. Thus, $\mathtt{p}_1(l)$ is the index in $\mathcal{S}$ of the right-hand side closing basepair of $\mathtt{p}_1$. We use the notation $\mathtt{t}_i$ to refer to the subtree of $\mathcal{T}$ whose root is $\mathtt{p}_i$.

Finally, we introduce the concept of a tree signature. The tree signature for a tree $\mathcal{T}$ is a list of the node depths when traversed in a depth-first pre-order fashion. To provide a concrete example, consider the following experimentally validated xpt G-box riboswitch from Bacillus subtilis subsp. subtilis str. 168 (NC_000964.3 2320197-2320054) with corresponding gene-off structure:

```
ACACUCAUAUAAUCGCGUGGAUAUGGCACGCAAGUUUCUACCGGGCACCGUAAAUGUCCGACUAUGGGUGAGCAAUGGAACCGCACGUGUACGGUUU
.(((((((((.....(((((.......)))))..........((((((.......))))))..))))))))).((((.(((((((.......))))))
```

The `RNAshapes` 'type 5' representation for this structure is `[[][]][][]` (note the coalesced left bulge in the hairpin immediately downstream the closing multiloop stem, at helix $\mathtt{p}_2$) and the tree signature for this parse tree of the structure is `[0, 1, 2, 2, 1, 1]`.

We leverage the notion of abstract structural filtering initially to ensure that all `Infernal` aptamer hits have a tree signature of `[0, 1, 2, 2]`, which represents a three-way junction, and that the binding site for the guanine ligand $\mathtt{p}_1(l-1) = \mathrm{C}$. These filters, in combination with the proximal terminator hairpins produced by `TransTermHP` yield the aforementioned 577 candidate guanine riboswitches for which we then try to produce reasonable gene-on and off structures.

3.3.2.2 Constrained Folding to Predict Switch Structures

To restrict our search to unannotated G-box riboswitches, and further ensure that we are not re-detecting sequences based off the RFam covariance model provided to `Infernal`, we constrain our search to those RefSeq organisms not represented in the RFam seed alignment. 503 of the 577 candidates, or 87.18% represent putative unannotated riboswitches not represented by RF00167.

The gene-off structure $\mathcal{S}_{\text{off}}$ for a G-box riboswitch is the easier of the two to find computationally, since the terminator loop is exceptionally thermodynamically stable. In the gene-on conformation $\mathcal{S}_{\text{on}}$, the P1 stem of the multiloop partially dissociates and an anti-terminator loop forms between the region immediately 3' of the P1 stem and what was the left-hand side of the terminator loop. This truncated P1 stem, which closes the three-way junction in the aptamer, is exceptionally unstable based on present energy models available for structural folding, and requires special treatment to reconstitute in our final structures.

The software `RNAfold` (v2.1.8) allows for the folding of RNA molecules with 'loose' constraints. In this model of constrained folding, the resulting structure produced by the software guarantees not to explicitly invalidate any user-provided constraints, but does not guarantee all constraints will be satisfied in the resulting structure. For each of the candidate guanine riboswitches, having $\mathcal{T}_{\texttt{Infernal}}$ and $\mathcal{T}_{\texttt{TransTermHP}}$, we build the following constraint masks:

G-box gene-off constraint mask	G-box gene-on constraint mask
Prohibit basepairing upstream of $\mathtt{p}_1(i)$ and downstream of $\mathtt{p}_2(l)$. Force basepairs and unpaired regions in $\mathtt{t}_1$, with the exception of $\mathtt{p}_1$. Prohibit formation of $\mathtt{p}_1$ stem, which closes the three-way junction.	
Force basepairs and unpaired regions in $\mathtt{t}_2$.	Require m nucleotides starting from $\mathtt{p}_1(l+3)$ to pair to the right, where $m = len(\mathtt{p}_2)$, and require the left-hand side of the $\mathtt{p}_2$ helix to pair to the left. Disallow pairing downstream of $\mathtt{p}_2(j)$.

These constraint masks are run using the command-line flags `-d 0 -P rna_turner1999.par` to disable dangles and use the Turner 1999 energies respectively. Experimental evidence using inline probing suggests that the 'on' conformation of the G-box riboswitch has a reduced P1 stem length of 3 base pairs; in practice we were unable to force `RNAfold` to respect this constraint regardless of command-line options specified. For this reason we reconstitute the P1 stem in both structures after constrained folding, having length equivalent to it the `Infernal` P1 stem (resp. 3 basepairs) in the gene-off (resp. gene-on) structure.

This difficulty with `RNAfold` can be shown by using the constraint-produced structures as exhaustive constraints themselves. All unpaired nucleotides in $\mathcal{S}_{\text{off}}$ and $\mathcal{S}_{\text{on}}$ are notated

by a '`x`' and all base pairs by '`()`' for the 5' and 3' side of the pair respectively to form new constraints mask $\mathcal{C}_{\text{off}}$ and $\mathcal{C}_{\text{on}}$, having all bases' state explicitly specified. By refolding all 577 candidate sequences with $\mathcal{C}_{\text{off}}$ and $\mathcal{C}_{\text{on}}$ using the same options as before, only 463 (or 80.24%) of the resulting structures from $\mathcal{C}_{\text{off}}$ have the tree signature prefix `[0, 1, 2, 2, 1]`, and just 21 (or 3.64%) of the $\mathcal{C}_{\text{on}}$ structures correctly re-fold their multiloop.

3.3.3 Step 3: Candidate curation

Until now, we have described our approach for generating the 503 guanine riboswitch candidates in RefSeq, alongside their gene-on and off structures. Unfortunately the experimental validation of all 503 candidates is not tractable, so it was necessary to reduce this collection again to a more manageable size, while only keeping the most promising candidates.

3.4 Using **Ribofinder** against the RefSeq database

3.5 Extending beyond guanine riboswitches

Chapter 4

FFTbor

4.1 Introduction

In this chapter, we present the `FFTbor` algorithm and accompanying software. `FFTbor` is a novel algorithm developed with the intent of efficiently computing the Boltzmann probability of those structures whom, for a given input RNA sequence $\mathbf{s}$, differ by k base pairs. By leveraging polynomial interpolation via the Fast Fourier Transform, this algorithm runs in $O(n^4)$ time and $O(n^2)$ space, a significant improvement over its predecessor. The accompanying software which implements this algorithm has been used to predict the location of expression platforms for putative riboswitches in genomic data, and to evaluate the correlation between kinetic folding speed and landscape ruggedness.

4.1.1 Organization

This chapter is organized in the following fashion. First, we provide background on the problem which `FFTbor` aims to address, as well as a brief overview of existing approaches. We follow by a formal explanation of the problem, and proceed to describe how the energy landscape is coarsified into discrete bins. We then develop the recursions for the parameterized partition function using the Nussinov-Jacobson energy model, which allows us to exposé the novel aspects of the algorithm. After developing the recursions, we indicate how they can be reformulated as a polynomial whose coefficients $c_k = \mathbf{Z}^k_{1,n}$. We then describe how the Fast Fourier Transform can be employed to efficiently compute the coefficients c_k, finishing our description of the underlying algorithm. Then we proceed to present two applications of `FFTbor`, an application to RNA kinetics and another

to riboswitch detection in genomic data. Finally, we give reference to the full recursions using the more accurate Turner energy model, which the underlying implementation actually uses.

4.2 Formalization of the problem

`FFTbor` aims to compute the coefficients $p_0, \ldots, p_{n-1}$ of the polynomial

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}, \tag{4.1}$$

where p_k is defined as $p_k = \frac{Z_k}{Z}$. We employ the Fast Fourier Transform to compute the inverse Discrete Fourier Transform on values $y_0, \ldots, y_{n-1}$, where $y_k = p(\omega^k)$ and $\omega = e^{2\pi i/n}$ is the principal complex nth root of unity and $p(x)$ is defined in equation (4.1). By leveraging complex nth roots of unity in conjunction with the inverse Discrete Fourier Transform the we subvert numeric instability issues observed with both Lagrange interpolation and Gaussian elimination.

Consider an RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, where $s_i \in \{\mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C}\}$, i.e. a sequence of nucleotides. We can describe a secondary structure $\mathcal{S}$ which is compatible with $\mathbf{s}$ as a collection of base pair tuples (i, j), where $1 \leq i \leq i + \theta < j \leq n$ and $\theta \geq 0$ (generally taken to be 3), the minimum number of unpaired bases in a hairpin loop due to steric constraints.

To more simply develop the underlying recursions for `FFTbor`, we introduce a number of constraints on the base pairs within $\mathcal{S}$. Firstly, we require that each base pair is either a Watson-Crick or G-U wobble, i.e. base pair (i, j) for sequence $\mathbf{s}$ has corresponding nucleotides (s_i, s_j), which are restricted to the set

$$\mathbb{B} = \{(\mathrm{A}, \mathrm{U}), (\mathrm{U}, \mathrm{A}), (\mathrm{G}, \mathrm{C}), (\mathrm{C}, \mathrm{G}), (\mathrm{G}, \mathrm{U}), (\mathrm{U}, \mathrm{G})\}. \tag{4.2}$$

With this constraint satisfied we say that $\mathcal{S}$ is *compatible* with $\mathbf{s}$, and for the remainder of this chapter will only consider those structures which are compatible with $\mathbf{s}$. Secondly, we insist that given two base pairs $(i, j), (x, y)$ from $\mathcal{S}$, $i = x \iff j = y$ (bases have at most one partner). Finally, we require that $i < x < j \iff i < y < j$ (no pseudoknots are allowed). While pseudoknots have been shown to be present in some biologically

relevant RNAs, their inclusion greatly complicates the recursive decomposition of the structure, and thus it is common to ignore them.

Provided two secondary structures $\mathcal{S}, \mathcal{T}$, we can define a notion of distance between them. There are a number of different definitions of distance used across the literature; we will use *base pair distance* for `FFTbor`. Base pair distance is defined as the symmetric difference between the sets $\mathcal{S}, \mathcal{T}$:

$$d_{\mathrm{BP}}(\mathcal{S}, \mathcal{T}) = |\mathcal{S} \cup \mathcal{T}| - |\mathcal{S} \cap \mathcal{T}|. \tag{4.3}$$

Given this definition of distance, two structures $\mathcal{S}$ and $\mathcal{T}$ are said to be *k-neighbors* if $d_{\mathrm{BP}}(\mathcal{S}, \mathcal{T}) = k$. It is important to note that the notion of base pair distance is also applicable to restrictions of secondary structures on the subsequence $\mathbf{s}_{i,j}$, i.e. $\mathcal{S}_{[i,j]} = \{(x, y) : i \leq x < y \leq j, (x, y) \in \mathcal{S}\}$.

For a restriction of base pairs for a given structure $\mathcal{S}_{[i,j]}$, $\mathcal{T}_{[i,j]}$ is said to be a *k-neighbor* of $\mathcal{S}_{[i,j]}$ if

$$d_{\mathrm{BP}}(\mathcal{S}_{[i,j]}, \mathcal{T}_{[i,j]}) = |\{(x, y) : i \leq x < y \leq j, (x, y) \in \mathcal{S} - \mathcal{T} \mathrm{or} (x, y) \in \mathcal{T} - \mathcal{S}\}| = k. \tag{4.4}$$

4.3 Derivation of the `FFTbor` algorithm

Given an RNA sequence $\mathbf{s} = s_1, \ldots, s_n$ and compatible secondary structure $\mathcal{S}^*$, let $\mathbf{Z}^k$ denote the sum of the Boltzmann factors $\exp(-E(\mathcal{S})/RT)$ of all *k-neighbors* $\mathcal{S}$ of $\mathcal{S}^*$; i.e.

$$\mathbf{Z}^k = \mathbf{Z}^k_{1,n} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) = k}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{4.5}$$

where $E(\mathcal{S})$ denotes the Turner (nearest neighbor) energy of $\mathcal{S}$, $R = 0.00198$ kcal/mol denotes the universal gas constant and T denotes absolute temperature. From this, it follows that the full partition function is defined as

$$\mathbf{Z} = \mathbf{Z}_{1,n} = \sum_{k=0}^{n} \mathbf{Z}_{1,n}^{k} \tag{4.6}$$

since the base pair distance between $\mathcal{S}^*$ and $\mathcal{S}$ is at most

$$d_{\mathrm{BP}}(\mathcal{S}^*, \mathcal{S}) \leq |\mathcal{S}^*| + \lfloor \frac{n-\theta}{2} \rfloor \leq n. \tag{4.7}$$

We can then define the Boltzmann probability of all *k-neighbors* of $\mathcal{S}^*$ as

$$p(k) = \frac{\mathbf{Z}_{1,n}^{k}}{\mathbf{Z}_{1,n}}. \tag{4.8}$$

By visualizing the probabilities p_k as a function of k, we generate a coarse grained view of the one-dimensional energy landscape of $\mathbf{s}$ with respect to $\mathcal{S}^*$. When $\mathcal{S}^*$ is taken to be the minimum free energy structure for example, one would anticipate to see a peak at $k = 0$, with additional peaks implying additional metastable structures; local energy minima which could suggest an energetic trap while folding.

4.3.1 Definition of the partition function $\mathbf{Z}_{1,n}^{k}$

For the rest of the paper, we consider both $\mathbf{s}$ as well as the secondary structure $\mathcal{S}^*$ on $\mathbf{s}$ to be fixed. We now recall the recursions from Freyhult et al. **?** to determine the partition function $\mathbf{Z}_{i,j}^{k}$ with respect to the Nussinov-Jacobson energy E_0 model **?**, defined by -1 times the number of base pairs; i.e. $E_0(S) = -1 \cdot |S|$. Although we describe here the recursions for the Nussinov-Jacobson model, for the sake of simplicity of exposition, both `RNAbor` **?** as well as our current software `FFTbor`, concern the Turner energy model, consisting of free energy parameters for stacked bases, hairpins, bulges, internal loops and multiloops.

