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

Tuesday 28th 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: _____

Acknowledgements

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Organization

Chapter 2

Ribofinder

2.1 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 2.2. 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.

2.1.1 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.

2.2 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 a downstream 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.

2.3 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.

2.3.1 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.

2.3.1.1 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 2.3.2.1). 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).

2.3.1.2 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.

2.3.2 Step 2: Structural prediction

2.3.2.1 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:

$$S \rightarrow S\,. \mid .\,S \mid (S) \mid SS \mid \epsilon \tag{2.1}$$

The grammar from (2.1) 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 $\mathtt{p}_1(l-1) = \text{C}$. 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.

2.3.2.2 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:

<table>
<tr><th>G-box gene-off constraint mask</th><th>G-box gene-on constraint mask</th></tr>
<tr><td colspan="2">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.</td></tr>
<tr><td>Force basepairs and unpaired regions in $\mathtt{t}_2$.</td><td>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 $\mathtt{p}_2(j)$.</td></tr>
</table>

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.

2.3.3 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.

2.4 Using `Ribofinder` against the RefSeq database

2.5 Extending beyond guanine riboswitches

Chapter 3

FFTbor

3.1 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.

3.1.1 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 an application of `FFTbor`, in the area of RNA folding kinetics.

3.2 Background

3.3 Formalization of the problem

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

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}, \tag{3.1}$$

where p_k is defined as $p_k = \frac{\mathbf{Z}^k}{\mathbf{Z}}$. 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

(3.1). 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

$$\mathbb{B} = \{(\mathrm{A}, \mathrm{U}), (\mathrm{U}, \mathrm{A}), (\mathrm{G}, \mathrm{C}), (\mathrm{C}, \mathrm{G}), (\mathrm{G}, \mathrm{U}), (\mathrm{U}, \mathrm{G})\}. \tag{3.2}$$

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 $\mathcal{S}, \mathcal{T}$:

$$d_{\mathrm{BP}}(\mathcal{S}, \mathcal{T}) = |\mathcal{S} \cup \mathcal{T}| - |\mathcal{S} \cap \mathcal{T}|. \tag{3.3}$$

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

$$d_{\mathrm{BP}}(\mathcal{S}_{[i,j]}, \mathcal{T}_{[i,j]}) = |\{(x, y) : i \leq x < y \leq j, (x, y) \in \mathcal{S} - \mathcal{T} \mathrm{or} (x, y) \in \mathcal{T} - \mathcal{S}\}| = k. \tag{3.4}$$

3.4 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.

$$\mathbf{Z}^k = \mathbf{Z}^k_{1,n} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{S}^*)=k}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{3.5}$$

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

$$\mathbf{Z} = \mathbf{Z}_{1,n} = \sum_{k=0}^{n} \mathbf{Z}^k_{1,n} \tag{3.6}$$

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

$$d_{\mathrm{BP}}(\mathcal{S}^*, \mathcal{S}) \leq |\mathcal{S}^*| + \lfloor \frac{n-\theta}{2} \rfloor \leq n. \tag{3.7}$$

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

$$p(k) = \frac{\mathbf{Z}^k_{1,n}}{\mathbf{Z}_{1,n}}. \tag{3.8}$$

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.

3.4.1 Definition of the partition function $\mathbf{Z}^k_{1,n}$

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. Freyhult et al. [2005] to determine the partition function $\mathbf{Z}^k_{i,j}$ with respect to the Nussinov-Jacobson energy E_0 model Nussinov and Jacobson [1980], 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` Freyhult et al. [2005] 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 $\mathbf{Z}^k_{i,j}$ is given by

$$\mathbf{Z}^0_{i,j} = 1, \text{ for } i \leq j, \tag{3.9}$$

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

$$\mathbf{Z}^k_{i,j} = 0, \text{ for } k > 0, i \leq j \leq i + \theta, \tag{3.10}$$

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 $\mathbf{Z}^k_{i,j}$ for $k > 0$ and $j > i + \theta$ is

$$\mathbf{Z}_{i,j}^{k} = \mathbf{Z}_{i,j-1}^{k-b_0} \; + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\le r<j}} \; \sum_{w+w'=k-b(r)} \exp(-E_0(r,j)/RT)\cdot \mathbf{Z}_{i,r-1}^{w}\mathbf{Z}_{r+1,j-1}^{w'}, \tag{3.11}$$

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 $i \le r < j$. 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*

$$\mathcal{Z}(x) = \sum_{k=0}^{n} c_k x^k \tag{3.12}$$

where coefficients $c_k = \mathbf{Z}_{1,n}^{k}$. Moreover, because of equation (3.7) 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

$$\mathcal{Z}(a_1) = y_1, \ldots, \mathcal{Z}(a_n) = y_n, \tag{3.13}$$

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

$$P_k(x) = \frac{\prod_{i \neq k}(x - x_i)}{\prod_{i \neq k}(x_k - x_i)}. \tag{3.14}$$

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 Higham [2004], 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 (3.1), rather than the polynomial $\mathcal{Z}(x)$, defined in equation (3.12), and performing interpolation with the Fast Fourier Transform (FFT) Cormen et al. [1990] where $\alpha_0, \ldots, \alpha_{n-1}$ are chosen to be complex nth roots of unity,

$\alpha_k = e^{2\pi ik/n}$. 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 (3.14) or by Gaussian elimination. Fewer numerical operations implies increased numerical stability in our application.

3.4.2 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

$$\mathbf{Z}_{1,n}^k = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{S}^*)=k}} e^{\frac{-E_0(\mathcal{S})}{RT}} \tag{3.15}$$

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

$$\mathbf{Z}_{i,j}^k = \mathbf{Z}_{i,j-1}^{k-b_0} + \sum_{\substack{s_r s_j \in \mathbb{B}, \\ i \le r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \sum_{w+w'=k-b(r)} \mathbf{Z}_{i,r-1}^{w} \mathbf{Z}_{r+1,j-1}^{w'} \right) \tag{3.16}$$

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

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k(i,j)\, x^k \tag{3.17}$$

where

$$c_k(i,j) \;=\; \mathbf{Z}_{i,j}^{k} \;=\; \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\,\mathcal{S}^*_{[i,j]})=k}} e^{\frac{-E_0(\mathcal{S})}{RT}}. \tag{3.18}$$

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

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k\, x^k \tag{3.19}$$

where

$$c_k(i,j) = \mathbf{Z}_{i,j}^{k}. \tag{3.20}$$

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

$$\mathcal{Z}_{i,j}(x) = \mathcal{Z}_{i,j-1}(x) \cdot x^{b_0} + \sum_{\substack{s_r s_j \in \mathbb{B}, \\ i \le r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(x) \cdot \mathcal{Z}_{r+1,j-1}(x) \cdot x^{b(r)} \right). \tag{3.21}$$

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 (3.1), $[x^k]p(x) = p_k$, and in equation (3.12), $[x^k]\mathcal{Z}(x) = c_k(i,j)$.

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

$$\begin{aligned}
[x^k]\mathcal{Z}_{i,j}(x) = c_k(i,j) &= \mathbf{Z}^k_{i,j} \\
&= \mathbf{Z}^{k-b_0}_{i,j-1} + \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \mathbf{Z}^{k_0}_{i,r-1} \cdot \mathbf{Z}^{k_1}_{r+1,j-1} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \left\{[x^{k_0}]\mathcal{Z}_{i,r-1}(x)\right\} \cdot \left\{[x^{k_1}]\mathcal{Z}_{r-1,j-1}(x)\right\} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot [x^{k_0+k_1}]\mathcal{Z}_{i,r-1}(x)\mathcal{Z}_{r-1,j-1}(x).
\end{aligned} \tag{3.22}$$

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:

$$\mathcal{Z}_{i,j}(\alpha) = \mathcal{Z}_{i,j-1}(\alpha) \cdot \alpha^{b_0} + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\leq r<j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(\alpha) \cdot \mathcal{Z}_{r+1,j-1}(\alpha) \cdot \alpha^{b(r)} \right). \quad (3.23)$$

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

$$\mathbf{Y} = (y_0, \ldots, y_{n-1})^{\mathrm{T}} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} \quad (3.24)$$

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 $O(n^2)$.

3.4.3 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.

$$V_n = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \dots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{pmatrix} \tag{3.25}$$

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 $\mathbf{Y} = V_n \mathbf{A}$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = V_n \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} \tag{3.26}$$

On page 837 of Cormen et al. [1990], it is shown that the (i, j) entry of V_n^{-1} is $\frac{\omega^{-ji}}{n}$ and that

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \, \omega^{-kj} \tag{3.27}$$

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

Since we defined $\mathbf{Y}$ in equation (3.24) by $\mathbf{Y} = (y_0, \dots, y_{n-1})^{\mathrm{T}}$, where $y_0 = \mathcal{Z}(\alpha_0), \dots, y_{n-1} = \mathcal{Z}(\alpha_{n-1}))$ 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 (3.12) 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 $y_1 = \frac{\mathcal{Z}(\alpha_1)}{\mathbf{Z}}, \ldots, y_n = \frac{\mathcal{Z}(\alpha_n)}{\mathbf{Z}}$, and $\mathbf{Z}$ is the partition function defined in equation (3.6). Using the Fast Fourier Transform to compute the inverse Discrete Fourier Transform, it follows from equation (3.27) 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 (3.1). 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.

3.5 Benchmarking and performance considerations

In this subsection, we show that we need only evaluate the polynomial $\mathcal{Z}(x)$, as defined in equation (3.12), 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 $\overline{z} = a - bi$.

Lemma 3.2. *If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (3.12), 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*

