On the Use of Coarse-Grained Thermodynamic Landscapes to Efficiently Estimate Folding Kinetics for RNA Molecules

Author: Evan Andrew Senter

Persistent link: http://hdl.handle.net/2345/bc-ir:104881

This work is posted on eScholarship@BC,
Boston College University Libraries.

Boston College

Doctoral Thesis

On the Use of Coarse-Grained Thermodynamic Landscapes to Efficiently Estimate Folding Kinetics for RNA Molecules

---

Author:
Evan Andrew Senter

Supervisor:
Dr. Peter Clote

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

in the

Clote Lab
Department of Biology
The Graduate School of Arts and Sciences

December 2015

BOSTON COLLEGE

Abstract

DR. PETER CLOTE
Department of Biology
The Graduate School of Arts and Sciences

Doctor of Philosophy

ON THE USE OF COARSE-GRAINED THERMODYNAMIC LANDSCAPES TO EFFICIENTLY
ESTIMATE FOLDING KINETICS FOR RNA MOLECULES

by EVAN ANDREW SENTER

RNA folding pathways play an important role in various biological processes, such as *1)* the conformational switch in spliced leader RNA from *Leptomonas collosoma*, which controls trans-splicing of a portion of the 5' exon, and *2)* riboswitches–portions of the 5' untranslated region of mRNA that regulate genes by allostery. Since RNA folding pathways are determined by the thermodynamic landscape, we have developed a number of novel algorithms—including `FFTbor` and `FFTbor2D`—which efficiently compute the coarse-grained energy landscape for a given RNA sequence. These energy landscapes can then be used to produce a model for RNA folding kinetics that can compute both the mean first passage time (MFPT) and equilibrium time in a deterministic and efficient manner, using a new software package we call `Hermes`. The speed of the software provided within `Hermes`—namely `FFTmfpt` and `FFTeq`—present what we believe to be the first suite of kinetic analysis tools for RNA sequences that are suitable for high throughput usage, something we believe to be of interest in the field of synthetic design.

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Peter Clote, for the countless hours

spent explaining the nuances of the computational RNA field. With all the guidance and sup-

port he provided over my graduate school career, I could not have asked for a better advisor.

Additionally, I'd like to thank my family and friends for their support and encouragement, and

the members of my thesis committee for their insights along the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduced in 1958, the central dogma of biology has been an excellent model for the biological flow of information, much as Newtonian classical mechanics stood the test of time for over 200 years. But just as Einstein's revolutionary principle of relativity has upended our understanding of space in a way unheard of since Copernicus, recent research has gone to confirm that for all our scientific progress, the cell still holds fundamental mysteries, and even the central dogma isn't sacred. Even Francis Crick himself indicated in [1] that ribonucleic acids (RNAs) likely had a role beyond the traditional *messenger* intermediary between DNA and proteins, as evidenced by viral RNAs [2]. Though recent research continues to suggest that the genome is pervasively transcribed, current estimates indicate that only 1.2% of the mammalian genome constitutes protein-coding sequences [3, 4, 5]. The study of the transcriptome has led to the identification of a wide variety of non-coding RNAs (ncRNAs) that highlight the diversity of roles for which RNA can be put to use [6]. Now understood to be much more than the intermediary step between DNA and proteins, RNAs have been implicated in a variety of regulatory and enzymatic activities, including gene knockdown and silencing [7, 8, 9, 10], transcriptional

and translational regulation [11, 12], intronic splicing [13, 14], cite-specific cleavage [15], and more. A prevailing theory now suggests that self-replicating RNA molecules were the predecessors to all life on Earth—the RNA world hypothesis [16]. As our appreciation of RNA diversity has increased, significant effort has been put forth by the scientific community to understand and characterize the properties of these molecules.