The base case for $\mathbf{Z}_{i,j}^{k}$ is given by

$$\mathbf{Z}_{i,j}^{0} = 1, \text{ for } i \leq j, \tag{4.9}$$

since the only 0-neighbor to a structure $\mathcal{S}^*$ is the structure $\mathcal{S}^*$ itself, and

$$\mathbf{Z}_{i,j}^{k} = 0, \text{ for } k > 0, i \leq j \leq i + \theta, \tag{4.10}$$

since the empty structure is the only possible structure for a sequence shorter than $\theta + 2$ nucleotides, and so there are no *k-neighbors* for $k > 0$. The recursion used to compute $\mathbf{Z}_{i,j}^{k}$ for $k > 0$ and $j > i + \theta$ is

$$\mathbf{Z}_{i,j}^{k} = \mathbf{Z}_{i,j-1}^{k-b_0} + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\leq r<j}} \sum_{w+w'=k-b(r)} \exp(-E_0(r,j)/RT) \cdot \mathbf{Z}_{i,r-1}^{w} \mathbf{Z}_{r+1,j-1}^{w'}, \tag{4.11}$$

where $E_0(r, j) = -1$ if positions r, j can pair in sequence $\mathbf{s}$, and otherwise $E_0(r, j) = +\infty$. Additionally, $b_0 = 1$ if j is base-paired in $\mathcal{S}^*_{[i,j]}$ and 0 otherwise, and $b(r) = d_{\mathrm{BP}}(\mathcal{S}^*_{[i,j]}, \mathcal{S}^*_{[i,r-1]} \cup \mathcal{S}^*_{[r+1,j-1]} \cup \{(r, j)\})$. This holds since in a secondary structure $T_{[i,j]}$ on $s_i, \ldots, s_j$ that is a *k-neighbor* of $\mathcal{S}^*_{[i,j]}$, either nucleotide j is unpaired in $[i, j]$ or it is paired to a nucleotide r such that $i \leq r < j$. In this latter case it is enough to study the smaller sequence segments $[i, r - 1]$ and $[r + 1, j - 1]$ noting that, except for (r, j), base pairs outside of these regions are not allowed, since there are no pseudoknots. In addition, for $d_{\mathrm{BP}}(\mathcal{S}^*_{[i,j]}, T_{[i,j]}) = k$ to hold, it is necessary for $w + w' = k - b(r)$ to hold, where $w = d_{\mathrm{BP}}(\mathcal{S}^*_{[i,r-1]}, T_{[i,r-1]})$ and $w' = d_{\mathrm{BP}}(\mathcal{S}^*_{[r+1,j-1]}, T_{[r+1,j-1]})$, since $b(r)$ is the number of base pairs that differ between $\mathcal{S}^*_{[i,j]}$ and a structure $T_{[i,j]}$, due to the introduction of the base pair (r, j).

Given RNA sequence $\mathbf{s}$ and compatible initial structure $\mathcal{S}^*$, we define the *polynomial*

$$\mathcal{Z}(x) = \sum_{k=0}^{n} c_k x^k \tag{4.12}$$

where coefficients $c_k = \mathbf{Z}_{1,n}^{k}$. Moreover, because of equation (4.7) and the fact that the minimum number of unpaired bases in a hairpin loop θ is 3, we know that $c_n = 0$, so that $\mathcal{Z}(x)$ is a polynomial of degree strictly less than n. If we evaluate the polynomial $\mathcal{Z}(x)$ for n distinct values

$$\mathcal{Z}(a_1) = y_1, \ldots, \mathcal{Z}(a_n) = y_n, \tag{4.13}$$

then the Lagrange polynomial interpolation formula guarantees that $\mathcal{Z}(x) = \sum_{k=1}^{n} y_k P_k(x)$, where the polynomials $P_k(x)$ have degree at most $n-1$ and are given by the Lagrange formula

$$P_k(x) = \frac{\prod_{i \neq k}(x - x_i)}{\prod_{i \neq k}(x_k - x_i)}. \tag{4.14}$$

Since the polynomials $P_k(x)$ can be explicitly computed, it follows that we can compute the coefficients c_k of polynomial $\mathcal{Z}(x)$. As we describe below, the evaluation of $\mathcal{Z}(x)$ for a fixed value of x can be done in time $O(n^3)$ and space $O(n^2)$. It follows that the coefficients $c_k = \mathbf{Z}_{1,n}^k$ can be computed after n evaluations of $\mathcal{Z}(x)$, where the space for each evaluation of $\mathcal{Z}(x)$ is re-used; hence these evaluations can be performed in time $O(n^4)$ and space $O(n^2)$. Finally, Lagrange interpolation is clearly computable in time $O(n^3)$. Although this approach is theoretically sound, there are severe numerical stability issues related to the interpolation method **?**, the choice of values $a_1, \ldots, a_n$ in the interpolation, and floating point arithmetic (round-off error) related to the astronomically large values of the partition functions $\mathbf{Z}_{1,n}^k$, for $0 \leq k < n$. After many unsuccessful approaches including scaling we obtained excellent results by interpolating the polynomial $p(x)$, defined in equation (4.1), rather than the polynomial $\mathcal{Z}(x)$, defined in equation (4.12), and performing interpolation with the Fast Fourier Transform (FFT) **?** where $\alpha_0, \ldots, \alpha_{n-1}$ are chosen to be complex nth roots of unity, $\alpha_k = e^{2\pi i k/n}$. One advantage of the FFT is that interpolation can be performed in $O(n \log n)$ time, rather than the cubic time required by using the Lagrange formula (4.14) or by Gaussian elimination. Fewer numerical operations implies increased numerical stability in our application.

4.3.2 Recursions to compute the polynomial $\mathcal{Z}_{i,j}(x)$

Given an initial secondary structure $\mathcal{S}^*$ of a given RNA sequence $\mathbf{s}$, our goal is to compute

$$\mathbf{Z}_{1,n}^k = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) = k}} e^{\frac{-E_0(\mathcal{S})}{RT}} \tag{4.15}$$

where $\mathcal{S}$ can be any structure compatible with $\mathbf{s}$. As previously mentioned, the recurrence relation for `RNAbor` with respect to the Nussinov energy model E_0 is

$$\mathbf{Z}_{i,j}^{k} = \mathbf{Z}_{i,j-1}^{k-b_0} \; + \sum_{\substack{s_r s_j \in \mathbb{B}, \\ i \le r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \sum_{w+w'=k-b(r)} \mathbf{Z}_{i,r-1}^{w} \mathbf{Z}_{r+1,j-1}^{w'} \right) \tag{4.16}$$

where $E_0(r,j) = -1$ if r and j can base-pair and otherwise $+\infty$, and $b_0 = 1$ if j is base paired in $\mathcal{S}^*_{[i,j]}$ and 0 otherwise, and $b(r) = d_{\mathrm{BP}}(\mathcal{S}^*_{[i,j]}, \mathcal{S}^*_{[i,r-1]} \cup \mathcal{S}^*_{[r+1,j-1]} \cup \{(r,j)\})$. The following theorem shows that an analogous recursion can be used to compute the *polynomial* $\mathcal{Z}_{i,j}(x)$ defined by

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k(i,j)\, x^k \tag{4.17}$$

where

$$c_k(i,j) \; = \; \mathbf{Z}_{i,j}^{k} \; = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*_{[i,j]}) = k}} e^{\frac{-E_0(\mathcal{S})}{RT}}. \tag{4.18}$$

Here, in the summation, $\mathcal{S}$ runs over structures on $s_i, \ldots, s_j$, which are *k-neighbors* of the restriction $\mathcal{S}^*_{[i,j]}$ of initial structure $\mathcal{S}^*$ to interval $[i,j]$, and $E_0(S) = -1 \cdot |S|$ denotes the Nussinov-Jacobson energy of $\mathcal{S}$.

Theorem 4.1. *Let $s_1, \ldots, s_n$ be a given RNA sequence. For any integers $1 \le i \le j \le n$, let*

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k\, x^k \tag{4.19}$$

where

$$c_k(i,j) = \mathbf{Z}_{i,j}^{k}. \tag{4.20}$$

Then for $i \le j \le i + \theta$, $\mathcal{Z}_{i,j}(x) = 1$ and for $j > i + \theta$ we have the recurrence relation

$$\mathcal{Z}_{i,j}(x) = \mathcal{Z}_{i,j-1}(x) \cdot x^{b_0} + \sum_{\substack{s_r s_j \in \mathbb{B}, \\ i \le r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(x) \cdot \mathcal{Z}_{r+1,j-1}(x) \cdot x^{b(r)} \right). \quad (4.21)$$

where $b_0 = 1$ *if* j *is base-paired in* $\mathcal{S}^*_{[i,j]}$ *and* 0 *otherwise, and* $b(r) = d_{BP}(\mathcal{S}^*_{[i,j]}, \mathcal{S}^*_{[i,r-1]} \cup \mathcal{S}^*_{[r+1,j-1]} \cup \{(r,j)\})$.

Proof. First, some notation is necessary. Recall that if F is an arbitrary polynomial [resp. analytic] function, then $[x^k]F(x)$ denotes the coefficient of x^k [resp. the kth Taylor coefficient in the Taylor expansion of F]. For instance, in equation (4.1), $[x^k]p(x) = p_k$, and in equation (4.12), $[x^k]\mathcal{Z}(x) = c_k(i,j)$.

By definition, it is clear that $\mathcal{Z}_{i,j}(x) = 1$ if $i \le j \le i + \theta$, where we recall that $\theta = 3$ is the minimum number of unpaired bases in a hairpin loop. For $j > i + \theta$, we have

$$\begin{aligned}
[x^k]\mathcal{Z}_{i,j}(x) = c_k(i,j) &= \mathbf{Z}^k_{i,j} \\
&= \mathbf{Z}^{k-b_0}_{i,j-1} + \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \mathbf{Z}^{k_0}_{i,r-1} \cdot \mathbf{Z}^{k_1}_{r+1,j-1} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \left\{ [x^{k_0}]\mathcal{Z}_{i,r-1}(x) \right\} \cdot \left\{ [x^{k_1}]\mathcal{Z}_{r-1,j-1}(x) \right\} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot [x^{k_0+k_1}]\mathcal{Z}_{i,r-1}(x)\mathcal{Z}_{r-1,j-1}(x).
\end{aligned} \quad (4.22)$$

By induction, the proof of the theorem now follows. □

Notice that if one were to compute all terms of the polynomial $\mathcal{Z}_{1,n}(x)$ by explicitly performing polynomial multiplications, then the computation would require $O(n^5)$ time and $O(n^3)$ space. Instead of explicitly performing polynomial expansion in *variable* x, we instantiate x to a fixed complex number $\alpha \in \mathbb{C}$, and apply the following recursion for this instantiation:

$$\mathcal{Z}_{i,j}(\alpha) = \mathcal{Z}_{i,j-1}(\alpha) \cdot \alpha^{b_0} + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\le r<j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(\alpha) \cdot \mathcal{Z}_{r+1,j-1}(\alpha) \cdot \alpha^{b(r)} \right). \quad (4.23)$$

In this fashion, we can compute $\mathcal{Z}(\alpha) = \mathcal{Z}_{1,n}(\alpha)$ in $O(n^3)$ time and $O(n^2)$ space. For n distinct complex values $\alpha_0, \ldots, \alpha_{n-1}$, we can compute and save only the values $\mathcal{Z}(\alpha_0), \ldots, \mathcal{Z}(\alpha_{n-1})$, each time re-using the $O(n^2)$ space for the next computation of $\mathcal{Z}(\alpha_k)$. It follows that the computation resources used to determine the (column) vector

$$\mathbf{Y} = (y_0, \ldots, y_{n-1})^{\mathrm{T}} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} \quad (4.24)$$

where $y_0 = \mathcal{Z}(\alpha_0), \ldots, y_{n-1} = \mathcal{Z}(\alpha_{n-1}))$ is thus quartic time $O(n^4)$ and quadratic space $O(n^2)$.

4.3.3 Polynomial interpolation to evaluate $\mathcal{Z}_{i,j}(x)$

Let $\omega = e^{2\pi i/n}$ be the principal complex nth root of unity. Recall that the Vandermonde matrix V_n is defined to be the $n \times n$ matrix, whose i, j entry is $\omega^{i \cdot j}$; i.e.

$$V_n = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \ldots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{pmatrix} \quad (4.25)$$

The Fast Fourier Transform is defined to be the $O(n \log n)$ algorithm to compute the Discrete Fourier Transform (DFT), defined as the matrix product $\mathbf{Y} = V_n \mathbf{A}$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = V_n \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} \tag{4.26}$$

On page 837 of **?**, it is shown that the (i, j) entry of V_n^{-1} is $\frac{\omega^{-ji}}{n}$ and that

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \, \omega^{-kj} \tag{4.27}$$

for $j = 0, \ldots, n-1$.