$$\mathcal{Z}(a - bi) = A - Bi. \tag{3.28}$$

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(s, S*, m)
2      n ← length(s)
3      for k ← 0, n − 1 do                    ▷ Compute all complex nth roots of unity
4          ω_k ← exp(2πik/n)
5      end for
6      for k ← 0, n − 1 do                    ▷ Note that Z(ω_0) = Z
7          y_k ← 10^m · Z(ω_k)/Z(ω_0)
8      end for
9      for k ← 0, n − 1 do                    ▷ Compute IDFT from equation (3.27)
10         a_k ← (1/n) Σ_{j=0}^{n−1} y_j ω^{−kj}
11         p_k ← 10^{−m} · ⌊a_k⌋               ▷ Truncate to m significant digits
12     end for
13     return p_0, ..., p_{n−1}               ▷ Return all p_k for 0 ≤ k < n, from equation (3.8)
14 end function
```

FIGURE 3.1: The function FFTBOR computes the m most significant digits of $p_0, \ldots, p_{n-1}$, where $p_k = \frac{\mathbf{Z}^k}{\mathbf{Z}}$. This algorithm operates in $O(n^4)$ time and $O(n^2)$ space, a significant improvement over its predecessor `RNAbor`.

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.

$$
\begin{aligned}
(a+bi)^m &= \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (bi)^k \\
(bi)^k &= \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ -ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases}
\end{aligned}
\tag{3.29}
$$

$$
\begin{aligned}
(a-bi)^m &= \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (-bi)^k \\
(-bi)^k &= \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ -ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases}
\end{aligned}
\tag{3.30}
$$

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 3.2 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`.

3.6 Coarse-grained kinetics with **FFTbor**

The output of `FFTbor`, as shown in Figure 3.3, 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 Wolynes [2005]. A hypothesis behind theoretical work in biomolecular folding theory in Bryngelson et al. [1995] 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 3.3.

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 3.2. Despite the fact that there is no sequence homology according to pairwise BLAST Altschul et al. [1990], 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 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.

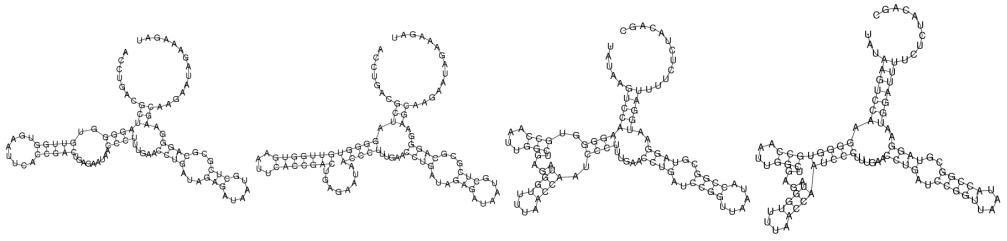

FIGURE 3.2: Rfam consensus structures (Rfam) and minimum free energy (MFE) secondary structures for two thiamine pyrophosphate (TPP) riboswitch aptamers, chosen at random from RF00059 Rfam family seed alignment Gardner et al. [2011]. Using pairwise BLAST Altschul et al. [1990], there is no sequence similarity, 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.

To test the hypothesis that folding is slower for rugged energy landscapes, we ran the kinetic folding software, `Kinfold` Flamm et al. [2000], 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 Flamm et al. [2000]), 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 $500,000$

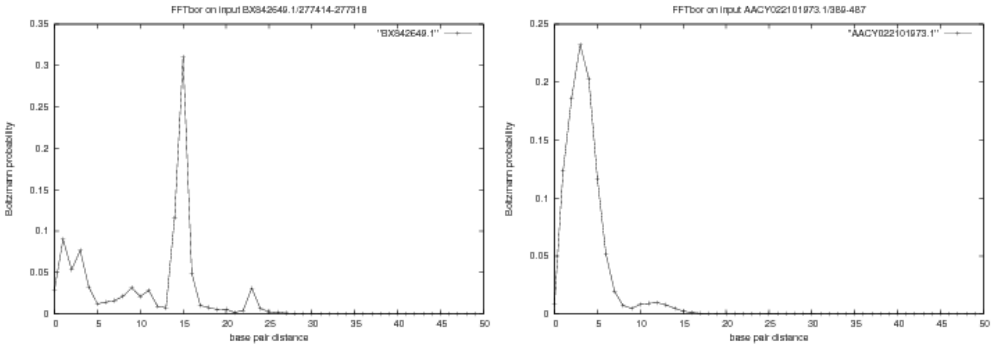

FIGURE 3.3: Output from `FFTbor` on two randomly selected thiamine pyrophosphate riboswitch (TPP) aptamers, taken from the Rfam database Gardner et al. [2011]. The x-axis represents base pair distance from the minimum free energy structure for each given sequence; the y-axis represents Boltzmann probabilities $p(k) = \mathbf{z}^k/\mathbf{z}$, 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` Flamm et al. [2000] simulations clearly show that the second sequence folds at least four times more quickly than the first sequence. See section 3.6 for details.

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 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

$$E[\{d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) : \mathcal{S} \in \mathbb{S}(s_1, \ldots, s_n)\}] = \sum_{\mathcal{S}} P(\mathcal{S}) \cdot d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) \tag{3.31}$$

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 Gardner et al. [2011]. 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	σ/μ	n	MFE	$\log_{10}$(MFPT)
μ	1					
σ	-0.4372	1				
σ/μ	-0.6914	0.9437	1			
n	0.7077	-0.1590	-0.3646	1		
MFE	-0.5695	0.7395	0.7596	-0.3685	1	
$\log_{10}$(MFPT)	-0.0363	0.4844	0.3762	0.4059	0.3990	1

TABLE 3.1: Pearson correlation between various aspects of selenocysteine insertion sequences from the seed alignment of Rfam family RF00031 Gardner et al. [2011]. 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.

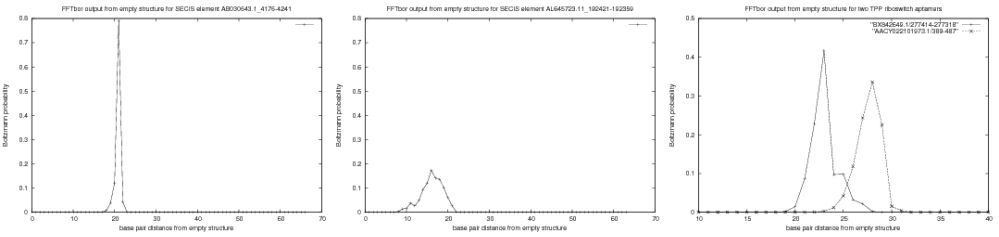

FIGURE 3.4: This figure represents the graphical output of `FFTbor`, when the empty structure is chosen as initial structure $\mathcal{S}^*$. The x-axis represents the number of base pairs per structure, taken over the ensemble of all secondary structures for the given RNA sequence; the y-axis represents 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 3.3. These anecdotal results bear up the correlation between standard deviation σ and $\log_{10}$(MFPT) described in 3.1.

In the right panel of Figure 3.4, 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 Altschul and Erikson [1985], 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 4 and 5, where a more detailed explanation of kinetics is provided, and additional evidence is provided to support this claim.

3.7 Performance characteristics of `FFTbor` and `RNAbor`

As visible from the defining recursions, the algorithmic time complexity of `RNAbor` is $O(n^5)$ and space complexity is $O(n^3)$, where n is the length of input RNA sequence. In contrast, the time complexity of `FFTbor` is $O(n^4)$ and space complexity is $O(n^2)$. Figure 3.5 displays run time curves for both `RNAbor` and `FFTbor`, when the initial structure $\mathcal{S}^*$ is taken to be either the empty structure or the minimum free energy (MFE) structure.

Here, we compare the run time of `RNAbor` Freyhult et al. [2007] and the (unparallelized version of) `FFTbor`, using a Dell Power Edge 1950, 2 x Intel Xeon E5430 Quad core with 2.80 GHz and 16 GB RAM. For $n = 20, 40, 60, \ldots, 300$, in step size of 20 nt, we generated n random RNA sequences of length n with equal probability for each nucleotide A,C,G,U (i.e. a 0th order Markov chain). For values of $n \leq 200$, 100 random sequences of length n were generated, while for values of $220 \leq n \leq 300$, only 10 sequences of length n were generated. RNA sequences larger than 300 nt were not tested, due to $O(n^3)$ memory constraints required by `RNAbor`. For each RNA sequence, `RNAbor` and `FFTbor` were both run, each starting with empty initial structure $\mathcal{S}^*$, and also with initial sequence $\mathcal{S}^*$ taken to be the MFE structure. Each data point in the table comprises the average run time for three independent evaluations.

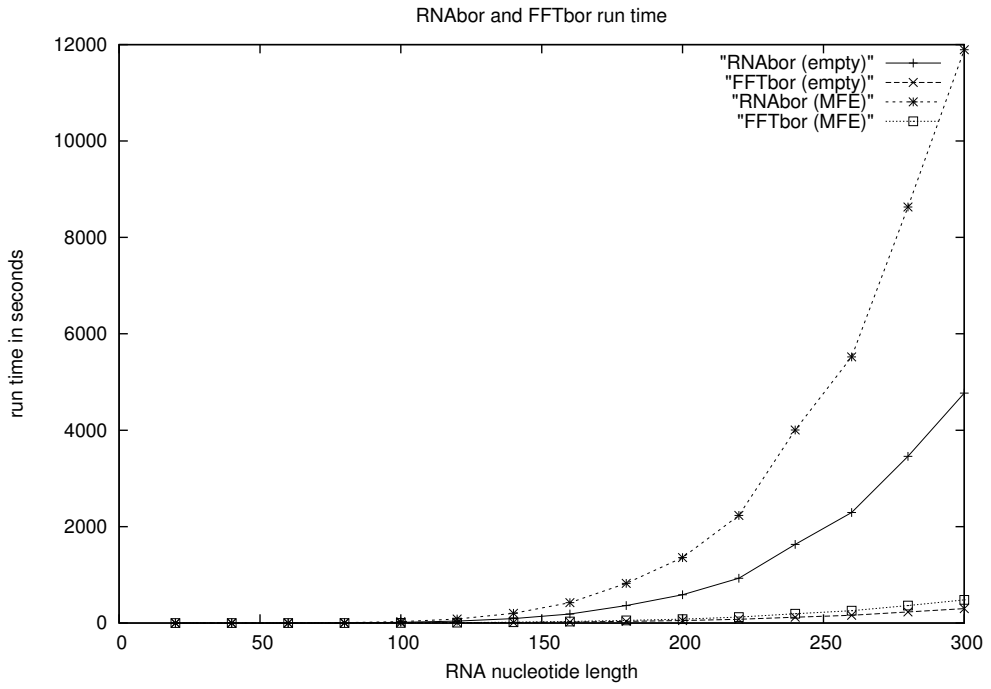

FIGURE 3.5: Run times in seconds for `RNAbor` and `FFTbor`, on random RNA of length $20, 40, 60, \ldots, 300$ in step size of 20 nt. Each algorithm was run with the empty initial structure $\mathcal{S}^*$, see rows `RNAbor` (empty), `FFTbor` (empty), and with the minimum free energy structure as the 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`.

3.7.1 OpenMP parallelization of `FFTbor`

OpenMP is a simple and flexible multi-platform shared-memory parallel programming environment, that supports parallelizations of C/C++ code—see `http://openmp.org/`. Using OpenMP primitives, we created multiple threads to evaluate the polynomial $\mathcal{Z}(x)$ on different complex nth roots of unity. Figure 3.6 presents benchmarks, executed on a 24-core AMD Opteron 6172 with 2.10GHz and 64GB RAM, for the speedup of `FFTbor` as a function of the number of cores. The left panel of Figure 3.6 describes average

run time in seconds (± one standard deviation) for running `FFTbor` on random RNA of length $200, 250, 300, 400, 450, 500$ with either 1 or 2 cores. The right panel of Figure 3.6 presents similar data for running `FFTbor` on $2, 3, 6, 4, 12, 15, 20$ cores.

Len	1	2
200	123.2 ± 16.2	61.8 ± 8.0
250	331.1 ± 27.2	166.1 ± 13.7
300	723.4 ± 59.9	365.2 ± 30.1
350	1,380.8 ± 95.2	698.4 ± 46.9
400	2,239.1 ± 210.9	1,129.5 ± 104.3
450	3,635.0 ± 857.4	1,980.9 ± 126.5
500	5,076.7 ± 1,292.1	3,389.8 ± 788.4

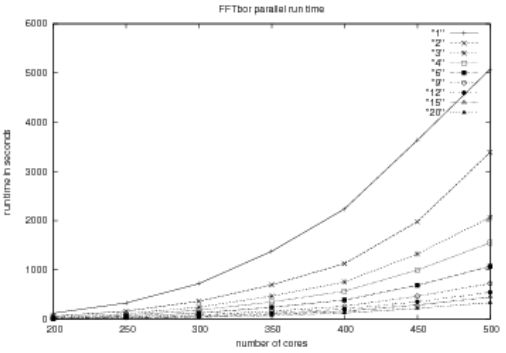

FIGURE 3.6: *(Left)* Table showing parallel run times in seconds for `FFTbor`, using OpenMP `http://openmp.org/`. Column headers 1 and 2 indicate the number of cores used in the computational experiment. For each sequence length $200, \ldots, 500$, five random RNAs were generated using equal probability for each nucleotide A,C,G,U. Run time in seconds, plus or minus one standard deviation, are given for a 24-core AMD Opteron 6172 with 2.10GHz and 64GB RAM, with only 1 (resp. 2) cores used. *(Right)* Graph showing parallel run time of `FFTbor` on an AMD Opteron 6172 with 2.10GHz and 64GB RAM, using respectively 1,2,3,4,6,9,12,15,20 cores.

Chapter 4

FFTbor2D

4.1 Introduction

In this chapter, we present the `FFTbor2D` algorithm and accompanying software. `FFTbor2D`, like `FFTbor` described in Chapter 3, is an algorithm which computes the paramerized partition function for an input RNA sequence $\mathbf{s}$. `FFTbor2D` computes the two-dimensional coarse energy landscape for $\mathbf{s}$ 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 $d_{\mathrm{BP}}(\mathcal{S}, \mathcal{A}) = x$ and $d_{\mathrm{BP}}(\mathcal{S}, \mathcal{B}) = y$ (where d_{BP} is as defined in equation 3.3). 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 5.

4.1.1 Organization

This chapter is organized in the following fashion. Because the history for this work arises naturally from the history provided in Section 3.2, 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. We elect to refrain from describing applications of `FFTbor2D` until Chapter 5, where the software is applied to quickly approximate mean first passage time and equilibrium time for the folding of RNA molecules between any two distinct, user provided structures $\mathcal{A}, \mathcal{B}$.

4.2 Background

4.3 Derivation of the `FFTbor2D` algorithm

For expository clarity, we describe `FFTbor2D` and all recursions in terms of the Nussinov energy model Nussinov and Jacobson [1980] (same as in Chapter 3), 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 Xia et al. [1999], 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.

4.3.1 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

$$\mathbf{Z}_{1,n}^{x,y} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=x,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=y}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{4.1}$$

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. [2009], Lorenz et al. generalized the dynamic programming recursions of our earlier work Freyhult et al. [2007], to yield recursions for the partition function $\mathbf{Z}_{i,j}^{x,y}$ in equation (4.1). In the context of the Nussinov model, $\mathbf{Z}_{i,j}^{x,y}$ is equal to

$$\mathbf{Z}_{i,j-1}^{x-\alpha_0,y-\beta_0} + \sum_{\substack{s_k,s_j\in\mathbb{B},\\ i\le k<j}} \left(e^{\frac{-E_0(k,j)}{RT}} \sum_{u+u'=x-\alpha(k)} \sum_{v+v'=y-\beta(k)} \mathbf{Z}_{i,k-1}^{u,v} \cdot \mathbf{Z}_{k+1,j-1}^{u',v'} \right) \tag{4.2}$$

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 3.2), 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 $\beta(k) = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k+1,j-1]} \cup \{(k,j)\})$.

4.3.2 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*

$$\mathcal{Z}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{4.3}$$

where (constant) coefficients

$$z_{rn+s} = \mathbf{Z}_{1,n}^{r,s} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{4.4}$$

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

$$\mathcal{Z}(a_0) = y_0, \ldots, \mathcal{Z}(a_{n^2-1}) = y_{n^2-1}, \tag{4.5}$$

then Lagrange polynomial interpolation (equation 3.14) guarantees that we can determine the coefficients c_{rn+s} of $\mathcal{Z}(x)$, for $0 \leq r, s < n$. Due to technical difficulties concerning numerical robustness observered while working on the `FFTbor` software (Chapter 3), 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 (4.2), can be used to compute the *polynomial* $\mathcal{Z}_{i,j}(x)$ defined by

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s}(i,j) \cdot x^{rn+s} = \sum_{k=0}^{n^2-1} z_k(i,j) \cdot x^k \tag{4.6}$$

where

$$z_{rn+s}(i,j) = \mathbf{Z}_{i,j}^{r,s} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}}. \tag{4.7}$$

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

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{4.8}$$

where

$$z_{rn+s}(i,j) = \mathbf{Z}_{i,j}^{r,s}. \tag{4.9}$$

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

$$\begin{aligned} &\mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n+\beta_0} + \\ &\sum_{\substack{s_k,s_j \in \mathbb{B}, \\ i \leq k < j}} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \cdot x^{\alpha(k)n+\beta(k)} \right) \end{aligned} \tag{4.10}$$

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* $\alpha(k) = d_{BP}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i,k-1]} \cup \mathcal{A}_{[k+1,j-1]} \cup \{(k,j)\})$, $\beta(k) = d_{BP}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k+1,j-1]} \cup \{(k,j)\})$.

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 `RNA2Dfold` Lorenz et al. [2009]. 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

$$\mathcal{Z}_{i,j-1}(\rho)\cdot\rho^{\alpha_0 n+\beta_0}+ \sum_{\substack{(s_k,s_j)\in\mathbb{B},\\ i\le k<j}}\left(e^{\frac{-E_0(k,j)}{RT}}\cdot\mathcal{Z}_{i,k-1}(\rho)\cdot\mathcal{Z}_{k+1,j-1}(\rho)\cdot\rho^{\alpha(k)n+\beta(k)}\right) \tag{4.11}$$

Note that this approach is similar to what we do in `FFTbor`—specifically equation (3.23)—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 $0\le i\le n^2-1$, 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

$$\mathbf{Y}=(y_0,\ldots,y_{n^2-1})^{\mathrm{T}}=\begin{pmatrix} y_0\\ y_1\\ \vdots\\ y_{n^2-1}\end{pmatrix} \tag{4.12}$$

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 $O(n^2)$.

4.3.3 Polynomial interpolation

Our plan is to determine the coefficients of the polynomial $\mathcal{Z}(x)$ in equation (4.3) by polynomial interpolation. For reasons of numerical stability, we instead determine the coefficients of the polynomial $p(x)$, defined by

$$p(x) = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} p_{rn+s}x^{rn+s} = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} \frac{\mathbf{Z}_{1,n}^{rn+s}}{\mathbf{Z}} x^{rn+s}, \tag{4.13}$$

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 4.4.3. 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 4.1. In the next section, we explain a highly non-trivial improvement of this algorithm to reduce time by a factor of 4.

4.4 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 4.2. *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 $y \in \{d_0 - x + 2k : k = 0, \ldots, x\}$.*

Pseudocode for `FFTbor2D`

PURPOSE: Computes the m most significant digits of probabilities $p_{rn+s} = \mathbf{z}^{r,s}_{1,n}/\mathbf{z}$
INPUT: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structures $\mathcal{A}$, $\mathcal{B}$ of $\mathbf{s}$, integer m
OUTPUT: $p_{rn+s} = \mathbf{z}^{r,s}_{1,n}/\mathbf{z}$ to $\lfloor \log_{10}(2^m) \rfloor$ 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 (4.26)
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
14 end function
```

FIGURE 4.1: 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 27—only the $\lfloor \log_{10}(2^m) \rfloor = 8$ most significant digits are computed with `FFTbor2D`. 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 $O(n^2 \log n)$.