Unlike DNA, RNA is generally single stranded and thus able to interact with itself to form interesting shapes with various functional characteristics, akin to proteins. RNA is a polymer composed of four monomer *building blocks*: the purines adenine (A) and guanine (G); and the pyrimidines cytosine (C) and uracil (U). These nucleotides can form planar base pairs composed of energetically favorable hydrogen bonds, the stacking of which produces a stable *helix* structure [17]. There are only a select set of possible commonly occuring base pairs; the Watson-Crick pairs (A-U, G-C) or the G-U wobble pair. Given an arbitrary RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, where $s_i \in \{A, U, G, C\}$, we can define a secondary structure $\mathcal{S}$ for $\mathbf{s}$ as the set of index tuples indicating those bases involved in a base pair within $\mathbf{s}$. Again like proteins, RNA molecules tend to fold into a 'native' conformation, usually that which minimizes free energy. While protein folding is predominantly motivated by hydrophobic interactions, RNA structure is driven by stacking base pair interactions, and therefore secondary structure tends to be a much better predictor for the function of the molecule in question than is the case with proteins, whose function is largely determined by 3D 'tertiary' structure.

From a computational perspective, the history of RNA folding is far too long for proper treatment within this introduction. Instead we will touch upon just some of the major milestones to get a flavor for what progress has been made. In 1960, the first modern model of RNA secondary structure was presented [18], followed by Michael Waterman's graph-theoretic model

of single stranded nucleic acids such as RNA in 1978 [19]. This was followed in 1980 by the

work of Ruth Nussinov and Ann Jacobson, who together presented an algorithm for deter-

mining the maximally matching secondary structure $\mathcal{S}$ for a given RNA sequence [20], using

dynamic programming [21]. In the following years Michael Zuker and Patrick Stiegler de-

veloped an algorithm and accompanying software for the minimum free energy formulation

of the problem [22, 23]. In 1990, John McCaskill showed how dynamic programming could be

used to compute the partition function for an RNA molecule, and even compute the probability

that an arbitrary base is bound [24]. Alongside these early developments, more robust energy

models were experimentally derived [25, 26], further improving the accuracy of computational

models.

Fast-forwarding to today, there is now a huge collection of software aimed at computing var-

ious properties of RNAs, be it folding, inverse folding, kinetics, design, and more. The work

that we present here intends to be a contribution to the diverse toolset that researchers have

at their disposal for the analysis and design of both existing and novel RNA sequences.

1.1   Thesis Content

The work of this thesis is based on the following four journal articles, alongside unpublished

data and observations. The journal articles constituting the primary body of research include:

- Senter, E., Sheikh, S., Dotu, I., Ponty, Y., & Clote, P. (2012). Using the Fast Fourier Trans-
  form to Accelerate the Computational Search for RNA Conformational Switches. PloS
  One, 7(12), e50506. `http://doi.org/10.1371/journal.pone.0050506`

- Ding, Y., Lorenz, W. A., Dotu, I., Senter, E., & Clote, P. (2014). Computing the Probability
  of RNA Hairpin and Multiloop Formation. Journal of Computational Biology : a Journal
  of Computational Molecular Cell Biology, 21(3), 201–218. `http://doi.org/10.1089/`
  `cmb.2013.0148`

- Senter, E., Dotu, I., & Clote, P. (2014). RNA folding pathways and kinetics using 2D energy landscapes. Journal of Mathematical Biology, 70(1-2), 173–196. `http://doi.org/10.1007/s00285-014-0760-4`

- Senter, E., & Clote, P. (2015). Fast, Approximate Kinetics of RNA Folding. Journal of Computational Biology : a Journal of Computational Molecular Cell Biology, 22(2), 124–144. `http://doi.org/10.1089/cmb.2014.0193`

Text, figures, and tables from these papers are used throughout this thesis without additional notice.