Since we defined $\mathbf{Y}$ in equation (4.24) by $\mathbf{Y} = (y_0, \ldots, y_{n-1})^{\mathrm{T}}$, where $y_0 = \mathcal{Z}(\alpha_0), \ldots, y_{n-1} = \mathcal{Z}(\alpha_{n-1}))$ and $\alpha_k = \omega^k e^{2\pi i k/n}$, it follows that the coefficients $c_k = \mathbf{Z}_{1,n}^k$ in the polynomial $\mathcal{Z}(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$ defined in equation (4.12) can be computed, at least in principle, by using the Fast Fourier Transform. It turns out, however, that the values of $\mathbf{Z}_{1,n}^k$ are so astronomically large, that the ensuing numerical instability makes even this approach infeasible for values of n that exceed 56 (data not shown). Nevertheless, our approach can be modified as follows. Define $\mathbf{Y}$ by $\mathbf{Y} = (y_1, \ldots, y_n)^{\mathrm{T}}$, where $y_1 = \frac{\mathcal{Z}(\alpha_1)}{\mathcal{Z}(x)}, \ldots, y_n = \frac{\mathcal{Z}(\alpha_n)}{\mathcal{Z}(x)}$, and $\mathbf{Z}$ is the partition function defined in equation (4.6). Using the Fast Fourier Transform to compute the inverse Discrete Fourier Transform, it follows from equation (4.27) that we can compute the probabilities $p_0, \ldots, p_{n-1}$ that are coefficients of the polynomial $p(x) = p_0 + p_1 x + \cdots + p_{n-1} x^{n-1}$ defined in equation (4.1). For genomics applications, we are only interested in the m most significant digits of each p_k, as described in the pseudocode on the following page.

4.4 Benchmarking and performance considerations

In this subsection, we show that we need only evaluate the polynomial $\mathcal{Z}(x)$, as defined in equation (4.12), for $n/2$ of the complex nth roots of unity. It is first necessary to recall the definition of complex conjugate. Recall that the complex conjugate of z is denoted by $\overline{z}$; i.e. if $z = a + bi$ where $a, b \in \mathbb{R}$ are real numbers and $i = \sqrt{-1}$, then $\overline{z} = a - bi$.

Pseudocode for `FFTbor`

PURPOSE:	Computes the m most significant digits of probabilities $p_k = \mathbf{z}^k_{1,n}/\mathbf{z}$
INPUT:	RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structure $\mathcal{S}^*$ of $\mathbf{s}$, integer m
OUTPUT:	Probabilities $p_k = \mathbf{z}^k_{1,n}/\mathbf{z}$ to m significant digits for $k = 0, \ldots, n-1$

---

1 **function** FFTBOR($\mathbf{s}$, $\mathcal{S}^*$, m)
2 $n \leftarrow$ *length*($\mathbf{s}$)
3 **for** $k \leftarrow 0, n-1$ **do** ▷ Compute all complex nth roots of unity
4 $\omega_k \leftarrow \exp(\frac{2\pi i k}{n})$
5 **end for**
6 **for** $k \leftarrow 0, n-1$ **do** ▷ Note that $\mathcal{Z}(\omega_0) = \mathbf{Z}$
7 $y_k \leftarrow 10^m \cdot \frac{\mathcal{Z}(\omega_k)}{\mathcal{Z}(\omega_0)}$
8 **end for**
9 **for** $k \leftarrow 0, n-1$ **do** ▷ Compute IDFT from equation (4.27)
10 $a_k \leftarrow \frac{1}{n} \sum_{j=0}^{n-1} y_j \, \omega^{-kj}$
11 $p_k \leftarrow 10^{-m} \cdot \lfloor a_k \rfloor$ ▷ Truncate to m significant digits
12 **end for**
13 **return** $p_0, \ldots, p_{n-1}$ ▷ Return all p_k for $0 \leq k < n$, from equation (4.8)
14 **end function**

FIGURE 4.1: Caption here

Lemma 4.2. *If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (4.12), then for any complex nth root of unity α, it is the case that $\mathcal{Z}(\overline{\alpha}) = \overline{\mathcal{Z}(\alpha)}$. In other words, if α is a complex nth root of unity of the form $a + bi$, where $a, b \in \mathbb{R}$ and $b > 0$, and if $\mathcal{Z}(a + bi) = A + Bi$ where $A, B \in \mathbb{R}$, then it is the case that*

$$\mathcal{Z}(a - bi) = A - Bi. \tag{4.28}$$

Proof. Letting $i = \sqrt{-1}$, if $\theta = \frac{2\pi}{n}$, then $\omega = e^{i\theta} = \cos(\theta) + i\sin(\theta)$ is the principal complex nth root of unity, and $e^{(0)\cdot i\theta} = 1 = \omega^0, \ldots, e^{(n-1)\cdot i\theta} = \omega^{n-1}$ together constitute the complete collection of all complex nth roots of unity—i.e. the n unique solutions of of the equation $x^n - 1 = 0$ over the field $\mathbb{C}$ of complex numbers. Clearly, for any $1 \leq k < n$, $e^{-ik\theta} = 1 \cdot e^{-ik\theta} = e^{2\pi i} \cdot e^{-ik\theta} = e^{i(2\pi - k\theta)} = e^{i(n\theta - k\theta)} = e^{i\theta(n-k)}$. Moreover, if $\omega^k = e^{ik\theta} = a + bi$ where $b > 0$, then we have $e^{-ik\theta} = a - bi$. It follows that for any

complex nth root of unity of the form $a + bi$, where $b > 0$, the number $a - bi$ is also an complex nth root of unity.

Recall that $\mathcal{Z}(x) = \sum_{k=0}^{n} c_k x^k$, where $c_k \in \mathbb{R}$ are real numbers representing the partition function $\mathbf{Z}_{1,n}^k$ over all secondary structures of a given RNA sequence $s_1, \ldots, s_n$, whose base pair distance from initial structure $\mathcal{S}^*$ is k. Thus, in order to prove the lemma, it suffices to show that for all values $k = 0, \ldots, n-1$, if $a + bi$ is a complex nth root of unity, where $a, b \in \mathbb{R}$ and $b > 0$, and if $(a + bi)^k = C + Di$ where $C, D \in \mathbb{R}$, *then* $(a - bi)^k = C - Di$. Indeed, we have the following.

$$(a + bi)^m = \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (bi)^k \tag{4.29}$$

$$(bi)^k = \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ -ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases} \tag{4.30}$$

$$(a - bi)^m = \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (-bi)^k \tag{4.31}$$

$$(-bi)^k = \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ -ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases} \tag{4.32}$$

It follows that each term of the form $a^{m-k} \cdot (bi)^k$, for $k = 0, \ldots, m$, is the complex conjugate of $a^{m-k} \cdot (-bi)^k$, and thus $(a + bi)^m$ is the complex conjugate of $(a - bi)^m$. Since $\mathcal{Z}(a + bi)$ is a sum of terms of the form $c_k(a + bi)^k$, it follows that $\mathcal{Z}(a - bi)$ is the complex conjugate of $\mathcal{Z}(a + bi)$. This completes the proof of the lemma.

□

Lemma 4.2 immediately entails that we need only to evaluate $\mathcal{Z}(x)$ on $n/2$ many of the complex nth roots of unity—namely, those of the form $a + bi$, where $b \geq 0$. The remaining values of $\mathcal{Z}(x)$ are obtained by taking conplex conjugates of the first $n/2$ values. This, along with a precomputation of powers of the complex nth roots of unity, leads to an enormous performance speed-up in our implementation of `FFTbor`.

4.5 Coarse-grained kinetics with **FFTbor**

The output of `FFTbor`, as shown in Figure 4.3, is a probability distribution, where the x-axis represents the base pair distance from an arbitrary, but fixed secondary structure $\mathcal{S}^*$, and the y-axis represents the Boltzmann probability $p(k) = \frac{\mathbf{Z}^k}{\mathbf{Z}}$ that a secondary structure has base pair distance k from $\mathcal{S}^*$. Arguably, this probability distribution is an accurate one-dimensional projection of the rugged, high dimensional energy landscape near structure $\mathcal{S}^*$, of the sort artistically rendered in the well-known energy landscape depicted in Figure 1 of **?**. A hypothesis behind theoretical work in biomolecular folding theory in **?** is that kinetic folding slows down as the energy landscape becomes more *rugged*. This is borne out in our computational experiments for RNA using `FFTbor`, as reported in Figure 4.3.

We randomly chose two TPP riboswitch aptamers from the seed alignment for Rfam family RF00059. The first sequence has EMBL accession code BX842649.1/277414-277318 and is comprised of the 97 nt sequence ACCUGACGCUAGGGGUGUUGG UGAAUUCACCGACUGAGAAUAACCCUUUGAACCUGAUAGAGAUAAUGCUC GCGCAGGGAAGCAAGAAUAGAAAGAU, while the second sequence has EMBL accession code AACY022101973.1/389-487 and is comprised of the 99 nt sequence UAU AAGUCCAAGGGGUGCCAAUUGGCUGAGAUGGUUUUAACCAAUCCCUUUGA ACCUGAUCCGGUUAAUACCGGCGUAGGAAUGGAUUUUCUCUACAGC. Rfam consensus and minimum free energy structures for both sequences are depicted in Figure 4.2. Despite the fact that there is no sequence homology according to pairwise BLAST **?**, this figure clearly demonstrates that consensus and minimum free energy structures closely resemble each other, and that the structures of both TPP riboswitch aptamers are quite similar, with the exception of the leftmost hairpin loop [resp. multiloop]. The MFE structures differ from the consensus structures principally by the addition of base pairs not determined by covariation in the Rfam alignment. Indeed, if we let $\mathcal{S}_0, \mathcal{S}_1$ denote the Rfam consensus structure resp. MFE structure for the 97 nt sequence with EMBL accession code BX842649.1/277414-277318, then $\mathcal{S}_0 \setminus \mathcal{S}_1$ has 4 base pairs, and $\mathcal{S}_1 \setminus \mathcal{S}_0$ has 7 base pairs. If we let $\mathcal{T}_0, \mathcal{T}_1$ denote the Rfam consensus structure resp. MFE structure for the 99 nt sequence with EMBL accession code AACY022101973.1/389-487, then $\mathcal{T}_0 \setminus \mathcal{T}_1$ has 1 base pair, and $\mathcal{T}_1 \setminus \mathcal{T}_0$ has 5 base pairs.

We ran `FFTbor` on each of the TPP riboswitch aptamer sequences, with the MFE structure of each sequence taken as the initial structure $\mathcal{S}^*$ for that sequence. For the first sequence, BX842649.1/277414-277318, the `FFTbor` output suggests that there are low

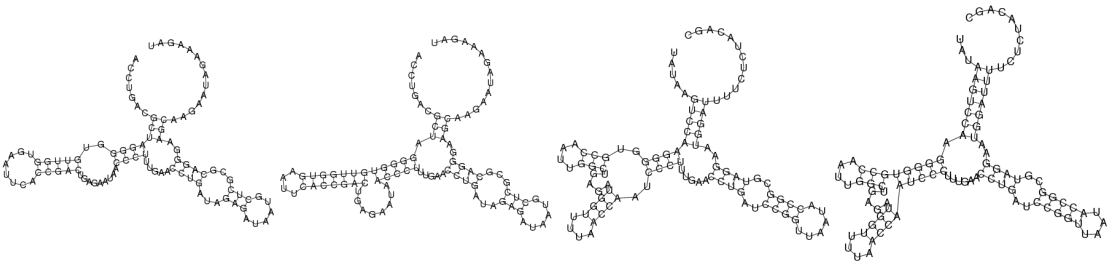

FIGURE 4.2: Rfam consensus structures (Rfam) and minimum free energy (MFE) secondary structures for two thiamine pyrophosphate (TPP) riboswitch aptamers, chosen at random from RF00059 Rfam family seed alignment **?**. Using pairwise BLAST **?**, there is no sequence similarity, although the secondary structures are very similar, as shown in this figure. From left to right: *(A)* Rfam consensus structure for BX842649.1/277414-277318. *(B)* MFE structure for BX842649.1/277414-277318. *(C)* Rfam consensus structure for AACY022101973.1/389-487. *(D)* Rfam consensus structure for AACY022101973.1/389-487.

energy structures at a distance from the MFE structure, which might compete with the MFE structure and hence slow the kinetics of folding. In contrast, for the second sequence, AACY022101973.1/389-487, the `FFTbor` output suggests that there are no such competing low energy structures, hence the second sequence should fold more quickly than the first.