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*

$$d_{BP}(\mathcal{A}, \mathcal{S}) + d_{BP}(\mathcal{S}, \mathcal{B}) \equiv d_{BP}(\mathcal{A}, \mathcal{B}) \bmod 2 \tag{4.14}$$

first noticed in Lorenz et al. [2009], 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. Lorenz et al. [2009] 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 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 4.3. *If* $\mathcal{Z}(x)$ *is the complex polynomial defined in equation (4.3), then for any complex* n*th root of unity* α*, it is the case that* $\mathcal{Z}(\overline{\alpha}) = \overline{\mathcal{Z}(\alpha)}$*.*

Lemma 4.4. *Let* $\mathcal{Z}(x)$ *be defined by equation (4.3), 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 4.5. *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*

$$\nu^k = -(\nu^{-(M/2-k)}) = -\overline{\nu^{M/2-k}}. \tag{4.15}$$

Lemma 4.2 is proved by simple induction; Lemma 4.3 is proved by a computation involving binomial coefficients; Lemma 4.4 is immediate by the parity observation above,

resulting from Lemma 4.2; Lemma 4.5 is elementary, relying on Euler's formula and trigonometric addition formulas. Details proofs of Lemmas 4.3, 4.4, 4.5 can be found in supplementary information.

Lemma 4.2 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 4.3 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 4.2 and Lemma 4.3, 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.

4.4.1 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 or equal to 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

$$\begin{aligned}\mathcal{Z}(x) &= \sum_{r=0}^{N}\sum_{s=0}^{N} z_{rN+s} x^{rN+s} \\ &= \sum_{k=0}^{(N+1)^2-1} z_{\pi_1(k)\cdot(N+1)+\pi_2(k)} x^{\pi_1(k)\cdot(N+1)+\pi_2(k)} \\ &= \sum_{k=0}^{(N+1)^2-1} z_k x^k\end{aligned} \tag{4.16}$$

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

$$\mathcal{Z}(x) = \sum_{r=0}^{N} \sum_{s=0}^{N} z_{rN+s} x^{rN+s} \quad = \sum_{k=0}^{(N+1)^2-1} z_k x^k. \tag{4.17}$$

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 (4.17) are zero.

In this case, we have $\mathcal{Z}(x) = \sum_{k=0}^{M_0-1} z_{2k} x^{2k}$. But then $\mathcal{Z}(x) = \mathcal{Y}(u) = \sum_{k=0}^{M_0-1} b_k u^k$, 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

$$\mathcal{Y}(u) = \sum_{k=0}^{M_0-1} b_k u^k = \sum_{k=0}^{M_0-1} z_{2k} x^{2k}. \tag{4.18}$$

Since $\mathcal{Y}(x^2) = \mathcal{Z}(x)$, we have $\mathcal{Y}(\exp(\frac{2\pi ik}{M/2})) = \mathcal{Y}(\exp(\frac{4\pi ik}{M})) = \mathcal{Z}(\exp(\frac{2\pi ik}{M}))$, hence we can use the previous recursions (4.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi ik}{M}))$. Instead of performing

M evaluations of $\mathcal{Z}(x)$ at M-roots of unity, this requires only $M_0 = M/2$ evaluations of $\mathcal{Y}(u)$ 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 $\mathcal{Y}(x)$. 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 (4.17) 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 $\mathcal{Z}(x) = x \cdot \mathcal{Y}(u)$, where $\mathcal{Y}(u) = \sum_{k=0}^{M_0-1} b_k u^k$, 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 $\mathcal{Y}(u) = \sum_{k=0}^{M_0-1} b_k u^k$ by evaluating M_0 many complex M_0-roots of unity. Since $\mathcal{Z}(x) = x \cdot \mathcal{Y}(x^2)$, $\mathcal{Y}(x^2) = x^{-1} \cdot \mathcal{Z}(x)$, so $\mathcal{Y}(\exp(\frac{2\pi ik}{M/2})) = \mathcal{Y}(\exp(\frac{4\pi ki}{M})) = \exp(\frac{-2\pi ik}{M}) \cdot \mathcal{Z}(\exp(\frac{2\pi ik}{M}))$, employing the previous recursions (4.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi ik}{M}))$. Note, that unlike the Case 1, since $\mathcal{Z}(x) = x \cdot \mathcal{Y}(x^2)$, we have $\mathcal{Y}(x^2) = \frac{\mathcal{Z}(x)}{x}$, 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 $\mathcal{Y}(u)$ 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 $\mathcal{Y}(x)$. But then, we immediately obtain the full polynomial $\mathcal{Z}(x)$, since $\mathcal{Z}(x) = x \cdot \mathcal{Y}(x^2)$, 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

$$\mathcal{Y}(x^2) = \mathcal{Z}(x) \tag{4.19}$$

while if the parity is odd, then

$$\mathcal{Y}(x^2) = \frac{1}{x} \cdot \mathcal{Z}(x). \tag{4.20}$$

4.4.2 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 $M_0 = \frac{M}{2}$, let $\nu = \exp(\frac{2\pi i}{M})$ and $\omega = \nu^2 = \exp(\frac{2\pi \cdot 2i}{M}) = \exp(\frac{2\pi i}{M_0})$. Clearly, ν is a principal complex M-root of unity, while ω is a principal complex M_0-root of unity. Evaluate $\mathcal{Z}$ for each M_0-root of unity that belongs to the first quadrant, and apply Lemma 4.3 to infer the values of $\mathcal{Z}$ for each M_0-root of unity that belongs to the fourth quadrant. More precisely, we compute $\mathcal{Z}(\nu^k)$, for $k = 0, \ldots, \frac{M_0}{2}$, and by Lemmas 4.3, 4.4, 4.5 infer that for $k = \frac{M_0}{2} + 1, \ldots, M_0 - 1$, we have $\mathcal{Z}(\nu^k) = -1^{d_0} \cdot \overline{\mathcal{Z}(\nu^{M_0-k})}$, where $d_0 = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$. This is justified in the following.

By induction on $k = \frac{M_0}{2} + 1, \ldots, M_0 - 1$, we have

$$\begin{aligned}
\mathcal{Y}(\omega^k) &= \mathcal{Y}(\nu^{2k}) \\
&= \begin{cases} \mathcal{Z}(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ \frac{1}{\nu^k} \cdot \mathcal{Z}(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \mathcal{Z}(-\overline{\nu^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ \nu^{-k} \cdot \mathcal{Z}(-\overline{\nu^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \mathcal{Z}(\overline{\nu^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ \nu^{-k} \cdot \mathcal{Z}(\overline{\nu^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases} \\
&= \begin{cases} \overline{\mathcal{Z}(\nu^{(M_0-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ -\nu^{-k} \cdot \overline{\mathcal{Z}(\nu^{(M_0-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases}
\end{aligned} \tag{4.21}$$

Line 1 follows by definition, since $\omega = \nu^2$; line 2 follows by equations (4.19) and (4.20); line 3 follows by Lemma 4.5; line 4 follows by Lemma 4.4. Thus if $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$ is even, then

$$y_k = \mathcal{Y}(\omega^k) = \begin{cases} \mathcal{Z}(\nu^k) & \text{for } k = 0, \ldots, \frac{M_0}{2} \\ \overline{\mathcal{Z}(\nu^{M_0-k})} & \text{for } k = \frac{M_0}{2}+1, \ldots, M_0-1 \end{cases} \tag{4.22}$$

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

$$y_k = \mathcal{Y}(\omega^k) = \begin{cases} \nu^{-k} \cdot \mathcal{Z}(\nu^k) & \text{for } k = 0, \ldots, \frac{M_0}{2} \\ -\nu^{-k} \cdot \overline{\mathcal{Z}(\nu^{M_0-k})} & \text{for } k = \frac{M_0}{2}+1, \ldots, M_0-1 \end{cases} \tag{4.23}$$

It follows that values $y_0, \ldots, y_{M_0-1}$ can be obtained by only $\frac{M}{4}$ evaluations of $\mathcal{Z}(x)$.

4.4.3 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.

$$V_{M_0} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{M_0-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(M_0-1)} \\ 1 & \omega^3 & \omega^6 & \ldots & \omega^{3(M_0-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{M_0-1} & \omega^{2(M_0-1)} & \ldots & \omega^{(M_0-1)(M_0-1)} \end{pmatrix} \tag{4.24}$$

As described in Chapter 3, the Fast Fourier Transform is the $O(n \log n)$ algorithm, which computes the Discrete Fourier Transform (DFT), defined as the matrix product $\mathbf{Y} = V_{M_0}\mathbf{A}$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{M_0-1} \end{pmatrix} = V_n \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{M_0-1} \end{pmatrix} \tag{4.25}$$

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

$$a_j = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \, \omega^{-kj} = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \, \nu^{-2kj} \tag{4.26}$$

for $j = 0, \ldots, M_0 - 1$.

Since we defined $\mathbf{Y}$ in equation (4.12) by $\mathbf{Y} = (y_0, \ldots, y_{M_0-1})^{\mathrm{T}}$, where $y_0 = \mathcal{Z}(\alpha_0), \ldots,$ $y_{M_0-1} = \mathcal{Z}(\alpha_{M_0-1})$ 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 (4.3) can be computed using the FFT. However, in practice we encounter the same issues of numerical instability observed in Section 3.4.3, and adopt a similar 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 `FFTbor2D` which includes the performance enhancements described in Section 4.4 follows below.

Pseudocode for improved `FFTbor2D`

PURPOSE: Computes the m most significant digits of probabilities $p_{r\cdot(N+1)+s} = \mathbf{z}^{r,s}_{1,n}/\mathbf{z}$
INPUT: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structures $\mathcal{A}$, $\mathcal{B}$ of $\mathbf{s}$, integer m
OUTPUT: $p_{r\cdot(N+1)+s} = \mathbf{z}^{r,s}_{1,n}/\mathbf{z}$ to $\lfloor \log_{10}(2^m) \rfloor$ significant digits for $r, s = 0, \ldots, N$

```
1  function FFTbor2D Improved(s, A, B, m)
2      n ← length(s)
3      N ← n + (n mod 2)
4      M ← (N + 1)^2 + ((N + 1)^2 mod 4)
5      M_0 ← M/2
6      for k ← 0, (N + 1)^2 − 1 do          ▷ Note that k ← π_1(k) · M + π_2(k)
7          π_1(k) ← ⌊k/(N+1)⌋
8          π_2(k) ← k mod (N + 1)
9      end for
10     for k ← 0, M − 1 do                  ▷ Compute all complex M and M_0-roots of unity
11         ν_k ← exp(2πik/M)
12         if k < M_0 then
13             ω_k ← exp(2πik/M_0)
14         end if
15     end for
16     for k ← 0, M_0 − 1 do
17         if d_BP(A, B) mod 2 = 0 then      ▷ From equation (4.22)
18             if k ≤ M_0/2 then
19                 y_k ← Z(ν^k)
20             else
21                 y_k ← conj(Z(ν^{M_0−k}))
22             end if
23         else                              ▷ From equation (4.23)
24             if k ≤ M_0/2 then
25                 y_k ← ν^{−k} · Z(ν^k)
26             else
27                 y_k ← −ν^{−k} · conj(Z(ν^{M_0−k}))
28             end if
29         end if
30     end for
```

for $k \leftarrow 0, M_0 - 1$ **do** ▷ Note that $\mathcal{Z}(\nu_0) = \mathbf{Z}$
  $y_k \leftarrow 2^m \cdot \frac{y_k}{\mathcal{Z}(\nu_0)}$
end for
for $k \leftarrow 0, M_0 - 1$ **do** ▷ Compute IDFT from equation (4.26)
  $a_k \leftarrow \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \omega^{-kj}$
end for
for $k \leftarrow 0, M - 1$ **do** ▷ Change the polynomial back to degree $M - 1$
  if $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) \bmod 2 = 0$ **then**
    if $k \bmod 2 = 0$ **then**
      $p_{\pi_1(k)\cdot(N+1)+\pi_2(k)} \leftarrow a_{k/2}$
    else
      $p_{\pi_1(k)\cdot(N+1)+\pi_2(k)} \leftarrow 0$
    end if
  else
    if $k \bmod 2 = 0$ **then**
      $p_{\pi_1(k)\cdot(N+1)+\pi_2(k)} \leftarrow 0$
    else
      $p_{\pi_1(k)\cdot(N+1)+\pi_2(k)} \leftarrow a_{\frac{k-1}{2}}$
    end if
  end if
end for
for $k \leftarrow 0, (N+1)^2 - 1$ **do** ▷ Truncate to m significant digits
  $p_k \leftarrow 2^{-m} \cdot \lfloor p_k \rfloor$
end for
return $p_0, \ldots, p_{(N+1)^2-1}$ ▷ Return all $p(r,s) = p_{r\cdot(N+1)+s} = \frac{\mathbf{Z}_{1,n}^{r,s}}{\mathbf{Z}}$
end function

FIGURE 4.2: Pseudocode to compute the m most significant digits for probabilities $p_{r\cdot(N+1)+s} = \mathbf{z}_{1,n}^{r,s}/\mathbf{z}$. In our implementation, due to numerical stability issues in the FFT engine, precision parameter m has an upper bound of 27—only the $\lfloor \log_{10}(2^m) \rfloor = 8$ most significant digits are computed with `FFTbor2D`. 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 $O(n^2 \log n)$.

4.5 Performance characteristics of FFTbor2D

To perform comparative benchmarking between RNA2Dfold and FFTbor2D, we took precision parameter $m = 8$, and proceeded as follows. For each sequence length $n = 20, 25, 30, \ldots, 300$, we generated 100 random sequences using probability 0.25 for each nucleotide A, C, G, U. For a given RNA sequence $\mathbf{s}$, the metastable structure $\mathcal{A}$ was taken to be the MFE structure of $\mathbf{s}$. Using RNAbor, we determined that value $k_0 \geq 10$, for which partition function $\mathbf{Z}^{k_0}$ constitutes a visible peak in the graphical output — see Figure 2 and 3 of Freyhult et al. [2007] for an example. Subsequently, metastable structure $\mathcal{B}$ was taken to be that structure having minimum free energy over all structures, whose base pair distance from $\mathcal{A}$ was k_0.

For all $0 \leq x, y \leq n$, RNA2Dfold and FFTbor2D were benchmarked in the computation of all Boltzmann probabilities $p(x, y) = \frac{\mathbf{Z}^{x,y}}{\mathbf{Z}}$, where x [resp. y] represents base pair distance to metastable structure $\mathcal{A}$ [resp. $\mathcal{B}$]. Care was taken for both software to employ the same energy model (Turner99 energy model, no dangles, suppression of minimum free energy structure computations for RNA2Dfold) and the same number of parallel threads (8 threads using OpenMP). Nonetheless, there are slight differences in the energy models — namely, RNA2Dfold includes mismatch penalties for multiloop stems and for exterior loops, while FFTbor2D does not. Even in the computation of the partition function Z, for spliced leader RNA from *L. collosoma* of length 56 nt, RNA2Dfold -d0 obtains a value of -9.660419 kcal/mol, while FFTbor2D obtains -9.660543 kcal/mol; similarly, for attenuator RNA of length 73 nt, RNA2Dfold -d0 obtains a value of -22.171785 kcal/mol, while FFTbor2D obtains -22.173213 kcal/mol. Note that the straightforward calculation

of the partition function, following McCaskill's algorithm McCaskill [1990] makes no use of the FFT engine, and thus the differences cannot be due to floating point or precision issues.