1.2   Thesis Organization

The remainder of this thesis is organized in the following fashion. We begin in Chapter 2 with the presentation of `Ribofinder`, a pipeline of software intended to facilitate the detection of full riboswitch sequences alongside their corresponding 'on' and 'off' structures in genomic data. In Chapter 3 we introduce the program `FFTbor`, which computes—for each integer k— the Boltzmann probability p_k of the subensemble of structures whose base pair distance to an input reference structure $\mathcal{S}$ is k. In Chapter 4 we extend this idea to simultaneously consider two reference structures $\mathcal{S}, \mathcal{T}$ and produce as a result the coarse-grained 2D energy landscape where—for each integer pair x,y—we compute the Boltzmann probability $p_{x,y}$ of those structures whose base pair distance from $\mathcal{S}$ [resp. $\mathcal{T}$] is x [resp. y]. This program—`FFTbor2D`— allows for the efficient approximation of kinetic characteristics of RNA molecules, presented in Chapter 5 through the software package `Hermes`. Finally in Chapter 6 we conclude with a summary of this work as a whole, and consider its place in the greater ecosystem of computational RNA tools.

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` [27, 28] and `TransTermHP` [29] 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` [30] 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` [31, 32] to measure the similarity between our candidate pool and a canonical guanine riboswitch well studied in the literature, the xanthine phosphoribosyltransferase (xpt) guanine 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 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.

Tools such as Riboswitch finder [33] and `RiboSW` [34] have used the conserved structural char-
acteristics of the aptamer as search criteria with promising results, while the webserver `RibEx`
relies solely on sequence conservation to detect putative aptamer domains in genomic data
[35]. Meanwhile other groups have used covariance model (CM)-based approaches to find ap-
tamer domains, most notably `CMfinder` [36].

Whereas there exists strong sequence and structural similarity within the aptamer of a ri-
boswitch 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 con-
tain 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 25th November, 2014 at 09:14), the RefSeq database hosted
by NCBI comprised 5,121 complete bacterial genomes with corresponding genomic annota-
tions. 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
1) identify a pool of candidate riboswitches across genomic data; *2*) perform a coarse-grained
filtering of the candidate pool based on structural characteristics; and finally *3*) fine-grained

curation of the candidates based on a collection of measures and pairwise similarity. Figure 2.1

outlines this approach as a flowchart.
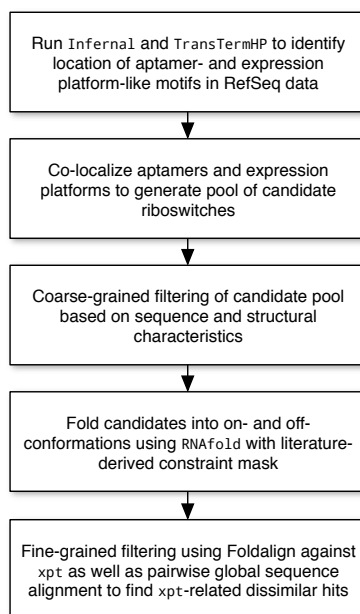
FIGURE 2.1: Outline of the approach for the `Ribofinder` pipeline.

In the following discussion, we describe the application of `Ribofinder` to identify unanno-

tated guanine 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 (downloaded on Tuesday 25th November, 2014 at 09:14)

contains 5,121 annotated bacterial genomes across 2,732 different organisms, totaling over $9.5 \times$

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` [27, 28] 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 guanine riboswitches exclusively we require the presence of a cytidine at this residue (Figure 2.2). In total, using `Infernal` with these additional filters yields 1,280 guanine aptamers across 555 unique organisms.

2.3.1.2 Detecting expression platforms with `TransTermHP`

`TransTermHP` [29] detects rho-independent terminators in bacterial genomes in a context-sensitive fashion by leveraging the protein annotations available in NCBI Protein Table (PTT) data. These terminator sequences canonically have a stable hairpin loop structure immediately
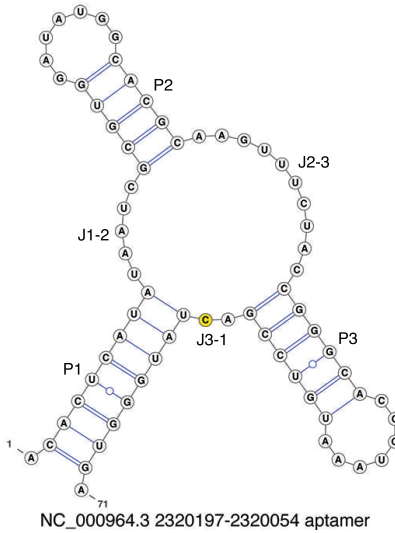
FIGURE 2.2: Diagram of the aptamer portion of the *B. subtilis* xpt guanine riboswitch, with annotations to indicate the P1, P2, and P3 stems of the multiloop, junctions between the hairpins, and the ligand binding site (in yellow). This structural diagram was generated using VARNA [37].

preceding a run of 5+ uracil residues, the combination of which causes RNA polymerase 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.