To test the hypothesis that folding is slower for rugged energy landscapes, we ran the kinetic folding software, `Kinfold` **?**, on each of the two TPP riboswitch aptamer sequences, BX842649.1/277414-277318 and AACY022101973.1/389-487, to determine the mean first passage time (MFPT) to fold into the MFE structure, when starting from the empty structure. In this computational experiment, we took MFPT to be the average number of Monte Carlo steps taken by `Kinfold`, each step consisting of the addition or removal of a single base pair (or shift – see **?**), to fold the empty structure into the MFE structure, where the average was taken over 30 runs, with an absolute maximum number of Monte Carlo steps taken to be $500,000$. The first sequence, BX842649.1/277414-277318, converged within $500,000$ steps only for 20 out of 30 runs. Assigning the maximum step count of $500,000$ for the 10 runs that did not converge, we found a mean first passage time of $311,075.06$ steps for this sequence. The second sequence, AACY022101973.1/389-487, converged within $500,000$ steps in 29 out of 30 runs, and we found a mean first passage time of $61,575.69$ steps for this sequence. From computational experiments of this type, it is suggestive that `FFTbor` may prove useful

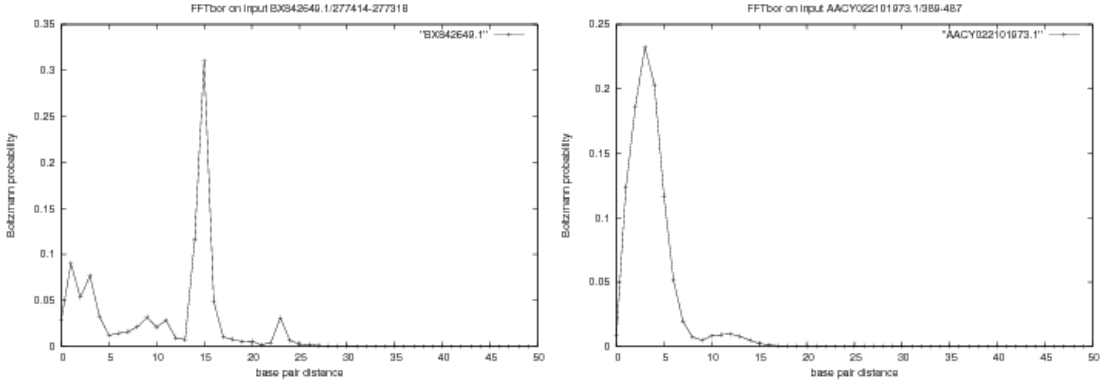

FIGURE 4.3: Output from `FFTbor` on two randomly selected thiamine pyrophosphate riboswitch (TPP) aptamers, taken from the Rfam database **?**. The x-axis represents base pair distance from the minimum free energy structure for each given sequence; the y-axis represents Boltzmann probabilities $p(k) = \mathbf{z}^k/\mathbf{z}$, where $\mathbf{Z}^k$ denotes the sum of Boltzmann factors or all secondary structures, whose base pair distance from the MFE structure is exactly k. *(Left)* The 97 nt sequence BX842649.1/277414-277318 appears to have a rugged energy landscape near its minimum free energy structure, with distinct low energy structures that may compete with the MFE structure during the folding process. *(Right)* The 99 nt sequence, AACY022101973.1/389-487 appears to have a smooth energy landscape near its MFE structure, with no distinct low energy structures to might compete with the MFE structure. Based on the `FFTbor` output or *structural profile* near MFE structure $\mathcal{S}^*$, one might expect folding time for the first sequence to increase due to competition from metastable structures, while one might expect the second sequence to have rapid folding time. Computational Monte Carlo folding experiments bear out this fact. `Kinfold ?` simulations clearly show that the second sequence folds at least four times more quickly than the first sequence. See section 4.5 for details.

in synthetic biology, where one would like to design rapidly folding RNA molecules that fold into a designated target structure.

In order to more systematically determine the relation between kinetic folding speed and the ruggedness of an energy landscape near the MFE structure, we need to numerically quantify ruggedness. To this end, in the following we define the notion of expected base pair distance to a designated structure. Let $\mathcal{S}^*$ be an arbitrary secondary structure of the RNA sequence $\mathbf{s} = s_1, \ldots, s_n$. The expected base pair distance to $\mathcal{S}^*$ is defined by

$$E[\{d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) : \mathcal{S} \in \mathbb{S}(s_1, \ldots, s_n)\}] = \sum_{\mathcal{S}} P(\mathcal{S}) \cdot d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) \tag{4.33}$$

where $\mathbb{S}(s_1, \ldots, s_n)$ denotes the set of secondary structures for $\mathbf{s} = s_1, \ldots, s_n$, $P(\mathcal{S}) = \frac{\exp(-E(\mathcal{S})/RT)}{\mathbf{Z}}$ is the Boltzmann probability of $\mathcal{S}$, and $d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*)$ denotes base pair distance between $\mathcal{S}$ and $\mathcal{S}^*$. If we run `FFTbor` on an input sequence $\mathbf{s}$ and secondary structure $\mathcal{S}^*$, then clearly $E[\{d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) : \mathcal{S} \in \mathbb{S}(s_1, \ldots, s_n)\}] = \sum_k k \cdot p(k)$, where $p(k) = \frac{\mathbf{Z}^k}{\mathbf{Z}}$, obtained from the program output. If $\mathcal{S}^*$ is the empty structure, then `FFTbor` output is simply the probability distribution of the number of base pairs per secondary structure, taken over the Boltzmann ensemble of all structures.

For the benchmarking assay, we took all 61 selenocysteine insertion sequence (SECIS) sequences from the seed alignment of Rfam family RF00031 **?**. Average length was 64.32 ± 2.83 nt. For each sequence, we ran both `FFTbor` (when starting from the empty structure rather than the MFE structure) and a Monte Carlo folding algorithm, developed by E. Freyhult and P. Clote (unpublished). Using the Monte Carlo algorithm, we determined the mean first passage time (MFPT), defined as the average taken over 50 runs, of the number of Monte Carlo steps taken to fold the empty structure into the MFE structure, where an absolute upper bound of 5 million steps was allowed in the simulation.

As described above, `FFTbor` output is simply the probability distribution for the number of base pairs per structure, taken over the ensemble of all secondary structure for the input RNA sequence. Surprisingly, we found that there is a significant correlation of 0.48436192 with one-tailed p-value of 0.00018249 between the standard deviation of the `FFTbor` output (when starting from the empty structure) and logarithm base 10 of the mean first passage time.

	μ	σ	$\frac{\sigma}{\mu}$	n	MFE	$\log_{10}$(MFPT)
μ	1					
σ	-0.43722448	1				
$\frac{\sigma}{\mu}$	-0.691411183	0.943650913	1			
n	0.707683898	-0.158951202	-0.364591789	1		
MFE	-0.569474125	0.739515083	0.759622716	-0.368485646	1	
$\log_{10}$(MFPT)	-0.036291124	0.48436192	0.376230235	0.405865529	0.399015556	1

TABLE 4.1: Pearson correlation between various aspects of selenocysteine insertion sequences from the seed alignment of Rfam family RF00031 **?**. For each of the 61 RNA sequences, we ran `FFTbor`, starting from empty initial structure $\mathcal{S}^*$, and we ran a Monte Carlo folding algorithm, developed by E. Freyhult and P. Clote (unpublished). Using the Monte Carlo algorithm, we determined the mean first passage time (MFPT), defined as the average taken over 50 runs, of the number of Monte Carlo steps taken to fold the empty structure into the MFE structure, where an absolute upper bound of 5 million steps was allowed in the simulation. From the output of `FFTbor`, we computed *(1)* the mean number (μ) of base pairs per structure, taken over the ensemble of all secondary structures for the given sequence, *(2)* the standard deviation (σ) of the number of base pairs per structure, *(3)* the coefficient of variation $\frac{\sigma}{\mu}$, *(4)* the RNA sequence length n, and *(5)* the minimum free energy (MFE). Additionally, we computed the logarithm base 10 of mean first passage time ($\log_{10}$(MFPT)), taken over 50 Monte Carlo runs per sequence (log base 10 of the standard deviation of number of Monte Carlo steps per run was approximately 9% of $\log_{10}$(MFPT) on average). The table shows the correlation between each of these aspects. Some correlations are obvious – for example, *(i)* the standard deviation σ is highly correlated with the coefficient of variation $\frac{\sigma}{\mu}$; *(ii)* the mean μ is negatively correlated with the coefficient of variation $\frac{\sigma}{\mu}$; *(iii)* the mean μ is negatively correlated with the minimum free energy (MFE) – if most low energy structures in the ensemble have many base pairs, then it is likely that the minimum free energy is very low (i.e. since MFE is negative, the absolute value of MFE increases); *(iv)* sequence length is negatively correlated with MFE – as sequence length increases, the minimum free energy (MFE) decreases. However, it may appear surprising that *(v)* the mean μ number of base pairs per structure is independent of MFPT (correlation -0.036291124), although *(vi)* MFE is correlated with MFPT (correlation 0.399015556) – i.e. from *(iii)*, lower MFE is correlated with a larger average μ number of base pairs per structure, from *(vi)* higher MFE is correlated with longer folding time, but from *(v)* the average μ number of base pairs per structure is independent of folding time. The most important insight from this table is that *(vii)* standard deviation σ is correlated with mean first passage time – the correlation is statistically significant, with one-tailed p-value of 0.00018249.

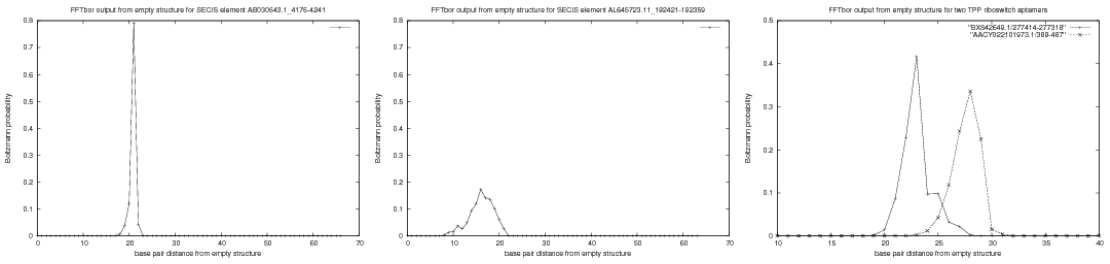

FIGURE 4.4: This figure represents the graphical output of `FFTbor`, when the empty structure is chosen as initial structure $\mathcal{S}^*$. The x-axis represents the number of base pairs per structure, taken over the ensemble of all secondary structures for the given RNA sequence; the y-axis represents Boltzmann probability $p(k) = \mathbf{Z}^k/\mathbf{Z}$, where $\mathbf{Z}$ is the partition function for all secondary structures having exactly k base pairs. *(Left)* For the selenocysteine (SECIS) element AB030643.1/4176-4241 from Rfam family RF00031, the standard deviation σ of the number of base pairs, taken over the ensemble of all secondary structures, is 0.727618, while the logarithm base 10 of the mean first passage time ($\log_{10}$(MFPT)) is 4.75. *(Center)* For the selenocysteine (SECIS) element AL645723.11/192421-192359 from Rfam family RF00031, the standard deviation σ of the number of base pairs, taken over the ensemble of all secondary structures, is 2.679446, while $\log_{10}$(MFPT) is 5.69. Among the 61 sequences in the seed alignment of RF00031, AB030643.1/4176-4241 was the fastest folder, while AL645723.11/192421-192359 was the slowest folder. *(Right)* Superimposition of output of `FFTbor` for two TPP riboswitch aptamers: the 97 nt sequence BX842649.1/277414-277318 and the 99 nt sequence AACY022101973.1/389-487, both obtained when taking the empty structure for the initial structure $\mathcal{S}^*$. The mean μ for the `FFTbor` structural profile near the empty structure is 23.0203 [resp. 27.5821], the standard deviation σ for the `FFTbor` structural profile is 2.22528791 [resp. 1.98565959], and the `Kinfold` MFPT is $311,075.06$ [resp. $61,575.69$] for the TPP riboswitch aptamer AB030643.1/4176-4241 [resp. AL645723.11/192421-192359]. The right panel of this figure should be compared with Figure 4.3. These anecdotal results bear up the correlation between standard deviation σ and $\log_{10}$(MFPT) described in 4.1.

In the right panel of Figure 4.4, we applied `FFTbor` to each of the two randomly chosen TPP riboswitch aptamers BX842649.1/277414-277318 and AACY022101973.1/389-487, starting from the empty reference structure $\mathcal{S}^* = \emptyset$. The mean for the `FFTbor` structural profile near the empty structure is $\mu_1 = 23.0203$ [resp. $\mu_2 = 27.5821$], the standard deviation σ for the `FFTbor` structural profile is $\sigma_1 = 2.22528791$ [resp. $\sigma_2 = 1.98565959$],

and the `Kinfold` MFPT is $311,075.06$ [resp. $61,575.69$] for the TPP riboswitch aptamer AB030643.1/4176-4241 [resp. AL645723.11/192421-192359]. This anecdotal evidence supports the hypothesis that small standard deviation in `FFTbor` distribution is correlated with fast folding.

Additionally, we randomized the TPP riboswitches BX842649.1/277414-277318 and AACY022101973.1/389-487 by using our implementation of the Altschul-Erikson dinucleotide shuffle algorithm `?`, and then applied `FFTbor` to these sequences, starting from the empty structure. The mean μ_1 and standard deviation σ_1 for the `FFTbor` distribution for randomized BX842649 are respectively $\mu_1 = 19.93$ and $\sigma_1 = 2.88$, while those for randomized AACY022101973 are $\mu_2 = 24.39$ and $\sigma_2 = 24.00$. Running `Kinfold`, with a maximum of $500,000$ steps with 30 replicates (as explained in the text), we found that for randomized BX842649, all 30 runs converged yielding a mean first passage time (MFPT) of $13,022.58$ with standard deviation of $15,221.78$. In contrast for randomized AACY022101973, only 15 out of 30 runs converged within $500,000$ steps, and discounting these nonconvergent data, we obtain an average mean first passage time (MFPT) of $94,446.93$ with standard deviation of $157,107.43$. This additional test provides more anecdotal evidence supporting our hypothesis that small standard deviation σ in `FFTbor` probability density is correlated with fast folding, as measured by MFPT.

This notion of correlation between the coarse-grained energy landscape and kinetics is what motivates the work described in Chapters 5 and 6, where a more detailed explanation of kinetics is provided, and additional evidence is provided to support this claim.