For benchmarking purposes, to allow for a fair comparison of `FFTbor2D` with `RNA2Dfold`, we restricted therange of x, y in the same manner as done in the source code of `RNA2Dfold`. In that code, parameters K [resp. L] are defined respectively to be the sum of the number of base pairs in reference structure $\mathcal{A}$ [resp. $\mathcal{B}$] plus the number of base pairs in the maximum matching (Nussinov) structure which contains no base pair of $\mathcal{A}$ [resp. $\mathcal{B}$]. For $x \geq K, y \geq L$, both `RNA2Dfold` and `FFTbor2D` set $p(x, y) = 0$. For the benchmarking results displayed in Figures 4.3, 4.4, 4.5, the values x, y are restricted in `FFTbor2D` to $0 \leq x, y \leq \max(K, L)$, while $0 \leq x \leq K$ and $0 \leq y \leq L$ in `RNA2Dfold`.

Figure 4.3 depicts average run time of `RNA2Dfold` and `FFTbor2D` as a function of RNA sequence length, for random RNA sequences of lengths 20–200 and their metastable structures $\mathcal{A}, \mathcal{B}$, as previously explained. We see that both programs have roughly comparable run times for sequences of length up to approximately 80 nt, while `FFTbor2D` is demonstrably faster for longer sequences. Figure 4.4 presents $\log_{10}$ run time as a function of sequence length, in order to more clearly determine the crossover point in performance. `RNA2Dfold` is marginally faster for sequences of length up to roughly 80 nt, though the difference is in the millisecond range. Figure 4.5 shows that the standard deviation of run times on random sequences is tiny for `FFTbor2D` compared with `RNA2Dfold`, where standard deviation increases rapidly as a function of sequence length. This figure shows that run time of `RNA2Dfold` depends on sequence details,

as well as sequence length, while the run time of `FFTbor2D` depends only on sequence length.

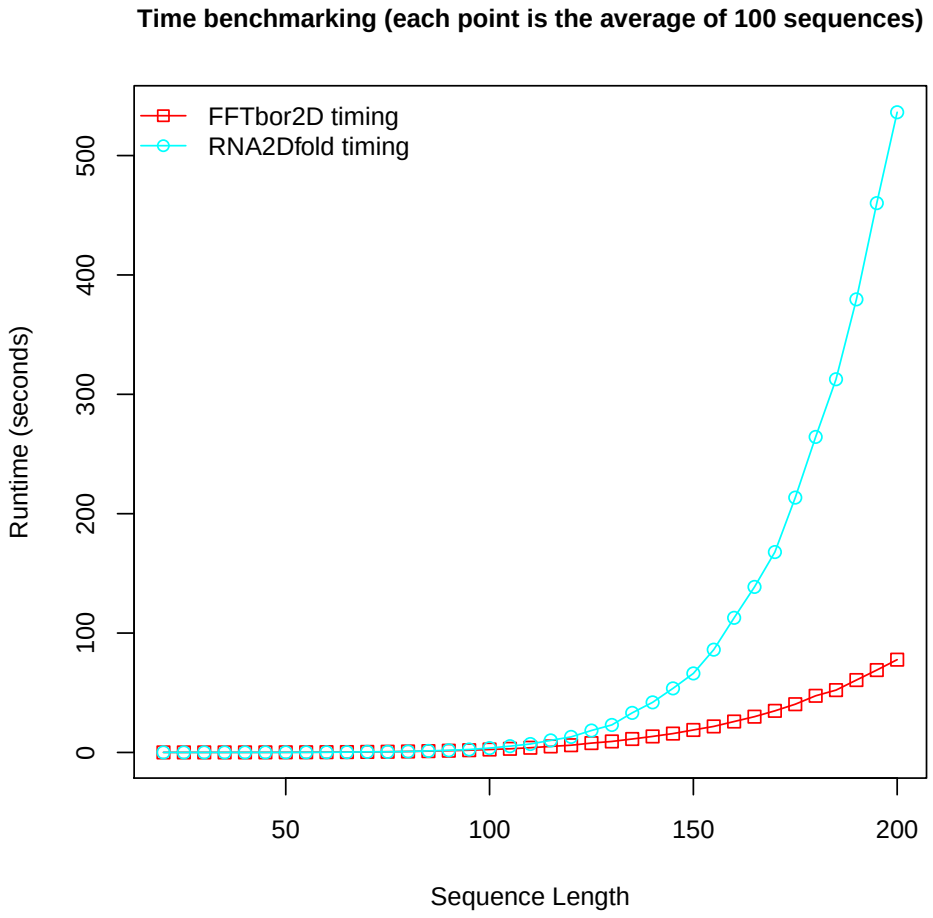

FIGURE 4.3: Run time in seconds for `RNA2Dfold` and `FFTbor2D` on random RNA sequences of length 20–200 nt, where sequence generation and choice of metastable structures $\mathcal{A}, \mathcal{B}$ is described in the text. Beyond a length of approximately 80 nt, `FFTbor2D` is demonstrably faster.

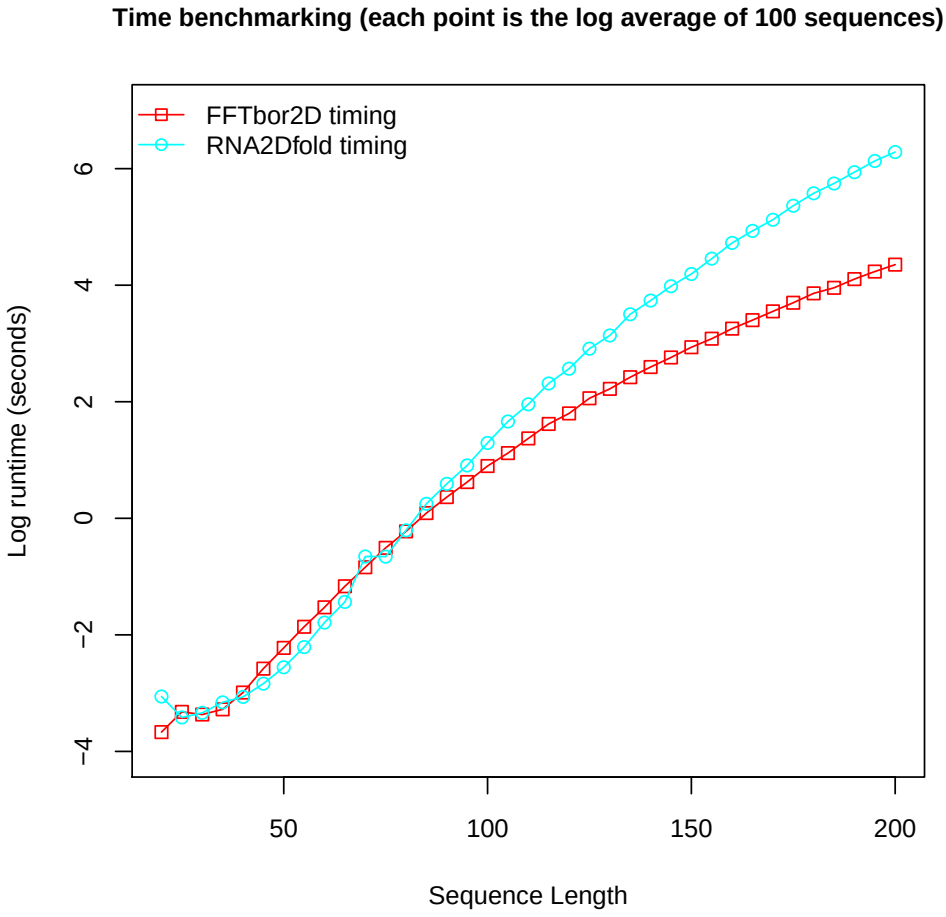

FIGURE 4.4: Logarithm of run time in seconds for `RNA2Dfold` and `FFTbor2D` on random RNA sequences of length less than 200 nt, for same data as that in Figure 4.3. By taking $\log_{10}$ of the run times, crossover points are apparent, where `FFTbor2D` is faster than `RNA2Dfold`. For very small sequences, `RNA2Dfold` is faster, though since both programs converge in a fraction of a second, this difference is of no practical consequence.

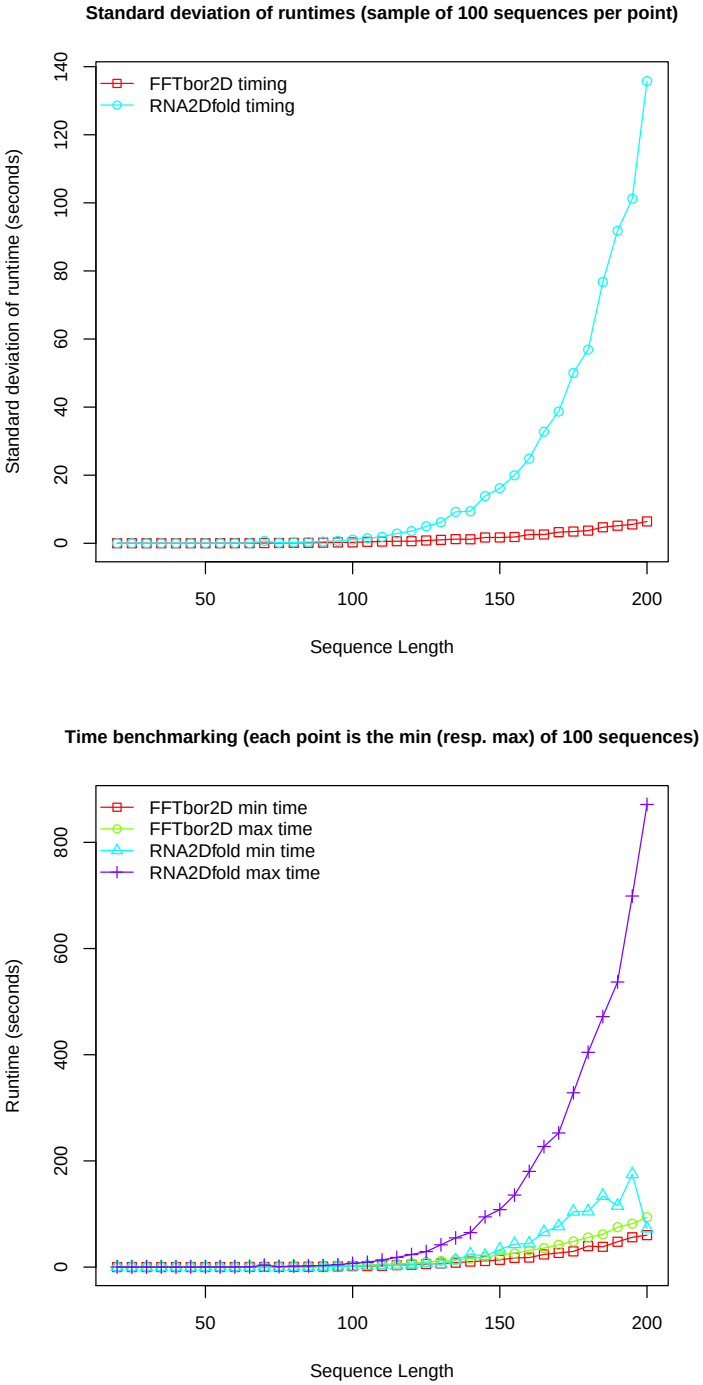

FIGURE 4.5: *(Top)* Standard deviation of run times of `RNA2Dfold` and `FFTbor2D` as a function of sequence length n. *(Bottom)* Minimum and maximum run times for `RNA2Dfold` and `FFTbor2D`. For each collection of 100 random sequences of length n, the minimum and maximum run time for a sequence of that length was computed. Taken together, these figures clearly show the run time dependence of `RNA2Dfold` on particular sequences, while the run time of `FFTbor2D` depends only on sequence length, rather than sequence details.

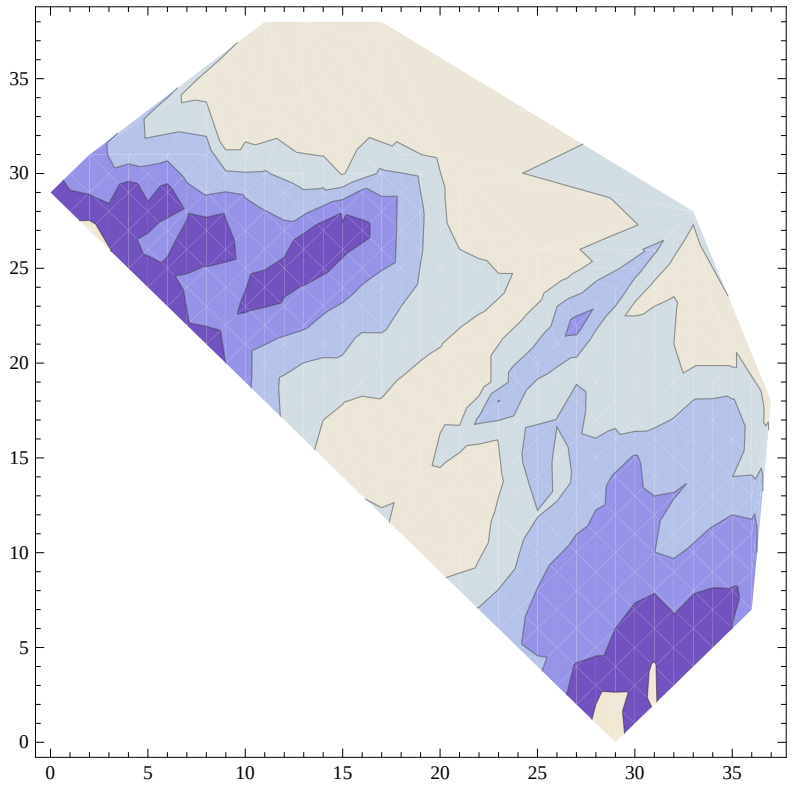

FIGURE 4.6: 2D projection of energy landscape for Spliced Leader (SL) RNA from *L. collosoma*, having sequence and metastable secondary structures:

$\mathbf{s}$ = AACUAAAACAAUUUUUGAAGAACAGUUUCUGUACUUCAUUGGUAUGUAGAGACUUC

$\mathcal{A}$ = ..((...((((((..(((((.((((...)))).)))))..)))).)))..))....