4.6 Performance characteristics of `FFTbor`

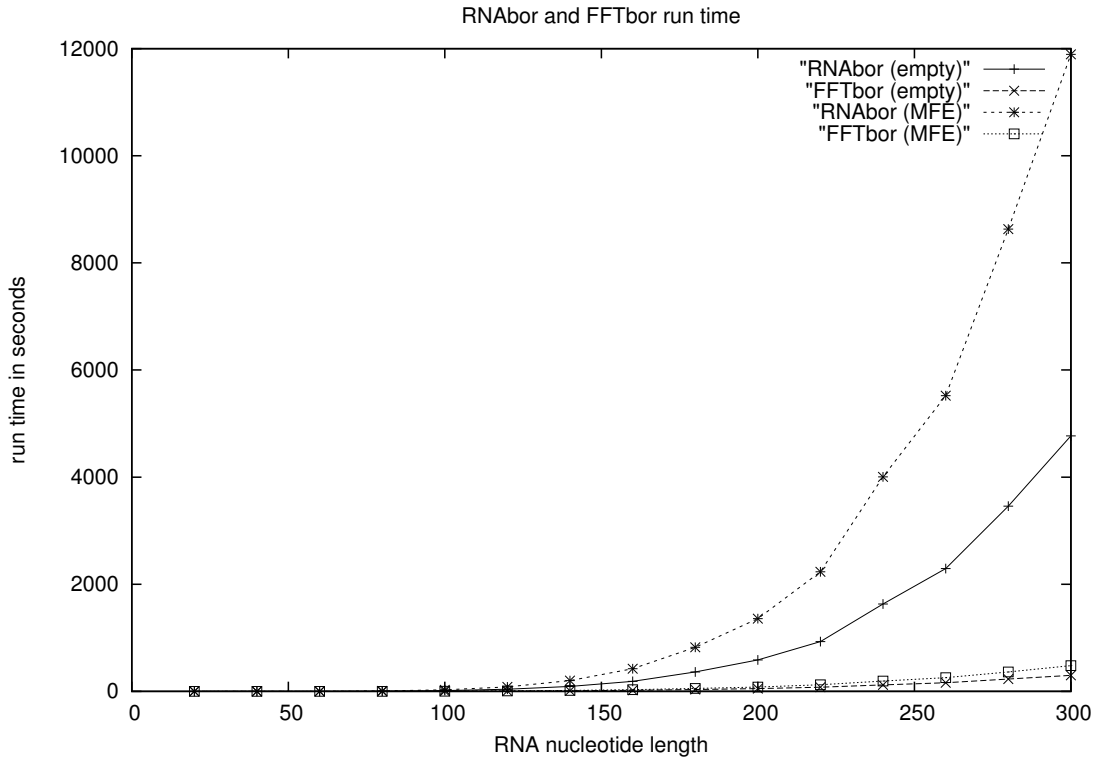

FIGURE 4.5: Run times in seconds for `RNAbor` and `FFTbor`, on random RNA of length $20, 40, 60, \ldots, 300$ in step size of 20 nt. Each algorithm was run with the empty initial structure $\mathcal{S}^*$, see rows `RNAbor` (empty), `FFTbor` (empty), and with the minimum free energy structure as the initial structure $\mathcal{S}^*$, see rows `RNAbor` (MFE) and `FFTbor` (MFE). Note that for both `RNAbor` and `FFTbor`, the run time increases when $\mathcal{S}^*$ is the MFE structure, rather than the empty structure. Notice the radical improvement in the run time of `FFTbor` over that of `RNAbor`.

Chapter 5

FFTbor2D

5.1 Introduction

In this chapter, we present the `FFTbor2D` algorithm and accompanying software. `FFTbor2D`, like `FFTbor` described in Chapter 4, is an algorithm which computes the paramerized partition function for an input RNA sequence `s`. `FFTbor2D` computes the two-dimensional coarse energy landscape for `s` given two compatible input secondary structures $\mathcal{A}$ and $\mathcal{B}$, where position (x, y) on the discrete energy landscape corresponds to the Boltzmann probability for those structures $\mathcal{S}$ which have $d_{\text{BP}}((,\mathcal{S}),\mathcal{A}) = x$ and $d_{\text{BP}}((,\mathcal{S}),\mathcal{B}) = y$ (where $d_{\text{BP}}(,\text{i})$ s as defined in equation 4.3). By again leveraging the Fast Fourier Transform, `FFTbor2D` runs in $O(n^5)$ time and only uses $O(n^2)$ space—a significant improvement over previous approaches. This permits the output energy landscape to be used in a high-throughput fashion to analyze folding kinetics; a topic covered in detail in Chapter 6.

5.1.1 Organization

This chapter is organized in the following fashion. Because the history for this work arises naturally from the background described in section **??**, we provide only a brief background and immediately fall into a technical discussion of the underlying algorithm. We first develop the recursions for the Nussinov energy model for expository clarity, the underlying implementation uses the more complicated and robust Turner energy model. Recursions in place, we then move to show how these lead to a single variable

polynomial $P(x)$ whose coeffecients can be computed by the inverse Discrete Fourier Transform, and map to the 2D energy landscape. We describe two exploitations of $P(x)$, a parity condition and complex conjugates which further reduce the runtime by a factor of 4. Finally, we contrast this software against `RNA2Dfold`, and outline the performance characteristics of both softwares and highlight the benefits and drawbacks of both.

5.2 Derivation of the **FFTbor2D** algorithm

For expository clarity, we describe `FFTbor2D` and all recursions in terms of the Nussinov energy model **?** (same as in Chapter 4), where the energy $E_0(i,j)$ of a base pair (i,j) is defined to be -1, and the energy $E(\mathcal{S})$ of a secondary structure $\mathcal{S}$ is -1 times the number $|\mathcal{S}|$ of base pairs in structure $\mathcal{S}$. Nevertheless, the implementation of `FFTbor2D` involves the full Turner energy model **?**, where free energy $E(\mathcal{S})$ depends on negative, stabilizing energy contributions from base stacking, and positive, destabilizing energy contributions due to loss of entropy in loops.

5.2.1 Definition of the partition function $\mathbf{Z}_{1,n}^{x,y}$

Given reference secondary structures $\mathcal{A}, \mathcal{B}$ of a given RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, our goal is to compute

$$\mathbf{Z}_{1,n}^{x,y} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=x,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=y}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{5.1}$$

for all $0 \leq x, y < n$, where R is the universal gas constant, T is absolute temperature, $E(\mathcal{S})$ denotes the free energy of $\mathcal{S}$, and $\mathcal{S}$ ranges over all secondary structures that are compatible with $\mathbf{s}$. As mentioned, we emphasize that for expository reasons alone, the Nussinov energy model is used in the recursions in this paper, although full recursions and the implementation of `FFTbor2D`, like `FFTbor`, involve the Turner energy model.

For any secondary structure $\mathcal{S}$ of $\mathbf{s}$, and any values $1 \leq i \leq j \leq n$, the restriction $\mathcal{S}_{[i,j]}$ is defined to be the collection of base pairs of $\mathcal{S}$, lying within interval $[i,j]$; i.e. $\mathcal{S}_{[i,j]} = \{(k,\ell) : i \leq k < \ell \leq j\}$. In **?**, Lorenz et al. generalized the dynamic programming recursions of our earlier work **?**, to yield recursions for the partition function $\mathbf{Z}_{i,j}^{x,y}$ in equation (5.1). In the context of the Nussinov model, $\mathbf{Z}_{i,j}^{x,y}$ is equal to

$$\mathbf{Z}_{i,j-1}^{x-\alpha_0,y-\beta_0} + \sum_{\substack{s_k,s_j\in\mathbb{B},\\ i\le k<j}} \left(e^{\frac{-E_0(k,j)}{RT}} \sum_{u+u'=x-\alpha(k)} \sum_{v+v'=y-\beta(k)} \mathbf{Z}_{i,k-1}^{u,v} \cdot \mathbf{Z}_{k+1,j-1}^{u',v'} \right) \tag{5.2}$$

where $\alpha_0 = 1$ if j is base paired in $\mathcal{A}_{[i,j]}$ and 0 otherwise, $\beta_0 = 1$ if j is base paired in $\mathcal{B}_{[i,j]}$ and 0 otherwise, $E_0(k,j) = -1$ if k, j can base-pair (see equation 4.2), and otherwise $E_0(k,j) = 0$, and $\alpha(k) = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i,k-1]} \cup \mathcal{A}_{[k+1,j-1]} \cup \{(k,j)\})$, and $\beta(k) = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k+1,j-1]} \cup \{(k,j)\})$.

5.2.2 Recursions to compute the polynomial $\mathcal{Z}_{i,j}(x)$

Given RNA sequence $\mathbf{s} = s_1, \ldots, s_n$ and two arbitrary, but fixed reference structures $\mathcal{A}, \mathcal{B}$, we define the *polynomial*

$$\mathcal{Z}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{5.3}$$

where (constant) coefficients

$$z_{rn+s} = \mathbf{Z}_{1,n}^{r,s} = \sum_{\substack{\mathcal{S} \text{ such that}\\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{5.4}$$

where $E(\mathcal{S})$ denotes the free energy of $\mathcal{S}$. If we evaluate the polynomial $\mathcal{Z}(x)$ at n^2 distinct pairs of values $a_0, \ldots, a_{n^2-1}$ in

$$\mathcal{Z}(a_0) = y_0, \ldots, \mathcal{Z}(a_{n^2-1}) = y_{n^2-1}, \tag{5.5}$$

then Lagrange polynomial interpolation (equation 4.14) guarantees that we can determine the coefficients c_{rn+s} of $\mathcal{Z}(x)$, for $0 \le r, s < n$. Due to technical difficulties concerning numerical robustness observered while working on the `FFTbor` software (Chapter 4), we will perform polynomial interpolation by using Vandermonde matrices and the Fast Fourier Transform (FFT).

The following theorem shows that a recursion, analogous to equation (5.2), can be used to compute the *polynomial* $\mathcal{Z}_{i,j}(x)$ defined by

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s}(i,j) \cdot x^{rn+s} = \sum_{k=0}^{n^2-1} z_k(i,j) \cdot x^k \tag{5.6}$$

where

$$z_{rn+s}(i,j) = \mathbf{Z}_{i,j}^{r,s} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}}. \tag{5.7}$$

Here, in the summation, $\mathcal{S}$ runs over structures on $s_i, \ldots, s_j$, which are r-neighbors of the restriction $\mathcal{A}_{[i,j]}$ of reference structure $\mathcal{A}$ to interval $[i,j]$, and simultaneously $\mathcal{S}$-neighbors of the restriction $\mathcal{B}_{[i,j]}$ of reference structure $\mathcal{B}$ to interval $[i,j]$.

Theorem 5.1. *Let $s_1, \ldots, s_n$ be a given RNA sequence. For any integers $1 \le i < j \le n$, let*

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{5.8}$$

where

$$z_{rn+s}(i,j) = \mathbf{Z}_{i,j}^{r,s}. \tag{5.9}$$

Inductively we define $\mathcal{Z}_{i,j}(x)$ to equal

$$\mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n + \beta_0} + \tag{5.10}$$

$$\sum_{\substack{s_k, s_j \in \mathbb{B}, \\ i \le k < j}} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \cdot x^{\alpha(k)n + \beta(k)} \right) \tag{5.11}$$

where $\alpha_0 = 1$ if j is base-paired in $\mathcal{A}_{[i,j]}$ and 0 otherwise, $\beta_0 = 1$ if j is base-paired in $\mathcal{B}_{[i,j]}$ and 0 otherwise, and $\alpha(k) = d_{BP}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i,k-1]} \cup \mathcal{A}_{[k+1,j-1]} \cup \{(k,j)\})$, $\beta(k) = d_{BP}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k+1,j-1]} \cup \{(k,j)\})$.

The proof is given in supplementary information.

Note that if one were to compute all terms of the polynomial $\mathcal{Z}_{1,n}(x)$ by explicitly performing polynomial multiplications, then the computation would require $O(n^7)$ time and $O(n^4)$ space, the same time complexity of **?**. Instead of explicitly performing polynomial expansion in *variable* x, we instantiate x to a complex number $\rho \in \mathbb{C}$, and apply the following recursion, by setting $\mathcal{Z}_{i,j}(\rho)$ equal to

$$\mathcal{Z}_{i,j-1}(\rho) \cdot \rho^{\alpha_0 n+\beta_0} + \sum_{\substack{(s_k,s_j)\in\mathbb{B},\\ i\leq k<j}} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(\rho) \cdot \mathcal{Z}_{k+1,j-1}(\rho) \cdot \rho^{\alpha(k)n+\beta(k)} \right) \tag{5.12}$$

Note that this approach is similar to what we do in `FFTbor`—specifically equation (4.23)—however notationally we will use the variable ρ instead of α, to avoid confusion. In this fashion, we can compute $\mathcal{Z}(\rho) = \mathcal{Z}_{1,n}(\rho)$ in $O(n^3)$ time and $O(n^2)$ space. For n^2 distinct complex numbers ρ_i where $0 \leq i \leq n^2 - 1$, we can compute and save only the values $\mathcal{Z}(\rho_0), \ldots, \mathcal{Z}(\rho_{n^2-1})$, each time re-using the $O(n^2)$ space for the next computation of $\mathcal{Z}(\rho_i)$. It follows that the computation resources used to determine the (column) vector

$$\mathbf{Y} = (y_0, \ldots, y_{n^2-1})^{\mathrm{T}} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n^2-1} \end{pmatrix} \tag{5.13}$$

where

$y_0 = \mathcal{Z}(\rho_0), \ldots, y_{n^2-1} = \mathcal{Z}(\rho_{n^2-1}))$ are thus quintic time $O(n^5)$ and quadratic space $O(n^2)$.

5.2.3 Polynomial interpolation

Our plan is to determine the coefficients of the polynomial $\mathcal{Z}(x)$ in equation (5.3) by polynomial interpolation. For reasons of numerical stability, we instead determine the coefficients of the polynomial $p(x)$, defined by

$$p(x) = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} p_{rn+s}x^{rn+s} = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} \frac{\mathbf{Z}_{1,n}^{rn+s}}{\mathbf{Z}} x^{rn+s}, \tag{5.14}$$

where the Fast Fourier Transform (FFT) is used to implement the interpolation of the coefficients using the inverse Discrete Fourier Transform (DFT), as described in section 5.3.3. The following pseudocode describes how to compute the m most significant digits for probabilities $p_{rn+s} = \frac{\mathbf{Z}_{1,n}^{r,s}}{\mathbf{Z}}$. It is well-known that the FFT requires $O(N \log N)$ time to solve the inverse Discrete Fourier Transform for a polynomial of degree N. In our case, $N = n^2$, and so the computation involving the FFT requires time $O(n^2 \log n)$.