$\mathcal{B}$ = ...((((((((((((.....))))).))))))).

The x-axis [resp. y-axis] represents base pair distance between metastable structure $\mathcal{A}$ [resp. $\mathcal{B}$], while the z-axis represents the ensemble free energy $-RT \log \mathbf{Z}^{x,y}$, where $\mathbf{Z}^{x,y}$ is computed in `FFTbor2D` by $\mathbf{Z}^{x,y} = p(x,y) \cdot \mathbf{Z}$. Low energy positions (x, y) correspond to high Boltzmann probability positions. The left panel depicts a heat map of the ensemble free energy, while the right panel depicts a contour map with level curves. In analogy with mountain climbing, one expects an optimal path to follow along the valley regions in traversing the landscape from $\mathcal{A}$ to $\mathcal{B}$. Data produced with `FFTbor2D`; graphics produced using Mathematica.

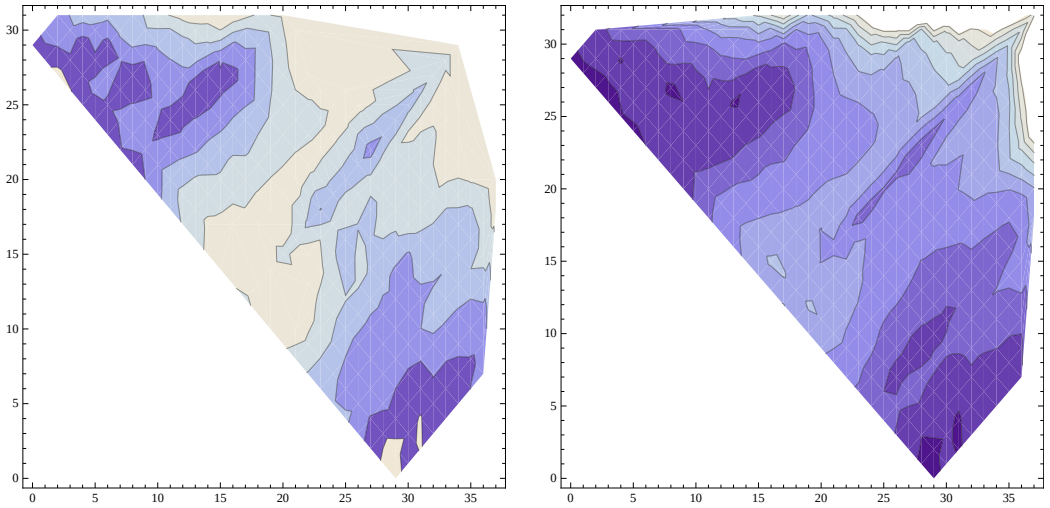

FIGURE 4.7: 2D projection of energy landscape for Spliced Leader (SL) RNA from *L. collosoma*, as in Figure 4.6, except that in the right panel, ensemble free energy $-RT \log \mathbf{Z}^{x,y}$ is computed from the values of $\mathbf{Z}^{x,y}$ output by `RNA2Dfold`, while in the right panel, ensemble free energy is computed from the values $\mathbf{Z}^{x,y} = p(x,y) \cdot \mathbf{Z}$, where values $p(x,y)$ are output by `RNA2Dfold`. The loss of detail in the 2D energy landscape is caused uniquely by working with probabilities $p(x,y)$, rather than partition function values $\mathbf{Z}^{x,y}$. Data produced with `RNA2Dfold`; graphics produced using Mathematica.

Chapter 5

Hermes

5.1 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 4), 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.

5.1.1 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.

5.2 Background

5.3 Traditional approaches for kinetics

To better understand the underlying algorithms behind the software, we describe two traditional approaches in kinetics.

5.3.1 Mean first passage time

Consider a physical process, which when monitored over time, yields the stochastic sequence $q_0, q_1, q_2, \ldots$ of discrete, observed states. If the transition from state q_t to q_{t+1}

depends only on q_t at time t and not on the historical sequence of prior states visited, as often assumed in the case of protein or RNA folding, then a Markov chain provides a reasonable mathematical model to simulate the process.

A first-order, time-homogeneous *Markov chain* $\mathbb{M} = (Q, \pi, P)$ is given by a finite set $Q = \{1, \ldots, n\}$ of states, an initial probability distribution $\pi = (\pi_1, \ldots, \pi_n)$, and the $n \times n$ *transition probability matrix* $P = (p_{i,j})$. At time $t = 0$, the initial state of the system is $q_t = i$ with probability π_i, and at discrete time $t = 1, 2, 3, \ldots$, the system makes a transition from state i to state j with probability $p_{i,j}$; i.e. the conditional probability $Pr[q_{t+1} = j | q_t = i] = p_{i,j}$. Define the *population occupancy frequency* of visiting state i at time t by $p_i(t) = Pr[q_t = i]$. Denote $p_{i,j}^{(t)} = Pr[q_t = j | q_0 = i]$ and notice that the (i, j)-th entry of the t-th power P^t of matrix P equals $p_{i,j}^{(t)}$.

The *mean first passage time* (MFPT) or *hitting time* for the Markov chain $\mathbb{M}$, starting from initial state x_0 and proceeding to the target state x_∞, is defined as the sum, taken over all paths ρ from x_0 to x_∞, of the path length *length*(ρ) times the probability of path ρ, where *length*$(\rho_0, \ldots, \rho_n)$ is defined to be n. In other words, $\mathrm{M}FPT = \sum_\rho Pr[\rho] \cdot$ *length*(ρ), where the sum is taken over sequences $\rho = \rho_0, \ldots, \rho_n$ of states where $\rho_0 = x_0$ and $\rho_n = x_\infty$, and no state is visited more than once in the path ρ.

Given the target state x_∞, MFPT can be exactly determined by computing the inverse $(I - P_{x_\infty}^-)^{-1}$, where I is the $(n-1) \times (n-1)$ identity matrix and $P_{x_\infty}^-$ denotes the $(n-1) \times (n-1)$ matrix obtained from the Markov chain transition probability matrix P, by deleting the row and column with index x_∞. Letting $\mathbf{e}$ denote the $(n-1) \times 1$ column vector consisting entirely of 1's, it can be shown that mean first passage time

from state x_0 to state x_∞ is the x_0-th coordinate of column vector $(I - P^-_{x_\infty})^{-1} \cdot \mathbf{e}$ Meyer [1975].

The *stationary* probability distribution $p^* = (p^*_1, \ldots, p^*_n)$ is a row vector such that $p^* \cdot P = p^*$; i.e. $p^*_j = \sum_i^n p^*_i \cdot p_{i,j}$, for all $1 \leq j \leq n$. It can be shown that the stationary probability p^*_i is the limit, as m tends to infinity, of the frequency of visiting state i in a random walk of length m on Markov chain $\mathbb{M}$. It is well-known that the stationary distribution exists and is unique for any finite aperiodic irreducible Markov chain Clote and Backofen [2000].

The Metropolis Monte Carlo algorithm Metropolis et al. [1953] can be used to simulate a random walk from initial state x_0 to target state x_∞, when energies are associated with the states, as is the case in macromolecular folding, where free energies can be determined for protein and RNA conformations from mean field theory, quantum theory, or experimental measurements. In such cases, a *move set* defines the set N_x of conformations reachable in unit time from conformation x, and the transition probability matrix $P = (p_{x,y})$ is defined as follows:

$$p_{x,y} = \begin{cases} \frac{1}{|N_x|} \cdot \min\left(1, \exp(-\frac{E(y)-E(x)}{RT})\right) & \text{if } y \in N_x \\ 1 - \sum_{k \in N_x} p_{x,k} & \text{if } x = y \\ 0 & \text{if } y \notin N_x, \text{ and } x \neq y \end{cases} \tag{5.1}$$

If $p^*_x \cdot p_{x,y} = p^*_y \cdot p_{y,x}$ holds for all distinct $x, y \in Q$, then *detailed balance* is said to hold, or equivalently the Markov chain $\mathbb{M}$ is said to be *reversible*. If transitional probabilities are defined as in equation (5.1), and if neighborhood size is constant ($|N_x| = |N_y|$ for all

x, y), then it is well-known that the stationary probability distribution $p^* = (p_1^*, \ldots, p_n^*)$ is the Boltzmann distribution; i.e.

$$p_k^* = \frac{\exp(-E(k)/RT)}{\mathbf{Z}} \tag{5.2}$$

where $E(k)$ is the energy of conformation k at temperature T, R is the universal gas constant, T is absolute temperature, and the *partition function* $Z = \sum_{k=1}^{n} \exp(-E(k)/RT)$ is a normalization constant Clote and Backofen [2000], Waterman [1995]. If neighborhood size is not constant, as in the case where states are RNA secondary structures and transitions are restricted to the addition or removal of a single base pair, then by *Hasting's trick*, an equivalent Markov chain can be defined which satisfies detailed balance—see equation (5.17).

Following Anfinsen's experimental work on the denaturation and refolding of bovine pancreatic ribonuclease Anfinsen [1973], the native conformation is assumed to be the ground state having minimum free energy. These results justify the use of the Monte Carlo Algorithm 5.1 in macromolecular kinetics and structure prediction.

The mean first passage time from state x to state y can be approximated by repeated runs of the Monte Carlo algorithm. In particular, Šali, Shakhnovich, and Karplus used such Monte Carlo simulations to investigate the Levinthal paradox of how a protein can fold to its native state within milliseconds to seconds. By repeated Monte Carlo simulations using a protein lattice model, Šali et al. observed that a large energy difference between the ground state and the first misfolded state appears to be correlated with fast folding.

Pseudocode for the Metropolis Monte Carlo algorithm

PURPOSE: Compute discrete time t to move from x_0 to x_∞, with upper limit of $T_{\max}$
INPUT: Starting state x_0, target state x_∞ and upper bound on time $T_{\max}$
OUTPUT: Discrete time t taken to reach x_∞ from x_0

```
1  function METROPOLIS(x_0, x_∞, T_max)
2      t ← 0
3      x ← x_0
4      while x ≠ x_∞ AND t < T_max do
5          y ∈ N_x
6          if E(y) < E(x) then                          ▷ Greedy move
7              x ← y
8          else                                         ▷ Metropolis move
9              z ∈ (0, 1)
10             if z < exp(−(E(y)−E(x))/(RT)) then
11                 x ← y
12             end if
13         end if
14         t ← t + 1
15     end while
16     return t                  ▷ Return discrete time t to travel from x_0 to x_∞
17 end function
```

FIGURE 5.1: The function METROPOLIS implements a discrete time simulation of folding trajectories for a Markov chain.

5.3.2 Equilibrium time

A continuous time *Markov process* $\mathbb{M} = (Q, \pi, P(t))$ is given by a finite set $Q = \{1, \ldots, n\}$ of states, the initial probability distribution π, and the $n \times n$ matrix $P(t) = (p_{i,j}(t))$ of probability transition functions.

Letting q_t denote the state at (continuous) time t, the probability that the initial state q_0 at time 0 is k is π_k, while

$$p_{i,j}(t) = Pr[q_t = j | q_0 = i]. \tag{5.3}$$

The matrix $P'(t)$ of derivatives, defined by

$$P'(t) = \begin{pmatrix} \frac{dp_{1,1}}{dt}(t) & \dots & \frac{dp_{1,n}}{dt}(t) \\ \vdots & \ddots & \vdots \\ \frac{dp_{n,1}}{dt}(t) & \dots & \frac{dp_{n,n}}{dt}(t) \end{pmatrix}, \tag{5.4}$$

can be shown to satisfy

$$P'(t) = P(t) \cdot R \tag{5.5}$$

where $R = (r_{i,j})$ is an $n \times n$ *rate matrix* with the property that each diagonal entry is -1 times the row sum

$$r_{i,i} = -\sum_{j \neq i} r_{i,j}. \tag{5.6}$$

Define the *population occupancy* distribution $p(t) = (p_1(t), \dots, p_n(t))$ by

$$p_i(t) = Pr[q(t) = i] = \sum_{k=1}^{n} \pi_k p_{k,i}(t) \tag{5.7}$$

where $q(t)$ denotes the state of the Markov process at (continuous) time t.

In the case of macromolecular folding, where Markov process states are molecular conformations and conformational energies are available, it is typical to define the rate matrix $R = (r_{x,y})$ as follows:

$$r_{x,y} = \begin{cases} \min\left(1, \exp(-\frac{E(y)-E(x)}{RT})\right) & \text{if } y \in N_x \\ -\sum_{k \in N_x} p_{x,k} & \text{if } x = y \\ 0 & \text{if } y \notin N_x, \text{ and } x \neq y \end{cases} \tag{5.8}$$

The *master equation* is defined by the matrix differential equation

$$\frac{dp(t)}{dt} = p(t) \cdot R \tag{5.9}$$

or equivalently, for all $1 \leq x \leq n$,

$$\frac{dp_x(t)}{dt} = \sum_{y=1}^{n} \left(p_y(t) \cdot r_{y,x} - p_x(t) \cdot r_{x,y}\right) = \sum_{x \neq y} \left(p_y(t) \cdot r_{y,x} - p_x(t) \cdot r_{x,y}\right). \tag{5.10}$$

As in the case of Markov chains, $p^* = (p_1^*, \ldots, p_n^*)$ is defined to be the stationary distribution if $p^* \cdot P(t) = p^*$; i.e. $p_k^* = Pr[q(0) = k]$ implies that $Pr[q(t) = k] = p_k^*$ for all $t \in \mathbb{R}$ and $1 \leq k \leq n$. Define the *equilibrium distribution* $p^* = (p_1^*, \ldots, p_n^*)$ to be the unique solution for $p(t)$, when the master equation (5.10) is set to equal zero; i.e.

$$\sum_{x \neq y} p_x^* \cdot r_{x,y} = \sum_{x \neq y} p_y^* \cdot r_{y,x}. \tag{5.11}$$

If the equilibrium distribution exists, then necessarily it is equal to the stationary distribution. A Markov process is said to satisfy detailed balance if $p^*_x \cdot r_{x,y} = p^*_y \cdot r_{y,x}$, for all $1 \leq x, y \leq n$, where the rate matrix $R = (r_{x,y})$.

The rate equation R for is usually defined as in (5.8) for Markov processes which model macromolecular folding, hence it is easy to see that such Markov processes satisfy detailed balance and moreover that the equilibrium distribution is the Boltzmann distribution; i.e. $p^*_x = \exp(-E(x)/RT)$ for all $1 \leq x \leq n$. Since detailed balance ensures that the eigenvalues of the rate matrix R are real, one can solve the matrix differential equation (5.10) by diagonalizing the rate matrix, and thus obtain the solution

$$p(t) = \sum_{k=1}^{n} c_k \mathbf{v}_k \, e^{\lambda_k t} \tag{5.12}$$

where $p(t) = (p_1(t), \ldots, p_n(t))$, and the values c_k are determined by the initial population occupancy distribution $p(0)$ at time 0. Here $\mathbf{v}_k$ denotes the kth eigenvector and λ_k the kth eigenvalue. In particular, $c_k = \left(p(0) \cdot T^{-1})\right)_k$, where the j-th row of T is the j-th left eigenvector of R, and $p(0)$ is the population occupancy distribution at time $t = 0$. If the eigenvalues are labeled in decreasing order, then $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, and the largest eigenvalue $\lambda_1 = 0$ has eigenvector p^*, corresponding to the equilibrium population occupancy distribution, which in this case is the Boltzmann distribution. The remaining $n - 1$ eigenvalues are negative, and their corresponding eigenvectors correspond to nonequilibrium kinetic *relaxation modes*.