The pseudocode for the algorithm to compute $p(x)$ is given in Figure **??**. In the next section, we explain a highly non-trivial improvement of this algorithm to reduce time by a factor of 4.

5.3 Acceleration of the `FFTbor2D` algorithm

Recall that if $a + bi$ is a complex number, where a, b are real values and i denotes $\sqrt{-1}$, then the complex conjugate of $a + bi$, denoted by $\overline{a + bi}$ is defined to be $a - bi$. Recall that a complex nth root of unity is a number whose nth power equals one. Moreover, $e^{2\pi i/n}$ is the *principal* complex nth root of unity; i.e. $\{e^{2\pi ik/n} : k = 0, \ldots, n-1\}$ is a set of pairwise distinct complex nth roots of unity. We have the following.

Lemma 5.2. *Let $\mathcal{A}, \mathcal{B}$ denote two distinct, arbitrary but fixed, secondary structures of RNA sequence* $\mathbf{s}$*, let $\mathcal{S}$ range over all secondary structures of* $\mathbf{s}$*, and let d_0 denote $d_{BP}(\mathcal{A}, \mathcal{B})$. If $x = d_{BP}(\mathcal{A}, \mathcal{S})$ and $y = d_{BP}(\mathcal{S}, \mathcal{B})$, then $y \in \{d_0 - x + 2k : k = 0, \ldots, x\}$.*

It follows that if $x = d_{BP}(\mathcal{A}, \mathcal{S})$ and $y = d_{BP}(\mathcal{S}, \mathcal{B})$, then the only possible values for (x, y) are $(0, d_0), (1, d_0 - 1), (1, d_0 + 1), (2, d_0 - 2), (2, d_0), (2, d_0 + 2), (3, d_0 - 3), (3, d_0 - 1), (3, d_0 + 1), (3, d_0 + 3), \ldots$. As a corollary, we have the parity condition*, that*

$$d_{BP}(\mathcal{A}, \mathcal{S}) + d_{BP}(\mathcal{S}, \mathcal{B}) \equiv d_{BP}(\mathcal{A}, \mathcal{B}) \bmod 2 \tag{5.15}$$

first noticed in **?***, as well as the triangle inequality $d_{BP}(\mathcal{A}, \mathcal{S}) + d_{BP}(\mathcal{S}, \mathcal{B}) \geq d_{BP}(\mathcal{A}, \mathcal{B})$ for base pair distance, probably folklore. Lorenz et al.* **?** *exploited the parity condition and the triangle inequality by using sparse matrix methods to improve on the efficiency of the naïve implementation of the $O(n^7)$ time and $O(n^4)$ space algorithm to compute the*

Pseudocode for `FFTbor2D`

PURPOSE: Computes the m most significant digits of probabilities $p_{rn+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$
INPUT: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structures $\mathcal{A}$, $\mathcal{B}$ of $\mathbf{s}$, integer m
OUTPUT: Probabilities $p_{rn+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$ to m significant digits for $r, s = 0, \ldots, n-1$

```
1  function FFTBOR2D(s, A, B, m)
2      n ← length(s)
3      for k ← 0, n^2 − 1 do                    ▷ Compute all complex n^2 roots of unity
4          ω_k ← exp(2πik/n^2)
5      end for
6      for k ← 0, n^2 − 1 do                    ▷ Note that Z(ω_0) = Z
7          y_k ← 2^m · Z(ω_k)/Z(ω_0)
8      end for
9      for k ← 0, n^2 − 1 do                    ▷ Compute IDFT from equation (5.25)
10         a_k (1/n) Σ_{j=0}^{n^2−1} a_j ω^{−kj}
11         p_k ← 2^{−m} · ⌊a_k⌋                 ▷ Truncate to m significant digits
12     end for
13     return p_0, ..., p_{n^2−1}    ▷ Return all p_k for 0 ≤ k < n^2 − 1, from equation (??)
14 end function
```

FIGURE 5.1: Pseudocode to compute the m most significant digits for probabilities $p_{rn+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$. In our implementation, due to numerical stability issues in the FFT engine, precision parameter m has an upper bound of 8 – only the $m = 8$ most significant digits are computed with `FFTbor2D`. (Note that the software actually usesbase 2 precision parameter, with maximum of 27, where $2^{27} \approx 10^8$) It is well-known that the FFT requires $O(N \log N)$ time to solve the inverse discrete Fourier transform for a polynomial of degree N. In our case, $N = n^2$, and so the FFT requires time $O(n^2 \log n)$.

partition function, $\mathbf{Z}_{1,n}^{r,s}$, and minimum free energy structure, $MFE_{1,n}^{r,s}$, over all structures having base pair distance r to $\mathcal{A}$ and $\mathcal{S}$ to $\mathcal{B}$. The following lemma is not difficult to establish.

Lemma 5.3. *If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (5.3), then for any complex nth root of unity α, it is the case that $\mathcal{Z}(\overline{\alpha}) = \overline{\mathcal{Z}(\alpha)}$.*

Lemma 5.4. *Let $\mathcal{Z}(x)$ be defined by equation (5.3), and let $\alpha \in \mathbb{C}$ be any complex number. If the base pair distance between reference structures $\mathcal{A}, \mathcal{B}$ is even, then $\mathcal{Z}(-\alpha) = \mathcal{Z}(\alpha)$, while if the distance is odd, then $\mathcal{Z}(-\alpha) = -\mathcal{Z}(\alpha)$.*

Lemma 5.5. *Suppose that M is evenly divisible by 4, $\nu = \exp(\frac{2\pi i}{M})$ is the principal M-root of unity, and $\frac{M}{4} < k \leq \frac{M}{2}$. Then*

$$\nu^k = -(\nu^{-(M/2-k)}) = -\overline{\nu^{M/2-k}}. \tag{5.16}$$

Lemma 5.2 is proved by simple induction; Lemma 5.3 is proved by a computation involving binomial coefficients; Lemma 5.4 is immediate by the parity observation above, resulting from Lemma 5.2; Lemma 5.5 is elementary, relying on Euler's formula and trigonometric addition formulas. Details proofs of Lemmas 5.3, 5.4, 5.5 can be found in supplementary information.

Lemma 5.2 entails that either all even coefficients, or all odd coefficients of $\mathcal{Z}(x)$ are zero, and so by a variable change described in detail below, we require only half the number of evaluations of $\mathcal{Z}(x)$, in order to perform polynomial interpolation. Lemma 5.3 entails that we require only half again the number of evaluations of $\mathcal{Z}(x)$, since the remainder can be inferred by taking the complex conjugate. Lemma 5.2 and Lemma 5.3, along with a precomputation of powers of the complex roots of unity, lead to a large performance speed-up in our implementation of `FFTbor2D`—by a factor of 4 or more.

5.3.1 Optimization due to parity condition

Let n denote the length of RNA sequence $\mathbf{s}$, and let N denote the least *even* integer greater than n. Since N is even, we have $(r+s) \equiv (r \cdot (N+1) + s) \bmod 2$. For distinct fixed structures $\mathcal{A}, \mathcal{B}$, let $\pi_1(k) = \lfloor \frac{k}{N+1} \rfloor$, and $\pi_2(k) = k \bmod (N+1)$, and define the polynomial

$$\begin{aligned} \mathcal{Z}(x) &= \sum_{r=0}^{N} \sum_{s=0}^{N} z_{rN+s} x^{rN+s} \\ &= \sum_{k=0}^{(N+1)^2-1} z_{\pi_1(k)\cdot(N+1)+\pi_2(k)} x^{\pi_1(k)\cdot(N+1)+\pi_2(k)} \\ &= \sum_{k=0}^{(N+1)^2-1} z_k x^k \end{aligned} \tag{5.17}$$

where for the last equality, we have used the fact that $k = \pi_1(k) \cdot (N+1) + \pi_2(k)$, well-known from row major order of a 0-indexed 2-dimensional array.

Consider the coefficients of the polynomial

$$\mathcal{Z}(x) = \sum_{r=0}^{N} \sum_{s=0}^{N} z_{rN+s} x^{rN+s} \quad = \sum_{k=0}^{(N+1)^2-1} z_k x^k. \tag{5.18}$$

Since N is even, the parity of $r+s$ equals the parity of $r(N+1)+s$, hence it follows from the parity condition that either *(i)* all coefficients $z_1, z_3, z_5, \ldots$ of odd parity are zero, or *(ii)* all coefficients $z_0, z_2, z_4, \ldots$ of even parity are zero. To simplify notation, in the remainder of this subsection, let M be the least integer greater than or equal to $(N+1)^2$ that is evenly divisible by 4, and let $M_0 = M/2$. We will assume that $\mathcal{Z}(x) = \sum_{k=0}^{M-1} z_k x^k$, whereupon coefficients $z_k = 0$ for $k > (N+1)^2$.

Case 1: All coefficients z_k of odd parity in equation (5.18) are zero.

In this case, we have $\mathcal{Z}(x) = \sum_{k=0}^{\frac{M}{2}-1} z_{2k} x^{2k}$. But then $\mathcal{Z}(x) = Y(u) = \sum_{k=0}^{M_0-1} b_k u^k$, where we have made a variable change $u = x^2$, and coefficient changes $b_k = a_{2k}$. By evaluating $M_0 = \frac{M}{2}$ many complex M_0-roots of unity, we can use polynomial interpolation to determine all coefficients b_k of the polynomial

$$Y(u) = \sum_{k=0}^{M_0-1} b_k u^k = \sum_{k=0}^{M_0-1} z_{2k} x^{2k}. \tag{5.19}$$

Since $Y(x^2) = \mathcal{Z}(x)$, we have $Y(\exp(\frac{2\pi i k}{M/2})) = Y(\exp(\frac{4\pi i k}{M})) = \mathcal{Z}(\exp(\frac{2\pi i k}{M}))$, hence we can use the previous recursions (5.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi i k}{M}))$. Instead of performing M evaluations of $\mathcal{Z}(x)$ at M-roots of unity, this requires only $M_0 = M/2$ evaluations of $Y(u)$ at M_0-roots of unity; i.e. only half the number of evaluations of $\mathcal{Z}(x)$ are necessary to obtain the coefficients of $Y(x)$. But then, we immediately obtain the full polynomial $\mathcal{Z}(x)$, since its coefficients of odd parity are zero.

Case 2: All coefficients z_k of even parity in equation (5.18) are zero.

In this case, $z_0, z_2, z_4, \ldots$ are zero, so $\mathcal{Z}(x) = \sum_{k=0}^{M/2-1} z_{2k+1} x^{2k+1}$. But then $\mathcal{Z}(x) = x \cdot Y(u)$, where $Y(u) = \sum_{k=0}^{M_0-1} b_k u^k$, where we have made a variable change $u = x^2$, and coefficient changes $b_k = z_{2k+1}$. Similarly to Case 1, we can interpolate the M_0

coefficients of the polynomial $Y(u) = \sum_{k=0}^{M_0-1} b_k u^k$ by evaluating M_0 many complex M_0-roots of unity. Since $\mathcal{Z}(x) = x \cdot Y(x^2)$, $Y(x^2) = x^{-1} \cdot \mathcal{Z}(x)$, so $Y(\exp(\frac{2\pi ik}{M/2})) = Y(\exp(\frac{4\pi ki}{M})) = \exp(\frac{-2\pi ik}{M}) \cdot \mathcal{Z}(\exp(\frac{2\pi ik}{M}))$, employing the previous recursions (5.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi ik}{M}))$. Note, that unlike the Case 1, since $\mathcal{Z}(x) = x \cdot Y(x^2)$, we have $Y(x^2) = \frac{\mathcal{Z}(x)}{x}$, which explains the presence of additional factor $\exp(\frac{-2\pi ik}{M}))$ in Case 2. Thus, instead of performing M evaluations of $\mathcal{Z}(x)$ at M-roots of unity, we perform only $M_0 = \frac{M}{2}$ evaluations of $Y(u)$ at M_0-roots of unity; i.e. only half the number of evaluations of $\mathcal{Z}(x)$ are necessary to obtain the coefficients of $Y(x)$. But then, we immediately obtain the full polynomial $\mathcal{Z}(x)$, since $\mathcal{Z}(x) = x \cdot Y(x^2)$, and the coefficients of $\mathcal{Z}(x)$ of even parity are zero.

In the following, we will need the observation, that if the parity of base pair distance $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$ between $\mathcal{A}, \mathcal{B}$ is even, then

$$Y(x^2) = \mathcal{Z}(x) \tag{5.20}$$

while if the parity is odd, then

$$Y(x^2) = \frac{1}{x} \cdot \mathcal{Z}(x). \tag{5.21}$$

5.3.2 Optimization due to complex conjugates

As before, let M be the the least number evenly divisible by 4, which is greater than or equal to $(N+1)^2$, let $\nu = \exp(\frac{2\pi i}{M})$ and $\omega = \nu^2 = \exp(\frac{2\pi i}{M})^2 = \exp(\frac{2\pi i}{M/2})$. Clearly, ν is a principal complex M-root of unity, while ω is a principal complex $\frac{M}{2}$-root of unity. Evaluate $\mathcal{Z}$ for each $\frac{M}{2}$-root of unity that belongs to the first quadrant, and apply Lemma 5.3 to infer the values of $\mathcal{Z}$ for each $\frac{M}{2}$-root of unity that belongs to the fourth quadrant. More precisely, we compute $\mathcal{Z}(\nu^k)$, for $k = 0, \ldots, \frac{M}{4}$, and by Lemmas 5.3, 5.4, 5.5 infer that for $k = \frac{M}{4} + 1, \ldots, \frac{M}{2} - 1$, we have $\mathcal{Z}(\nu^k) = -1^{d_0} \cdot \overline{\mathcal{Z}(\nu^{\frac{M}{2}-k})}$, where $d_0 = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$. This is justified in the following.

By induction on $k = \frac{M}{4} + 1, \ldots, \frac{M}{2} - 1$, we have

$$
\begin{aligned}
Y(\omega^k) &= Y(\nu^{2k}) \\
&= \begin{cases} \mathcal{Z}(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 0 \bmod 2 \\ \frac{1}{\nu^k} \cdot Z(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \mathcal{Z}(-\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 0 \bmod 2 \\ \nu^{-k} \cdot \mathcal{Z}(-\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \mathcal{Z}(\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 0 \bmod 2 \\ \nu^{-k} \cdot \mathcal{Z}(\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \overline{\mathcal{Z}(\nu^{(\frac{M}{2}-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 0 \bmod 2 \\ -\nu^{-k} \cdot \overline{\mathcal{Z}(\nu^{(\frac{M}{2}-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B}) = 1 \bmod 2 \end{cases}
\end{aligned} \tag{5.22}
$$

Line 1 follows by definition, since $\omega = \nu^2$; line 2 follows by equations (5.20) and (5.21); line 3 follows by Lemma 5.5; line 4 follows by Lemma 5.4. Thus if $d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B})$ is even, then

$$
y_k = Y(\omega^k) = \begin{cases} \mathcal{Z}(\nu^k) & \text{for } k = 0, \ldots, \frac{M}{4} \\ \overline{\mathcal{Z}(\nu^{\frac{M}{2}-k})} & \text{for } k = \frac{M}{4}+1, \ldots, \frac{M}{2}-1 \end{cases} \tag{5.23}
$$

while if $d_{\mathrm{BP}}(\mathcal{A},\, \mathcal{B})$ is odd, then

$$
y_k = Y(\omega^k) = \begin{cases} \nu^{-k} \cdot \mathcal{Z}(\nu^k) & \text{for } k = 0, \ldots, \frac{M}{4} \\ -\nu^{-k} \cdot \overline{\mathcal{Z}(\nu^{\frac{M}{2}-k})} & \text{for } k = \frac{M}{4}+1, \ldots, \frac{M}{2}-1 \end{cases} \tag{5.24}
$$

It follows that values $y_0, \ldots, y_{\frac{M}{2}-1}$ can be obtained by only $\frac{M}{4}$ evaluations of $\mathcal{Z}(x)$.

Pseudocode for improved `FFTbor2D`

PURPOSE: Computes the m most significant digits of probabilities $p_{r\cdot(N+1)+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$
INPUT: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structures $\mathcal{A}$, $\mathcal{B}$ of $\mathbf{s}$, integer m
OUTPUT: Probabilities $p_{r\cdot(N+1)+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$ to m significant digits for $r, s = 0, \ldots, N$

---

1 **function** FFTBOR2D IMPROVED($\mathbf{s}$, $\mathcal{A}$, $\mathcal{B}$, m)
2 $n \leftarrow \mathit{length}(\mathbf{s})$
3 $N \leftarrow n + (n \bmod 2)$
4 $M \leftarrow (N+1)^2 + ((N+1)^2 \bmod 4)$
5 $M_0 \leftarrow \frac{M}{2}$
6 **for** $k \leftarrow 0, (N+1)^2 - 1$ **do** ▷ Note that $k \leftarrow \pi_1(k) \cdot M + \pi_2(k)$
7 $\pi_1(k) \leftarrow \lfloor \frac{k}{N+1} \rfloor$
8 $\pi_2(k) \leftarrow k \bmod (N+1)$
9 **end for**
10 **for** $k \leftarrow 0, M-1$ **do** ▷ Compute all complex M and M_0 roots of unity
11 $\nu_k \leftarrow \exp(\frac{2\pi i k}{M})$
12 **if** $k < M_0$ **then**
13 $\omega_k \leftarrow \exp(\frac{2\pi i k}{M_0})$
14 **end if**
15 **end for**
16 **for** $k \leftarrow 0, M_0 - 1$ **do**
17 **if** $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) \bmod 2 = 0$ **then** ▷ From equation (5.23)
18 **if** $k \leq \frac{M_0}{2}$ **then**
19 $y_k \leftarrow \mathcal{Z}(\nu^k)$
20 **else**
21 $y_k \leftarrow \overline{\mathcal{Z}(\nu^{M_0-k})}$
22 **end if**
23 **else** ▷ From equation (5.24)
24 **if** $k \leq \frac{M_0}{2}$ **then**
25 $y_k \leftarrow \nu^{-k} \cdot \mathcal{Z}(\nu^k)$
26 **else**
27 $y_k \leftarrow -\nu^{-k} \cdot \overline{\mathcal{Z}(\nu^{M_0-k})}$
28 **end if**
29 **end if**
30 **end for**

```
31    for k ← 0, M_0 − 1 do                          ▷ Note that Z(ν_0) = Z
32        y_k ← 2^m · y_k / Z(ν_0)
33    end for
34    for k ← 0, M_0 − 1 do                          ▷ Compute IDFT from equation (5.25)
35        a_k ← (1/M_0) Σ_{k=0}^{M_0−1} y_k ω^{−kj}
36    end for
37    for k ← 0, M − 1 do                            ▷ Change the polynomial back to degree M
38        if d_BP(A, B) mod 2 = 0 then
39            if k mod 2 = 0 then
40                p_{π_1(k)·n+π_2(k)} ← a_{k/2}
41            else
42                p_{π_1(k)·n+π_2(k)} ← 0
43            end if
44        else
45            if k mod 2 = 0 then
46                p_{π_1(k)·n+π_2(k)} ← 0
47            else
48                p_{π_1(k)·n+π_2(k)} ← a_{(k−1)/2}
49            end if
50        end if
51    end for
52    for k ← 0, (N + 1)^2 − 1 do                    ▷ Truncate to m significant digits
53        p_k ← 2^{−m} · ⌊p_k⌋
54    end for
55    return p_0, ..., p_{(N+1)^2−1}                 ▷ Return all p(r, s) = p_{r·(N+1)+s} = Z^{r,s}_{1,n} / Z
56 end function
```

FIGURE 5.2: Pseudocode to compute the m most significant digits for probabilities $p_{rn+s} = \mathbf{Z}^{r,s}_{1,n}/\mathbf{Z}$. In our implementation, due to numerical stability issues in the FFT engine, precision parameter m has an upper bound of 8 – only the $m = 8$ most significant digits are computed with `FFTbor2D`. (Note that the software actually usesbase 2 precision parameter, with maximum of 27, where $2^{27} \approx 10^8$) It is well-known that the FFT requires $O(N \log N)$ time to solve the inverse discrete Fourier transform for a polynomial of degree N. In our case, $N = n^2$, and so the FFT requires time $O(n^2 \log n)$.

5.3.3 Polynomial interpolation to evaluate $\mathcal{Z}_{i,j}(x)$

Now let $M_0 = \frac{M}{2}$, let $\nu = \exp(\frac{2\pi i}{M})$ be the principal M-root of unity, and $\omega = \nu^2 = \exp(\frac{2\pi i}{M/2}) = \exp(\frac{2\pi \cdot 2i}{M})$ be the principal M_0-root of unity. Recall that the Vandermonde matrix V_{M_0} is defined to be the $M_0 \times M_0$ matrix, whose i,j entry is $\omega^{i \cdot j} = \nu^{2i \cdot j}$; i.e.

$$V_{M_0} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{M_0-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(M_0-1)} \\ 1 & \omega^3 & \omega^6 & \dots & \omega^{3(M_0-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{M_0-1} & \omega^{2(M_0-1)} & \dots & \omega^{(M_0-1)(M_0-1)} \end{pmatrix}$$

The Fast Fourier Transform (FFT) is the $O(n \log n)$ algorithm, which computes the Discrete Fourier Transform (DFT), defined as the matrix product $\mathbf{Y} = V_{M_0}\mathbf{A}$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{M_0-1} \end{pmatrix} = V_{M_0} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{M_0-1} \end{pmatrix}$$

The (i,j) entry of $V_{M_0}^{-1}$ is $\frac{\omega^{-ji}}{M_0}$ and that

$$a_j = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \omega^{-kj} = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \nu^{-2kj} \tag{5.25}$$

for $j = 0, \dots, M_0 - 1$ (for more on FFT, see **?**).

Since we defined $\mathbf{Y}$ in (**??**) by $\mathbf{Y} = (y_0, \dots, y_{M_0-1})^T$, where $y_0 = \mathcal{Z}(\alpha_0), \dots, y_{M_0-1} = \mathcal{Z}(\alpha_{M_0-1})$ and $\alpha_k = \omega^k \exp(\frac{k \cdot 2\pi i}{M_0})$, it follows that the coefficients $z_k = \mathbf{Z}_{1,n}^{\pi_1(k),\pi_2(k)}$ in the polynomial $\mathcal{Z}(x) = z_0 + z_1 x + \cdots + z_M x^M$ defined in (**??**) can be computed, at least in principle, by using the FFT. However, since the values of z_k are astronomically large, numerical instability makes even this approach infeasible for moderate values of n. Nevertheless, we apply this approach to compute the m most significant digits of $\frac{\mathbf{Z}_{1,n}^{\pi_1(k),\pi_2(k)}}{\mathbf{Z}}$, where the partition function $\mathbf{Z} = \sum_S \exp(-E(\mathcal{S})/RT)$ satisfies $\mathbf{Z} = \sum_{x,y} \mathbf{Z}_{1,n}^{x,y}$. This leads to numerical stability, allowing `FFTbor2D` to compute the m most significant digits of $p(x,y) = \frac{\mathbf{Z}_{1,n}^{x,y}}{\mathbf{Z}}$. Pseudocode for the complete algorithm, `FFTbor`, is given in Figure 2.

5.4 Benchmarking and performance considerations

5.5 Applications of the `FFTbor2D` algorithm

Chapter 6

Hermes

6.1 Introduction

In this chapter, we present the `Hermes` software suite—a collection of programs aimed at evaluating the kinetic properties of RNA molecules. Provided a coarse-grained energy landscape generated by `FFTbor2D` (described in Chapter 5), we present software which computes both the mean first passage time and equilibrium time for this discretized energy landscape. We also provide software which computes the exact kinetics for an RNA molecule, however since this requires exhaustive enumeration of all secondary structures—which is known to be an exponential quantity for the length of the RNA in consideration—the full kinetics are not expected to be practical for anything beyond a sequence of trivial length. The software in `Hermes` presents a practical application of the energy landscapes computed by the `FFTbor2D` algorithm. Contrasted against the other kinetics software in the field, `Hermes` offers similar accuracy with unparalleled performance which opens up the possibility for large-scale kinetic analysis *in silico*, which we expect to be of use for synthetic design.

6.1.1 Organization

This chapter is organized in the following fashion. We begin by providing background on the state-of-the-art approaches for kinetic analysis of RNAs. From there, we move into a technical discussion of two traditional approaches for kinetics, computation of the mean first passage time and the equilibrium time. With this foundation in place, we proceed

to discuss the high-level organization of the `Hermes` software package, and describe in detail each of the four underlying programs which comprise the kinetics suite. We then move on to present comparitive benchmarking of `Hermes` against other methods, before finally concluding with some remarks on the accuracy and applicability of `Hermes` to computational RNA design.

6.2 Background

6.3 Traditional approaches for kinetics

6.3.1 Mean first passage time

6.3.2 Equilibrium time

6.4 Software within the `Hermes` suite

The `Hermes` software package was developed on the Macintosh OS X operating system (10.9.2 and 10.10) and should work with any Unix-like platform (Ubuntu, Debian, and CentOS were tested). We make the source code freely available under the MIT License in two locations. Our lab hosts the latest stable version of the code at `http://bioinformatics.bc.edu/clotelab/Hermes` and a fully version-controlled copy at `https://github.com/evansenter/hermes`. The data and figures presented in this article were generated with the source code hosted at the first URL, and we make no guarantee as to the stability of development branches in our Git repository.

External dependencies for the software include a C (resp. C++) compiler supporting the GNU99 language specification (resp. C++98), FFTW implementation of Fast Fourier Transform `?` ($\geq$ 3.3.4) `http://www.fftw.org/`, Gnu Scientific Library GSL ($\geq$ 1.15) `http://www.gnu.org/software/gsl/`, Vienna RNA Package `?` ($\geq$ 2.0.7) `http://www.viennarna.at`, and any corresponding sub-packages included with the aforementioned software. For a more detailed explanation of both external dependencies and installation instructions, refer to the 'DOCS.pdf' file at the web site outlining the configuration and compilation process for the `Hermes` suite.

`Hermes` is organized into three independent directories: (1) `FFTbor2D`, (2) `RNAmfpt`, and (3) `RNAeq` (see Figure **??**). These packages compile into both standalone executables and archive files. The archives provide a DRY API which allow the development of novel applications using source from across the `Hermes` package without having to copy-and-paste relevant functions. We provide two such examples of this in the `ext` subdirectory: `FFTmfpt` and `FFTeq`. These applications are simple C drivers that use functions from `FFTbor2D`, `RNAmfpt` and `RNAeq` to replicate a pipeline of executable calls without having to deal with intermediary data transformation, I/O between calls or slow-down due to a scripting language driver such as Python or R.