In our software `Hermes`, we prefer to work with column vectors and right eigenvectors due to our usage of the GSL scientific computing library, and so the population occupancy

frequency $p(t)$ is defined to be the column vector $p(t) = (p_1(t), \ldots, p_n(t))^{\mathrm{T}}$. Let $\mathbf{P} = (p_1(0), \ldots, p_n(0))^{\mathrm{T}}$ be the column vector of initial population occupancy probabilities, $\mathbf{t}^1, \ldots, \mathbf{t}^n$ be the right eigenvectors and $\lambda_1, \ldots, \lambda_n$ be the corresponding right eigenvalues of the transpose R^{T} of the rate matrix. Letting T be the $n \times n$ matrix, whose columns are $\mathbf{t}^1, \ldots, \mathbf{t}^n$, using standard matrix algebra Franklin [2000], it can be shown that

$$p(t) = \sum_{j=1}^{n} (T^{-1}\mathbf{P}(0))_j \, \mathbf{t}^j \, e^{\lambda_j t}. \tag{5.13}$$

or for a single state i,

$$p_i(t) = \sum_{j=1}^{n} \sum_{k=1}^{n} \left(T_{ij} \, T_{jk}^{-1} \, \mathbf{P}_k(0) \right)_j e^{\lambda_j t}. \tag{5.14}$$

If we assume that the initial population starts in a single state i', i.e. $\mathbf{P}_{i'}(0) = 1$, equation 5.14 can be simplified to

$$p_i(t) = \sum_{j=1}^{n} T_{ij} \, T_{ji'}^{-1} \, e^{\lambda_j t}. \tag{5.15}$$

The *equilibrium time* can be directly computed by using a nonlinear solver to solve for t in

$$p^* = \sum_{j=1}^{n} (T^{-1}\mathbf{P}(0))_j \, \mathbf{t}^j \, e^{\lambda_j t} \tag{5.16}$$

where $p^* = (p_1^*, \ldots, p_n^*)^{\mathrm{T}}$ and $p_k^* = \frac{\exp(-E(k)/RT)}{\mathbf{Z}}$. However, we have found it more expedient to compute the *equilibrium time* as the smallest t_0, such that for $t \in \{t_0 + 1, t_0 + 2, t_0 + 3, t_0 + 4\}$, the absolute difference $|p(t)[x_\infty] - p(t_0)[x_\infty]| < \epsilon$, for $\epsilon = 10^{-4}$, where x_∞ is the target RNA structure (usually taken to be the minimum free energy structure, though this is not necessary for the software). We provide a more detailed explanation of the reasoning behind this decision in Section 5.4.4.2.

In Gillespie [1976] Gillespie described a very influential algorithm to simulate a finite Markov process. The pseudocode, is given in Algorithm 5.2. Though Gillespie's original application was for chemical kinetics, Flamm et al. adapted the method for the kinetics of RNA secondary structure folding, as implemented in `Kinfold` Flamm et al. [2000], Flamm [1998].

5.4 Software within the `Hermes` suite

The `Hermes` software package was developed on the Macintosh OS X operating system (10.9.2–10.11) 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

Pseudocode for the Gillespie algorithm

PURPOSE: Compute continuous time t to move from x_0 to x_∞, with a limit of $T_{\max}$
INPUT: Starting state x_0, target state x_∞ and upper bound on time $T_{\max}$
OUTPUT: Continuous time t taken to reach x_∞ from x_0

```
1  function GILLESPIE(x_0, x_∞, T_max)
2      t ← 0
3      x ← x_0
4      while x ≠ x_∞ AND t < T_max do
5          Φ ← 0                                   ▷ Φ is the flux out of x
6          for y ∈ N_x do
7              r_{x,y} ← min(1, exp(−(E(y)−E(x))/(RT)))
8              Φ ← Φ + r_{x,y}
9          end for
10         for y ∈ N_x do
11             r_{x,y} ← r_{x,y}/Φ
12         end for
13         z_1 ∈ (0, 1)
14         t ← t + (−(1/Φ) ln(z_1))                ▷ Update t to reflect time spent in state x
15         z_2 ∈ (0, 1)
16         s ← 0
17         for y ∈ N_x do                          ▷ Use roulette wheel to select new state for x
18             s ← s + r_{x,y}
19             if z_2 ≤ s then
20                 x ← y
21                 break
22             end if
23         end for
24     end while
25     return t                                    ▷ Return continuous time t to travel from x_0 to x_∞
26 end function
```

FIGURE 5.2: The function GILLESPIE implements a continuous time simulation of folding trajectories for a Markov process.

Fourier Transform Frigo and Johnson [2005] ($\geq$ 3.3.4) `http://www.fftw.org/`, Gnu Scientific Library GSL ($\geq$ 1.15) `http://www.gnu.org/software/gsl/`, Vienna RNA Package Lorenz et al. [2011] ($\geq$ 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 5.3). These packages compile into both standalone executables and archive files. The archives provide an 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++ wrappers 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 wrapper such as Python, Perl or R. We believe the API that `Hermes` provides via its archives to be one of the exciting aspects of this software for other developers—the C++ sourcecode comprising `FFTmfpt` is only 56 lines of code (LOC) and `FFTeq` is only 58 LOC.

5.4.1 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 Meyer [1975]—see Section

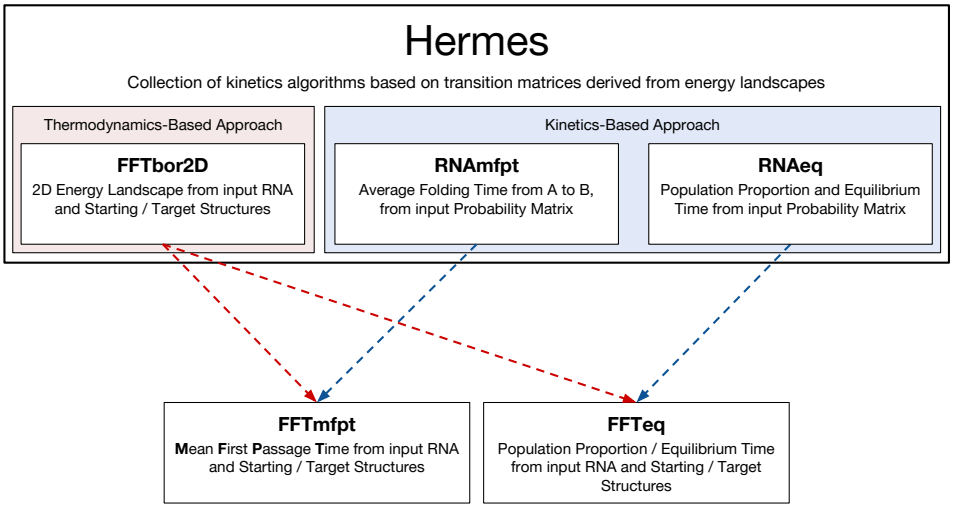

FIGURE 5.3: Overall organization of `Hermes`. `FFTbor2D`, `RNAmfpt`, and `RNAeq` are three distinct software packages we have developed, which compile into both standalone executables and archive files, providing an API that allow novel applications development using source from each of the packages, without having to copy-and-paste relevant functions. The applications `FFTmfpt` and `FFTeq` are C++ wrappers that use data structures and functions from `FFTbor2D`, `RNAmfpt` and `RNAeq`. `FFTmfpt` computes the mean first passage time (MFPT) for an RNA secondary structure to fold from an initial structure, such as the empty structure or a given metastable structure, into a target structure, such as the minimum free energy (MFE) structure or possibly the Rfam Gardner et al. [2011] consensus structure. `FFTeq` uses spectral decomposition to compute the equilibrium time and the fraction of the population of RNA structures that are equal to a given target structure, as a function of time.

5.3.1. 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 probability grid, and the final column p is the stationary probability $p_{i,j}$. From this input, the probability transition matrix is constructed using equation (5.24) and the mean first passage time is computed by matrix inversion.

Given RNA sequence $\mathbf{s}$ and secondary structure x, let $N(x)$ denote the set of neighboring secondary structures of $\mathbf{s}$, whose base pair distance with x is one. Define the Markov chain $\mathcal{M}(\mathbf{s}) = (\mathcal{S}, P)$, where $\mathcal{S}$ denotes the set of all secondary structures of $\mathbf{s}$, $p^*(x)$ is the stationary probability of structure x, defined by $p^*(x) = \exp(-E(x)/RT)/\mathbf{Z}$, $\mathbf{Z} = \sum_x \exp(-E(x)/RT)$, and the transition probability matrix $P = (p_{x,y})$ is defined either with or without the Hastings modification as follows.

With the Hastings modification,

$$p_{x,y} = \begin{cases} \frac{1}{|N(x,y)|} \cdot \min\left(1, \frac{p^*(y)}{p^*(x)} \cdot \frac{N(x)}{N(y)}\right) & \text{if } y \in N(x) \\ 0 & \text{if } x \neq y, y \notin N(x) \\ 1 - \sum_{k \in N(x)} p_{x,k} & \text{if } x = y \end{cases} \tag{5.17}$$

Without the Hastings modification,

$$p_{x,y} = \begin{cases} \frac{1}{|N(x,y)|} \cdot \min\left(1, \frac{p^*(y)}{p^*(x)}\right) & \text{if } y \in N(x) \\ 0 & \text{if } x \neq y, y \notin N(x) \\ 1 - \sum_{k \in N(x)} p_{x,k} & \text{if } x = y \end{cases} \tag{5.18}$$

The exact value of mean first passage time (MFPT) can be computed as follows. Let x_0 [resp. x_∞] denote the empty structure [resp. MFE structure] for sequence $\mathbf{s}$ (here we have implicitly identified integer indices with secondary structures). Let $\widetilde{P}_{x_\infty}$ be the matrix obtained from P by removal of the row and column with index x_∞, and I denote the $(n-1) \times (n-1)$ identity matrix, where $n = |\mathcal{S}|$ is the number of secondary structures of $\mathbf{s}$. Let e denote the vector of size $n-1$, each of whose coordinates is 1. It is well-known Meyer [1975] that for each $k \neq x_\infty$, the kth coordinate of the vector $(I - \widetilde{P}_{x_\infty})^{-1} \cdot \mathbf{e}$ is exactly equal to the mean first passage time from the structure with

index k to the target structure x_∞. In particular, the MFPT from the empty structure to the MFE structure is computable by applying matrix inversion using a computational library such as the GSL.

Since this computation of the mean first passage time is mathematically exact, we consider that MFPT to be the *gold standard* value for RNA folding kinetics.

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` Senter et al. [2012] or arbitrary transition matrices without any change to the underlying implementation. The software additionally provides many options for defining the format of the input as well as the structure of the graph underlying the Markov chain.

Due to our interest in efficiently estimating MFPT using coarse grained 2D energy landscapes, the default input for `RNAmfpt` (and `RNAeq`) is a 2D probability grid, as described above. We additionally allow as alternative input an energy landscape (whereby transition probabilities are defined as in equation 5.1). Some of the options to influence the underlying transition matrix 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.

5.4.2 Approximate mean first passage time with `FFTmfpt`

`FFTmfpt` approximates the mean first passage time of a given RNA sequence folding from input structure $\mathcal{A}$ to $\mathcal{B}$, by computing *exactly* 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 $\mathcal{A}, \mathcal{B}$, 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 $p_{(x,y),(x',y')} = \frac{P(x',y')}{P(x,y)}$.

More formally, given an RNA sequence $\mathbf{s}$, starting structure $\mathcal{A}$, and target structure $\mathcal{B}$, we proceed in the following fashion. Let $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$ denote the base pair distance between structures $\mathcal{A}, \mathcal{B}$. Then $\mathbf{Z}_{x,y} = \sum_{\mathcal{S}} \exp(-E(\mathcal{S})/RT)$, where the sum is over structures $\mathcal{S}$, such that $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{S}) = x$, and $d_{\mathrm{BP}}(\mathcal{B}, \mathcal{S}) = y$; as well the partition function $\mathbf{Z} = \sum_{\mathcal{S}} \exp(-E(\mathcal{S})/RT)$, where the sum is over all secondary structures $\mathcal{S}$ of the sequence $\mathbf{s}$.

Let $d_0 = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B})$, the base pair distance between initial structure $\mathcal{A}$ and target structure $\mathcal{B}$. Let n denote the length of sequence $\mathbf{s}$. Define the Markov chain $\mathcal{M}_1(\mathbf{s}) = (Q_1, P_1)$, where

$$\begin{aligned} Q_1 = \{(x, y) : 0 \leq x, y \leq n, (x + y \bmod 2) = (d_0 \bmod 2), \\ (d_0 \leq x + y), (x \leq d_0 + y), (y \leq d_0 + x)\}. \end{aligned} \tag{5.19}$$

For reference, we say that the *parity condition* holds for (x, y) if

$$(x + y \bmod 2) = (d_0 \bmod 2). \tag{5.20}$$

We say that the *triangle inequality* holds for (x, y) if

$$(d_0 \leq x + y), (x \leq d_0 + y), (y \leq d_0 + x) \tag{5.21}$$

Since we allow transitions between secondary structures that differ by exactly one base pair, Markov chain transitions are allowed to occur only between states $(x, y), (u, v) \in Q_1$, such that $u = x \pm 1$, $v = y \pm 1$. However, we have found that for some RNA sequences, there is no non-zero probability path from $(0, d_0)$ to $(d_0, 0)$ (corresponding to a folding pathway from structure $\mathcal{A}$ to $\mathcal{B}$). Since `FFTbor2D` computes *probabilities* $p(x, y)$ by polynomial interpolation using the Fast Fourier Transform, any probability less than 10^{-8} is set to 0. Also with `RNA2Dfold`, it may arise that there is no non-zero probability path from structure $(0, d_0)$ to $(d_0, 0)$.

To address this situation, we proceed as follows. Let $\epsilon = 10^{-8}$ and for all $(x, y) \in Q_1$, modify probabilities $p(x, y)$ by

$$p(x, y) = \frac{p(x, y) + \epsilon/|Q_1|}{1 + \epsilon}. \tag{5.22}$$

This operation corresponds to adding the negligeable value of $\epsilon = 10^{-8}$ to the total probability, thus ensuring that there are paths of non-zero probability between any

two states. After this ϵ-modification and renormalization, *when using the Hastings modification*, the transition probabilities $P((u,v)|(x,y))$ are given by

$$P((u,v)|(x,y)) = \begin{cases} \frac{1}{|N(x,y)|} \cdot \min\left(1, \frac{p(u,v)}{p(x,y)} \cdot \frac{N(x,y)}{N(u,v)}\right) & \text{if } (u,v) \in N(x,y) \\ 0 & \text{if } (u,v) \neq (x,y), (u,v) \notin N(x,y) \\ 1 - \sum_{(k,\ell)\in N(x,y)} P((k,\ell)|(x,y)) & \text{if } (u,v) = (x,y) \end{cases} \tag{5.23}$$

Here the set $N(x,y)$ of adjacent neighbors is defined by $N(x,y) = \{(u,v) \in Q_1 : u = x \pm 1, v = y \pm 1\}$, and the stationary probability $p(x,y)$ is obtained from `FFTbor2D`.

Without the Hastings modification, the transition probabilities $P((u,v)|(x,y))$ are instead given by

$$P((u,v)|(x,y)) = \begin{cases} \frac{1}{|N(x,y)|} \cdot \min\left(1, \frac{p(u,v)}{p(x,y)}\right) & \text{if } (u,v) \in N(x,y) \\ 0 & \text{if } (u,v) \neq (x,y), (u,v) \notin N(x,y) \\ 1 - \sum_{(k,\ell)\in N(x,y)} P((k,\ell)|(x,y)) & \text{if } (u,v) = (x,y) \end{cases} \tag{5.24}$$

As we report in Section 5.5.1, given an RNA sequence $\mathbf{s}$, if $\mathcal{A}$ is the empty structure and $\mathcal{B}$ the MFE structure of $\mathbf{s}$, 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.

5.4.3 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.

Define the continuous Markov process $\mathcal{M} = (\mathcal{S}, R)$, where $\mathcal{S}$ is defined as in Section 5.4.1, R is the *rate matrix* defined by

$$k_{x,y} = \begin{cases} \min\left(1, \frac{p^*(y)}{p^*(x)}\right) & \text{if } y \in N(x) \\ 0 & \text{if } x \neq y, y \notin N(x) \\ -\sum_{k \in N(x)} p_{x,k} & \text{if } x = y \end{cases} \tag{5.25}$$

Clearly the rate matrix satisfies *detailed balance*; i.e. $p^*(x) \cdot k_{x,y} = p^*(y) \cdot k_{y,x}$ for all distinct $x, y \in \mathcal{S}$. In fact, the rate matrix for Markov processes is usually defined as above, precisely to ensure detailed balance, which then implies that all eigenvalues of the rate matrix are real, thus permitting explicit solution of the population occupancy frequency for all states. We additionally considered a Hastings correction for the rate matrix, where $k_{x,y} = \min(1, \frac{p^*(y)}{p^*(x)} \cdot \frac{N(x)}{N(y)})$. The correlation in Table 5.1 for equilibrium time computed from this modified rate matrix is somewhat better than without the

Hastings correction. However, the Hastings correction is never used for rate matrices, hence we only consider the usual definition of rate matrix given in equation (5.25).

`RNAeq` can additionall call the Vienna RNA Package program `RNAsubopt` Wuchty et al. [1999], 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 `RNAsubopt`, 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.

5.4.4 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` (in a fashion similar to `FFTmfpt`, explained in Section 5.4.2), which makes the graph size more tractable than structural sampling with `RNAsubopt` using `RNAeq`, even with constraints.

This method consists of computing the equilibrium time from the master equation for the coarse-grain Markov process $\mathcal{M} = (Q_1, R)$, where Q_1 is defined in equation (5.19), and the rate matrix $R = (k((x, y), (u, v)))$ is defined by

$$k((u,v),(x,y)) = \begin{cases} \min\left(1, \frac{p(u,v)}{p(x,y)}\right) & \text{if } (u,v) \in N(x,y) \\ 0 & \text{if } (u,v) \neq (x,y), (u,v) \notin N(x,y) \\ -\sum_{(k,\ell)\in N(x,y)} P((k,\ell)|(x,y)) & \text{if } (u,v) = (x,y) \end{cases} \tag{5.26}$$

Equilibrium time is then computed for this Markov process using equation 5.12.

5.4.4.1 Population occupancy curves with `FFTeq`

Population occupancy curves provide valuable visual insight into the folding kinetics of RNA molecules. Equation 5.14 describes a computation for the probability distribution $p(t) = (p_1(t), \ldots, p_n(t))^{\mathrm{T}}$ of all states $1 \leq x \leq n$ at time t. Rather than plot $p(t)$ as a function of t, we find it more visually impactful to plot the log-logistic function $p(\log_{10}(t))$, yielding the curvess seen in Figure 5.4.

5.4.4.2 Approximating equilibrium time from occupancy curves

The computation of an equilibrium time value from the eigendecomposition of the rate matrix is a rather thorny issue. While in principle a nonlinear solver can be used to compute the equilibrium time t (from equation 5.16), in practice due to what we believe to be numeric instability issues the approach was intractable. For this reason, we estimate the equilibrium time from the population occupancy curve, an approach also taken by `treekin` Wolfinger et al. [2004]. Our first approximation of equilibrium time involved using a sliding window approach, where we compute the smallest t_0, such that for $t \in \{t_0+t_\Delta, t_0+t_{2\Delta}, t_0+t_{3\Delta}, t_0+t_{4\Delta}\}$, the absolute difference $|p(t)[x_\infty] - p(t_0)[x_\infty]| < \epsilon$, for $\epsilon = 10^{-4}$ and step size t_Δ. In systems where there are local energy minima, this

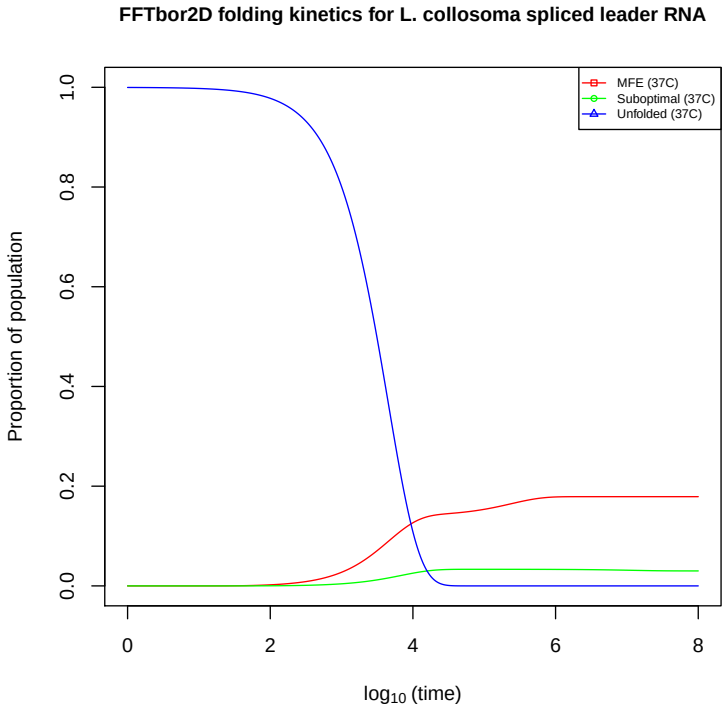

FIGURE 5.4: Population occupancy curves computed with `FFTeq` for the 56 nt conformational switch *L. collosoma* spliced leader RNA, with sequence `AACUAAAACAAUUUUUGAAGAACAGUUUCUGUACUUCAUUGGUAUGUAGA` GACUUC. The dot bracket format for the MFE structure, as computed by version 2.1.7 of `RNAfold -d0`, is `.....((((((((((((.....))))))..)))))))..` with free energy −8.6 kcal/mol, while that of the the alternate suboptimal structure is `..((...((((((..(((((.((((...)))).)))))..)))).))))..)).....` with free energy −7.5 kcal/mol. In the case of the MFE structure, the equilibrium occupancy $P(t_\infty)$, which `Hermes` approximates as 0.17893806 should equal the Boltzmann probability 0.17909227, since the MFE structure is the only structure at distance x_0 [resp. y_0] from the reference structures $\mathcal{A}$ (empty structure) [resp. $\mathcal{B}$ (MFE structure)]. As well, if there are few other low energy structures at the same base pair distance x_1 [resp. y_1] from $\mathcal{A}$ [resp. $\mathcal{B}$] as that of the alternate suboptimal structure, then we expect that the occupancy probability 0.03003426 for the (x_1, y_1) be approximately the Boltzmann probability 0.03005854 of the alternate structure.

approach has the drawback of prematurely indicating equilibrium, as seen in Figure 5.6. To address this concern, we introduce the concept of δ, ϵ equilibrium.

Given a starting state i', target state i, and user definable parameters $w = 5, t_\Delta = 10^{-3}, \delta = 10^{-3}$ and $\epsilon = 10^{-4}$, we compute the estimated equilirium using the following algorithm.

Pseudocode for estimating equilibrium time

PURPOSE:	Estimate time $\log_{10}(t)$ at which state x_0 appears within ϵ of equilibrium
INPUT:	States x_0, x_∞; $T_{\min}$, $T_{\max}$; $w = 5, t_\Delta = 10^{-3}, \delta = 10^{-3}, \epsilon = 10^{-4}$
OUTPUT:	Time $\log_{10}(t)$ at which $p_{x_\infty}(10^t)$ varies by no more than ϵ within w

---

```
1  function EstimateEquilibrium(x_0, x_∞, T_min, T_max, t_Δ, δ, ϵ, w)
2      t ← SoftBound(x_∞, T_min, T_max, t_Δ, δ, s)
3      if WindowEq(x_∞, t, t_Δ, w, ϵ) then
4          while t > T_min AND WindowEq(x_∞, t, t_Δ, w, ϵ) do
5              t ← t − t_Δ
6          end while
7      end if
8      while t < T_max AND NOT WindowEq(x_∞, t, t_Δ, w, ϵ) do
9          t ← t + t_Δ
10     end while
11     if t = T_min then                    ▷ x doesn't leave equilibrium after T_min
12         return −∞
13     else if t = T_max then               ▷ x doesn't reach equilibrium before T_max
14         return ∞
15     else
16         return t
17     end if
18 end function
```

```
19 function SoftBound(i, T_min, T_max, t_Δ, δ, s)
20     i_∞ ← p_i(10^{T_max})                          ▷ From equation 5.15
21     t ← T_min + (T_max − T_min)/2
22     t' ← T_max
23     while |t − t'| > t_Δ do          ▷ Binary search for time t within δ of T_max
24         t_temp ← t
25         if |p_i(10^t) − p_i(10^{t'})| > δ then
26             t ← t + s ∗ |t−t'|/2      ▷ s is 1 [resp. −1] for the right [resp. left] bound
27         else
28             t ← t − s ∗ |t−t'|/2
29         end if
30         t' ← t_temp
31     end while
32     return t
33 end function
34 function WindowEq(x, t, t_Δ, w, ϵ)  ▷ Is x within ϵ of equilibrium for [t, t_{(w−1)Δ}]
35     for i ← 1, w − 1 do
36         if |p_x(10^{t+t_{iΔ}}) − p_x(10^t)| > ϵ then
37             return false
38         end if
39     end for
40     return true
41 end function
```

FIGURE 5.5: The function EstimateEquilibrium computes the smallest $t_0 > t'$, such that for $t \in \{t_0 + t_\Delta, t_0 + t_{2\Delta}, t_0 + t_{3\Delta}, t_0 + t_{4\Delta}\}$, the absolute difference $|p(t)[x_\infty] - p(t_0)[x_\infty]| < \epsilon$ and $|p(t')[x_\infty] - p(T_{\max})[x_\infty]| \approx \delta|$. If t' is already in equilibrium, we relax the constraint that $t_0 > t'$ and instead find the first $t_0 < t'$ that satisfies the equilibrium requirements within the window w. SoftBound is a helper function that uses binary search to find the starting time t' from which the window starts. WindowEq returns a boolean value if the state of interest x varies by no more than ϵ across the window w starting at time t.

In using the approach outlined in Figure 5.5, the correlation between $p_{x_\infty}(t_0)$ and

$\frac{\exp(-E(x_\infty)/RT)}{\mathbf{Z}}$ is 0.9997, and visual inspection of the estimated equilibrium times (Figure 5.6) are much better than the naïve approach used by **`treekin`**. We also allow defining the equilibrium time to be the smallest t_0, such that for $t \in \{t_0 + t_\Delta, t_0 + t_{2\Delta}, t_0 + t_{3\Delta}, t_0 + t_{4\Delta}\}$, the absolute difference $|p(t)[x] - p(t_0)[x]| < \epsilon$ for all $x \in Q$ using Algorithm 5.5; however, results suggest that this definition is inferior to the consideration of a single single target state i, perhaps due to numerical instability issues when Q is taken to be the set of all secondary structures for sequences in the benchmarking set described later.

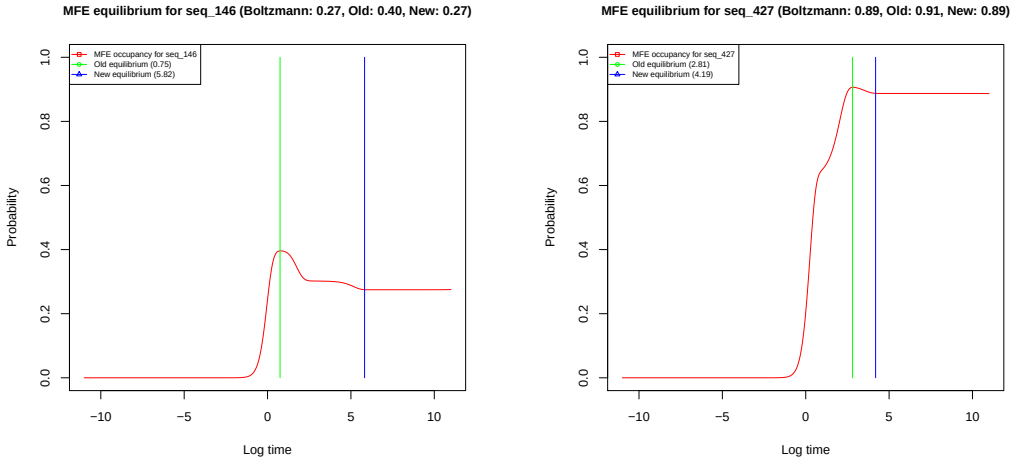

FIGURE 5.6: Example of the differences in using the simple sliding window approach (green) versus the approach outlined in Algorithm 5.5 (blue). There are two representative sequences shown from the benchmarking dataset described in Section 5.5, sequence #146 *(Left)* and #427 *(Right)*. Note that the $p(t)$ for the approach from Algorithm 5.5 is much closer to the computed Boltzmann probability—the correlation across the dataset of $1,000$ sequences is 0.9997.