6.4.1 Exact mean first passage time with `RNAmfpt`

`RNAmfpt` computes the *mean first passage time* (MFPT), sometimes referred to as the *hitting time* of a Markov chain, by using matrix inversion **?** – see Section **??**. The program takes as input a comma separated value (CSV) file containing the non-zero positions and values of a 2D probability grid; i.e. a CSV format file having columns i, j, and p. The first two columns, i and j correspond to the *0-indexed* row-ordered position in the rate matrix and the final column p is the probability $p_{i,j}$ of a transition from i to j. From this input, the mean first passage time is constructed by matrix inversion. Because this program was designed with the original intent of handling 2D-probability grids, all vertexes are uniquely identified by index tuples (which conceptually correspond to positioning in a 2D array). However, it is trivial to use this code with both 1D-probability grids such as those produced by `FFTbor ?` or arbitrary transition matrices without any change to the underlying implementation. The software additionally provides many options for defining the structure of the graph underlying the Markov chain. Some of these include the option to force a fully connected graph (useful in cases where there is no non-zero path between the start / end state) or to enforce detailed balance. Finally, `RNAmfpt` also accepts as input the probability transition matrix, a stochastic matrix with row sums equal to 1, and computes the mean first passage time for the corresponding Markov chain.

6.4.2 Approximate mean first passage time with `FFTmfpt`

`FFTmfpt` approximates the mean first passage time of a given RNA sequence folding from input structure A to B, by *exactly* computing the mean first passage time from state $(0, d_0)$ to state $(d_0, 0)$ in the 2D probability grid obtained from running `FFTbor2D`.

Here, d_0 is the base pair distance between structures A, B, and the MFPT is computed for the Markov chain, whose states are the non-empty 2D probability grid points, and whose transition probabilities are defined by $p_{(x,y),(x',y')} = \frac{P(x',y')}{P(x,y)}$. As we report in this paper, given an RNA sequence **s**, if A is the empty structure and B the MFE structure of **s**, then `FFTmfpt` output is well correlated with the exact MFPT in folding the empty structure to the MFE structure, where transitions between structures involve the addition or removal of a single base pair.

6.4.3 Exact equilibrium time with `RNAeq`

`RNAeq` computes the population proportion of a user-provided structure over arbitrary time units. Like `RNAmfpt`, this program takes as input a comma separated value (CSV) file containing the non-zero positions and values of a 2D probability grid. From this input a rate matrix is constructed for the underlying Markov process. Alternatively, `RNAeq` can accept as input an arbitrary rate matrix. Performing spectral decomposition of the column-ordered rate matrix that underlies the corresponding Markov process, `RNAeq` computes either the equilibrium time or population occupancy frequencies. Additionally, `RNAeq` can call the Vienna RNA Package program `RNAsubopt ?`, with a user-specified upper bound to the energy difference with the minimum free energy. With this option, the rate matrix is constructed for the Markov process, whose states consist of all the structures returned by `RNAeq`, and the equilibrium time or population occupancy frequencies are computed. Due to the time and memory required for this option, we do not expect it to be used except for small sequences.

6.4.4 Approximate equilibrium time with `FFTeq`

`FFTeq` allows an investigator to efficiently estimate population kinetics for a sequence folding between two arbitrary, but fixed, structures. The transition rate matrix underlying the Markov process necessary for eigendecomposition is derived from the 2D-energy landscape. Vertices in the rate matrix represent a subset of structures compatible with the input sequence as modeled by `FFTbor2D`, which makes the graph size more tractable than structural sampling with `RNAsubopt`, even with constraints.

6.4.4.1 Population occupancy curves with `FFTeq`

6.4.4.2 Approximating equilibrium time from occupancy curves

The computation of an equilibrium time value from the eigendecomposition of the rate matrixis a rather thorny issue. (something about the non-linear solver approach not working, same as in FFTbor w/ numeric instability)

6.5 Correlations of kinetics data across software

6.5.0.1 Benchmarking data for computational comparison

In this section, we describe a benchmarking set of 1,000 small RNAs used to benchmark the previously described kinetics methods in a comparative study. To ensure that mean first passage time can be computed from $(I - P_{x_\infty}^{-})^{-1} \cdot \mathbf{e}$ by using matrix inversion, that spectral decomposition of the rate matrix is possible, and to ensure that `Kinfold` simulations would provide sufficient statistics, we generated a collection of 1,000 random RNA sequences of length 20 nt, each having expected compositional frequency of 1/4 for A,C,G,U, and each having at most 2,500 distinct secondary structures, such that the minimum free energy is less than or equal to −5.5 kcal/mol.

For example, one of the 1,000 sequences is ACGCGACGUGCACCGCACGU with minimum free energy structure `.....((((((...))))))` having free energy of −6.4 kcal/mol. Statistics for the free energies of the 2,453 secondary structures of this 20-mer are the following: mean is 10.695, standary deviation is 4.804, maximum is 25.00, minimum is −6.40. A histogram for the free energy of all secondary structures of ACGCGACGUGCACCGCACGU is depicted in the left panel of Figure 6.1. The right panel of the same figure depicts the minimum free energy structure of the 54 nt hammerhead type III ribozyme from Peach Latent Mosaic Viroid (PLMVd), discussed later. This secondary structure is identical to the consensus structure from Rfam 11.0 **?**.

Figure 6.2 displays the mean and standard deviation for `Kinfold` simulations of folding time for each of the 1,000 RNA sequences from our benchmarking data. For each sequence, the mean and standard deviation of the time required to fold the empty structure to the MFE structure were computed from 10,000 `Kinfold` runs, each run with an upper bound of 10^8 Monte Carlo steps, thus ensuring that all simulations converged.

FIGURE 6.1: *(Left)* Histogram of free energies of secondary structures of ACGCGACGUGCACCGCACGU, which range from −6.5 to +25 kcal/mol, with mean of 10.695 kcal/mol. *(Right)* Minimum free energy structure of the 54 nt Peach Latent Mosaic Viroid (PLMVd) AJ005312.1/282-335, which is identical to the consensus structure from Rfam 11.0 **?**. `RNAfold` from Vienna RNA Package 2.1.7 with energy parameters from the Turner 1999 model were used, since the minimum free energy structure determined by the more recent Turner 2004 energy parameters does *not* agree with the Rfam consensus structure – see **?**. Positional entropy, a measure of divergence in the base pairing status at each positions for the low energy ensemble of structures, is indicated by color, using the RNA Vienna Package utility script `relplot.pl`.

FIGURE 6.2: *(Left)* Histogram of `Kinfold` folding times for 20-mer CCGAUUGGCG AAAGGCCACC. The mean [resp. standard deviation] of 10,000 runs of `Kinfold` for this 20-mer is 538.37 [resp. 755.65]. Note the close fit to the exponential distribution, *(Right)* Mean minus standard deviation ($\mu - \sigma$), mean (μ), and mean plus standard deviation ($\mu + \sigma$) of the logarithm of `Kinfold` folding times, taken over 10,000 runs for each of the 1,000 sequences from the benchmarking set of 20-mers. For graphical illustration, we have sorted the log folding times in increasing order.

The sequences were then sorted by increasing folding time mean. Standard deviation exceeded the mean in 83.9% of the 1,000 cases, indicating the enormous variation between separate `Kinfold` runs, even for 20 nt RNA sequences having at most 2,500 secondary structures. In our opinion, `Kinfold` is an expertly crafted implementation of Gillespie's algorithm for an event driven Monte Carlo simulation of one-step RNA secondary structure folding. From the standpoint of biophysics and physical chemistry, there is no more reliable simulation method, except of course the exact computation of mean first passage time using linear algebra. Nevertheless, the enormous time required for reliable `Kinfold` estimations and the large standard deviations observed point out the need for a faster method to approximate folding time.

6.5.0.2 Pearson correlation coefficients for various kinetics packages

In this section, we display the correlation between (1) the *gold standard* method `MFPT`, both with and without the Hastings modification using equations (**??**) and (**??**), (2) the *platinum standard* method `Equilibrium`, (3) the *silver standard* method `Kinfold`, (4) `FFTmfpt` with and without the Hastings modification using equations (**??**) and (**??**), (5)

FFTeq which computes equilibrium time for the 2D-grid, (6) RNA2Dfold with and without the Hastings modification using equations (??) and (??). Correlations with [resp. without] the Hastings modification are summarized in the lower [resp. upper] triangular portion of Table 6.1. It is clear that correlations between the mathematically exact methods MFPT, Equilibrium, and approximation methods Kinfold, FFTmfpt, FFTeq, RNA2Dfold are improved when using the Hastings correction.

Figures 6.3, 6.4, 6.5 depict scatterplots for kinetics obtained by some of the algorithms above. The left panel of Figure 6.3 shows a scatter plots for gold standard MFPT versus platinum standard Equilibrium, with correlation value 0.5652. The right panel of the same figure shows a scatter plot for Kinfold versus Equilibrium, with correlation 0.7814. Note the persence of two clusters in this and some of the other scatter plots. Cluster A consists of RNA sequences whose folding time, as determined by MFPT or Equilibrium, is rapid – specifically, the natural logarithm of the MFPT is at most 7.5. Cluster B consists of the remaining RNA sequences, whose folding time is longer than that of cluster A. There are no significant differences between RNA sequences in clusters A and B with respect to GC-content, sequence logo, minimum free energy, number of secondary structures, etc. The left panel of Figure 6.4 shows the scatter plot for MFPT versus Kinfold, with correlation 0.7933, and the right panel shows the scatter plot for MFPT versus FFTmfpt, with correlation 0.6035. Figure 6.5 shows scatter plots for FFTmfpt versus Kinfold (left) and for FFTmfpt versus FFTeq (right), with respective correlation values 0.7608 and 0.9589. Kinfold obviously provides a better correlation with the exact value of mean first passage time; however, since the standard deviation of Kinfold runs is as large as the mean,[1] accurate kinetics estimates from Kinfold require prohibitively large computational time – indeed, in ? reliable kinetics for phe-tRNA from yeast were obtained by 9,000 Kinfold simulations, each for 10^8 steps, requiring 3 months of CPU time on an Intel Pentium 4 running at 2.4 GHz under Linux. Although the correlation value of 0.6035 between MFPT and FFTmfpt is much less than that obtained by Kinfold, the runtime required by our method FFTmfpt is measured in seconds, even for moderate to large RNAs. For this reason, we advocate the use of FFTmfpt in synthetic biology screens to design RNA sequences having certain desired kinetic properties. Once promising candidates are found, it is possible to devote additional computational time to Kinfold simulations for more accurate kinetics.

[1] It follows from spectral decomposition that equilibrium time follows an exponential distribution (or sum of exponential distributions). Exponential distributions have the property that the mean is equal to the standard deviation, hence it is not surprising that Kinfold kinetics have this property.

FIGURE 6.3: Scatter plots of the natural logarithm of times from `MFPT` versus `Equilibrium` (left) and for `Kinfold` versus `Equilibrium` (right).

FIGURE 6.4: Scatter plots of the natural logarithm of times from `MFPT` versus `Kinfold` (left) and for `MFPT` versus `FFTmfpt` (right).

FIGURE 6.5: Scatter plots of the natural logarithm of times from `Kinfold` versus `FFTmfpt` (left) and for `FFTmfpt` versus `FFTeq` (right).

Hastings (Yes\No)	MFPT	Equilibrium	Kinfold	FFTmfpt	RNA2Dfold	FFTbor	BarrierBasins	FFTeq
MFPT	1	0.5683	0.7945	0.5060	0.5110	0.5204	0.5280	0.4472
Equilibrium	0.5798	1	0.7814	0.7043	0.7025	0.5080	0.5979	0.6820
Kinfold	0.7933	0.7507	1	0.7312	0.7358	0.6241	0.6328	0.6445
FFTmfpt	0.6035	0.7935	0.7608	1	0.9980	0.5485	0.8614	0.9589
RNA2Dfold	0.6076	0.7919	0.7655	0.9983	1	0.5584	0.8538	0.9515
FFTbor	0.5416	0.5218	0.6241	0.5748	0.5855	1	0.3450	0.4229
BarrierBasins	0.6346	0.6578	0.6328	0.8310	0.8217	0.3450	1	0.9149
FFTeq	0.5614	0.7916	0.6966	0.9670	0.9590	0.4757	0.8940	1

TABLE 6.1: Table of Pearson correlation coefficients for various methods to compute or approximate RNA secondary structure folding kinetics. Lower [resp. upper] triangular entries are with [resp. without] the Hastings correction for Markov chain probability matrices. The methods are: `MFPT` (mean first passage time, computed by matrix inversion for the Markov chain consisting of all secondary structures, with move allowed between structures differing by one base pair), `Equilibrium` (equilibrium time, computed by spectral decomposition of a rate matrix comprising all secondary structures to compute population fraction $P(t)$ at time t), `Kinfold` (an implementation of Gillespie's Algorithm to approximate refolding pathways using an event-based Monte Carlo simulation), `FFTmfpt` (mean first passage time for Markov chain consisting of "grid point" states (x, y) with probability $P(x, y) = \sum_S exp(-E(S)/RT)/Z$, computed by `FFTbor2D`, where the sum is taken over structures having base pair distance x to the empty structure and y to the MFE structure, `RNA2Dfold` (mean first passage time, computed as previously explained, but using `RNA2Dfold` in place of `FFTbor2D` to compute $P(x, y)$), `FFTbor` (mean first passage time, computed for the Markov chain consisting of states $0, 1, \ldots, n$, for which $P(x) = \sum_S \exp(-E(S)/RT)/Z$, where the sum is taken over all secondary structures whose base pair distance is x from the MFE structure), `BarrierBasins` (equilibrium time, computed using spectral decomposition on the Markov process consisting of "grid point" states output from `Barriers`), and `FFTeq` (equilibrium time, computed in the same fashion as `BarrierBasins` using a Markov process derived from the energy landscape output by `FFTbor2D`).

Chapter 7

Discussion

Bibliography