5.5 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 5.7. 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 Gardner et al. [2011].

Figure 5.8 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. 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

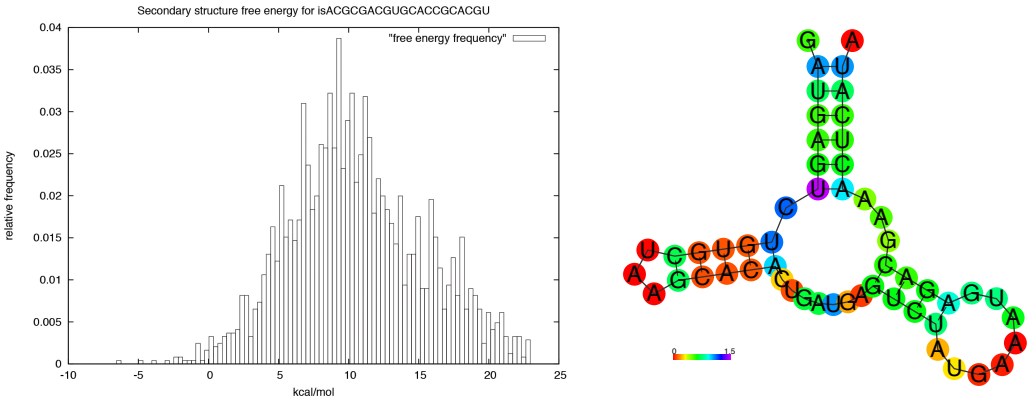

FIGURE 5.7: *(Left)* Histogram of free energies of secondary structures of `ACGCGACGUGCACCGCACGU`, which range from −6.5 to +25 kcal/mol, with mean of 10.695 kcal/mol. *(Right)* Minimum free energy structure of the 54 nt Peach Latent Mosaic Viroid (PLMVd) AJ005312.1/282-335, which is identical to the consensus structure from Rfam 11.0 Gardner et al. [2011]. `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 Dotu et al. [2014]. 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`.

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.

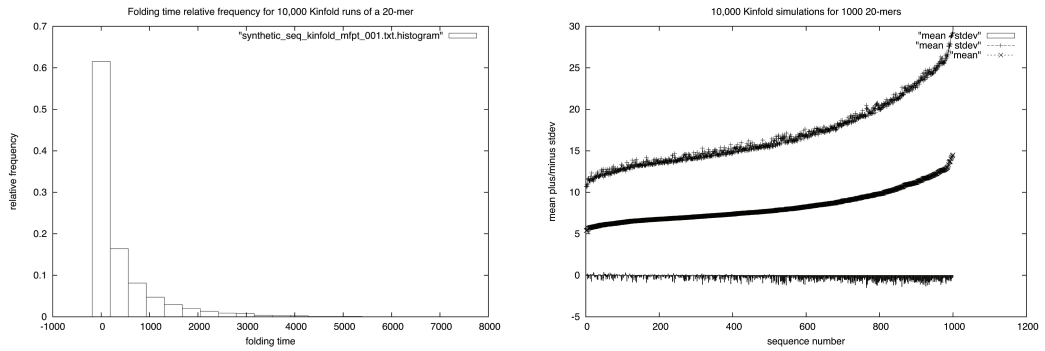

FIGURE 5.8: *(Left)* Histogram of `Kinfold` folding times for 20-mer `CCGAUUGGCGAAAGGCCACC`. The mean [resp. standard deviation] of 10,000 runs of `Kinfold` for this 20-mer is 538.37 [resp. 755.65]. Note the close fit to the exponential distribution, *(Right)* Mean minus standard deviation ($\mu - \sigma$), 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.

5.5.1 Pearson correlation coefficients for various kinetics packages

In this section, we display the correlation between *(1)* the *gold standard* method `RNAmfpt`, both with and without the Hastings modification using equations (5.17) and (5.18), *(2)* the *platinum standard* method `Equilibrium`, *(3)* the *silver standard* method `Kinfold`, *(4)* `FFTmfpt` with and without the Hastings modification using equations (5.23) and (5.24), *(5)* `FFTeq` which computes equilibrium time for the 2D-grid, *(6)* `RNA2Dfold` with and without the Hastings modification using equations (5.23) and (5.24). Correlations with [resp. without] the Hastings modification are summarized in the lower [resp. upper] triangular portion of Table 5.1. It is clear that correlations between the mathematically exact methods `RNAmfpt`, `RNAeq`, and approximation methods `Kinfold`, `FFTmfpt`, `FFTeq`, `RNA2Dfold` are improved when using the Hastings correction.

Figures 5.9, 5.10, 5.11 depict scatterplots for kinetics obtained by some of the algorithms above. The left panel of Figure 5.9 shows a scatter plots for gold standard `RNAmfpt` versus platinum standard `RNAeq`, with correlation value 0.5652. The right panel of the same figure shows a scatter plot for `Kinfold` versus `RNAeq`, 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 `RNAmfpt` or `RNAeq`, 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 5.10 shows the scatter plot for `RNAmfpt` versus `Kinfold`, with correlation 0.7933, and the right panel shows the scatter plot for `RNAmfpt` versus `FFTmfpt`, with correlation 0.6035. Figure 5.11 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, accurate kinetics estimates from `Kinfold` require prohibitively large computational time—indeed, in Wolfinger et al. [2004] 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 `RNAmfpt` 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.

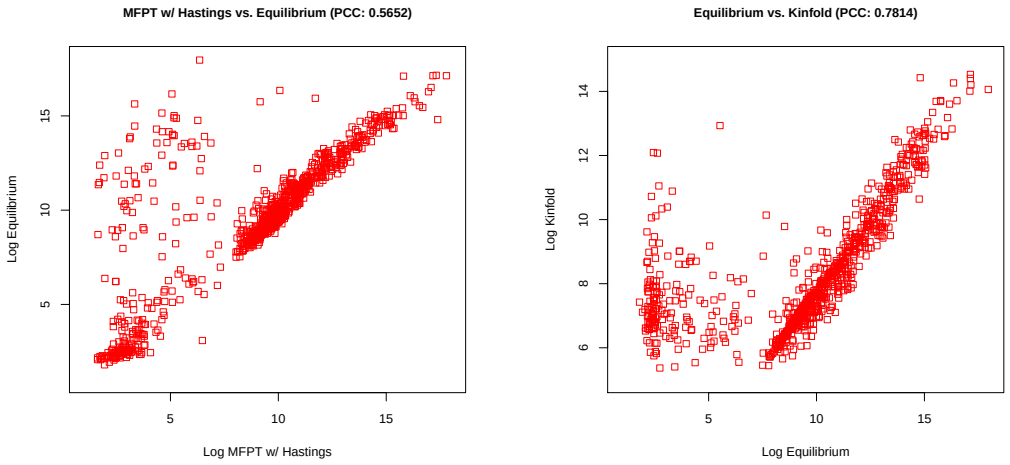

FIGURE 5.9: Scatter plots of the natural logarithm of times from `RNAmfpt` versus `RNAeq` *(Left)* and for `Kinfold` versus `RNAeq` *(Right)*.

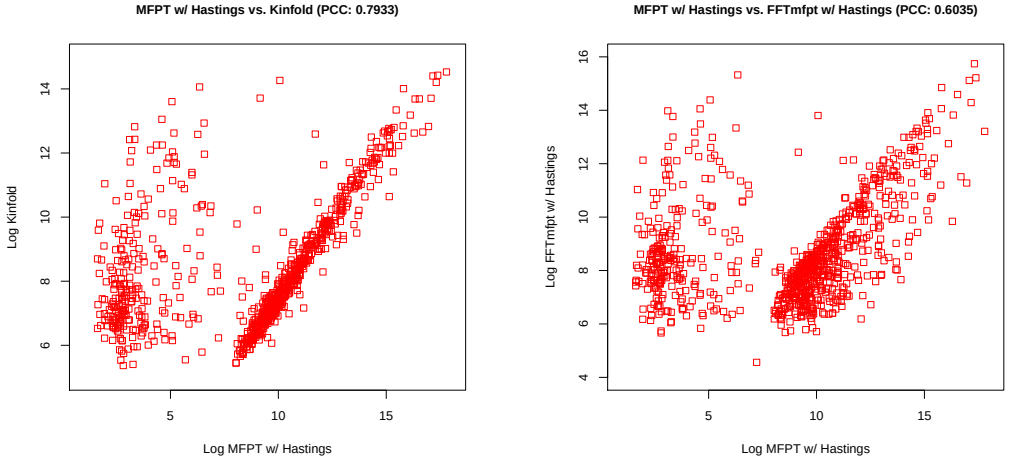

FIGURE 5.10: Scatter plots of the natural logarithm of times from `RNAmfpt` versus `Kinfold` *(Left)* and for `RNAmfpt` versus `FFTmfpt` *(Right)*.

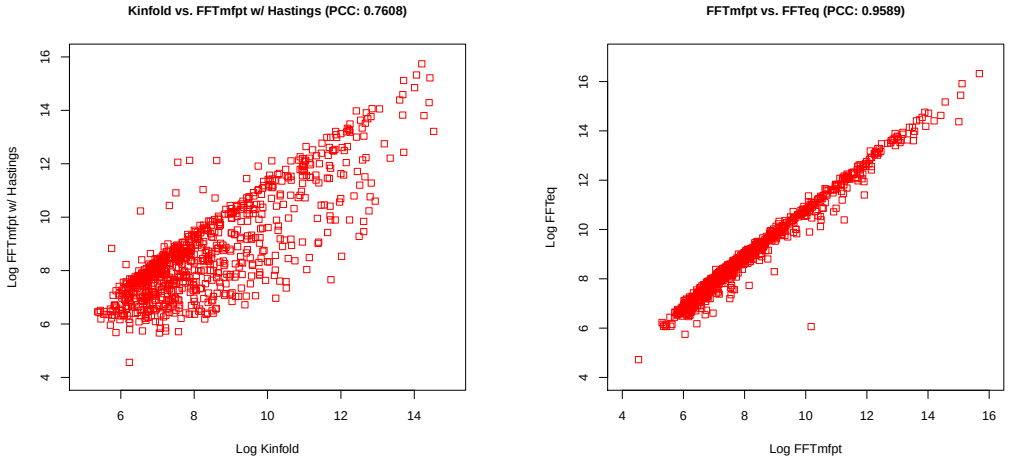

FIGURE 5.11: Scatter plots of the natural logarithm of times from `Kinfold` versus `FFTmfpt` *(Left)* and for `FFTmfpt` versus `FFTeq` *(Right)*.

Hastings (Yes\No)	RNAmfpt	RNAeq	Kinfold	FFTmfpt	RNA2Dfold	FFTbor	BarriersEq	FFTeq
RNAmfpt	1	0.5683	0.7945	0.5060	0.5110	0.5204	0.5280	0.4472
RNAeq	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
BarriersEq	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 5.1: 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: `RNAmfpt` (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), `RNAeq` (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), `BarriersEq` (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 `BarriersEq` using a Markov process derived from the energy landscape output by `FFTbor2D`).

Chapter 6

Discussion

Bibliography

P. P. Gardner, J. Daub, J. Tate, B. L. Moore, I. H. Osuch, S. Griffiths-Jones, R. D. Finn, E. P. Nawrocki, D. L. Kolbe, S. R. Eddy, and A. Bateman. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic. Acids. Res.*, 39(Database):D141–D145, January 2011.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, October 1990.

C. Flamm, W. Fontana, I.L. Hofacker, and P. Schuster. RNA folding at elementary step resolution. *RNA*, 6:325–338, 2000.

I. Dotu, J.A. Garcia-Martin, B.L. Slinger, V. Mechery, M.M. Meyer, and P. Clote. Complete RNA inverse folding: computational design of functional hammerhead ribozymes. *Nucleic Acids Res.*, 2014. in press.

E. Freyhult, V. Moulton, and P. Gardner. Predicting RNA structure using mutual information. *Appl. Bioinformatics*, 4(1):53–59, 2005.

R. Nussinov and A. B. Jacobson. Fast algorithm for predicting the secondary structure of single stranded RNA. *Proceedings of the National Academy of Sciences, USA*, 77 (11):6309–6313, 1980.

N. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, 24:547–556, 2004.

T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Algorithms*. McGraw-Hill, 1990. 1028 pages.

P. G. Wolynes. Energy landscapes and solved protein-folding problems. *Philos. Transact. A. Math. Phys. Eng. Sci.*, 363(1827):453–464, February 2005.

J. D. Bryngelson, J. N. Onuchic, N. D. Socci, and P. G. Wolynes. Funnels, pathways, and the energy landscape of protein folding: a synthesis. *Proteins.*, 21(3):167–195, March 1995.

S.F. Altschul and B.W. Erikson. Significance of nucleotide sequence alignments: A method for random sequence permutation that preserves dinucleotide and codon usage. *Mol. Biol. Evol*, 2(6):526–538, 1985.

E. Freyhult, V. Moulton, and P. Clote. Boltzmann probability of RNA structural neighbors and riboswitch detection. *Bioinformatics*, 23(16):2054–2062, August 2007.

T. Xia, Jr. J. SantaLucia, M.E. Burkard, R. Kierzek, S.J. Schroeder, X. Jiao, C. Cox, and D.H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37:14719–35, 1999.

R. Lorenz, C. Flamm, and I.L. Hofacker. 2D projections of RNA folding landscapes. In I. Grosse, S. Neumann, S. Posch, F. Schreiber, and P.F. Stadler, editors, *German Conference on Bioinformatics 2009*, volume 157 of *Lecture Notes in Informatics*, pages 11–20, 2009.

J.S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.

C.D. Meyer. The role of the group inverse in the theory of finite Markov chains. *SIAM Rev.*, 17(46):443–464, 1975.

P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction.* John Wiley & Sons, 2000. 279 pages.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman & Hall/CRC Press, 1995.

C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181: 223–230, 1973.

Joel N. Franklin. *Matrix Theory.* Dover Publications, Mineola, New York, 2000. 292 pages.

D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comp Phys*, 22(403):403–434, 1976.

C. Flamm. *Kinetic Folding of RNA*. PhD thesis, University of Vienna, 1998. Department of Chemistry.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

R. Lorenz, S. H. Bernhart, C. Honer Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. Viennarna Package 2.0. *Algorithms. Mol. Biol.*, 6:26, 2011.

E. Senter, S. Sheikh, I. Dotu, Y. Ponty, and P. Clote. Using the fast fourier transform to accelerate the computational search for RNA conformational switches. *PLoS. One.*, 7 (12):e50506, 2012.

S. Wuchty, W. Fontana, I. L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49:145–165, 1999.

M. Wolfinger, W.A. Svrcek-Seiler1, C. Flamm, and P.F. Stadler. Efficient computation of RNA folding dynamics. *J Phys. A: Math. Gen.*, 37:4731–4741, 2004.