Genomic size of bacterial NC entries in RefSeq ($\log_{10}$ scale, 11/25/2014 9:14AM)
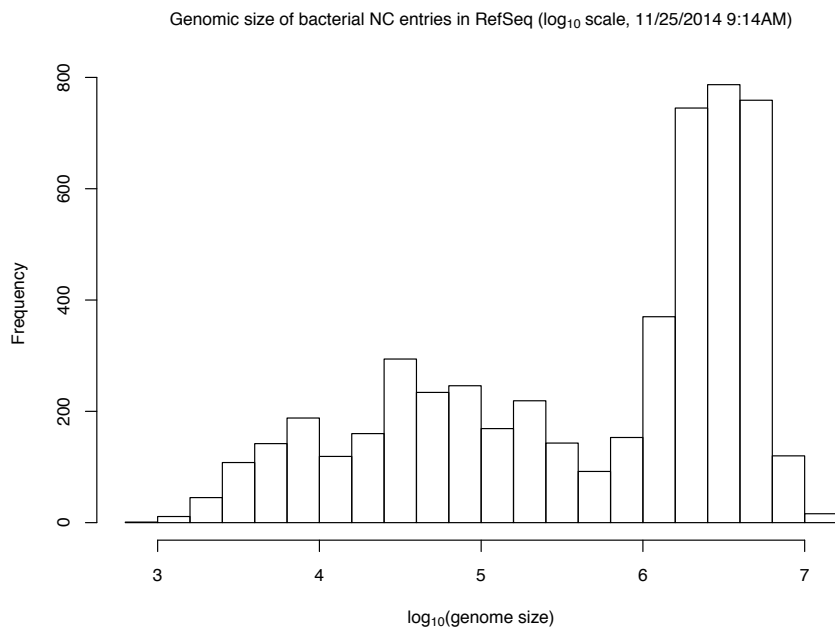
FIGURE 2.3: Histogram displaying the distribution of genome sizes across the RefSeq data analyzed, comprising 5,172 bacterial genomes. Genome size is shown using a $\log_{10}$ scale, and appears to have a bimodal distribution.

2.3.2   Step 2: Structural prediction

Until this point we have been focused on the generation of candidate sequences from our RefSeq dataset, without yet focusing on the specifics of underlying secondary structures for these candidates. In the following section, we explain how constraint folding is used to generate putative 'on' and 'off' conformations for each candidate.

2.3.2.1 Notation for representing abstract RNA shapes

Given an RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, where positions s_i are drawn from the collection of single-letter nucleotide codes, i.e. $s_i \in \{A, U, G, C\}$, 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 base pair. Given any two base pairs (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 the following, derived from [38]:

$$S \rightarrow \bullet \mid S \bullet \mid (S) \mid S(S) \tag{2.1}$$

The grammar from equation (2.1), where the minimum number of unpaired bases θ in a hairpin loop is taken to be 1 for expository clarity, 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 [39]. This is analogous to what the Giegerich lab refers to as their 'type 5' structural abstraction using the `RNAshapes` tool, and can be described using the following grammar [40], which generates all combinations of matched brackets:

$$S \rightarrow [T]S \mid [T]$$
$$T \rightarrow [T] \mid \epsilon \tag{2.2}$$

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) base pair. We use a level-order naming convention to refer to helices within the parse tree, whereby a position $\mathbf{p}_1$ references the first child of the root node, $\mathbf{p}_{1,2}$ references the second child of $\mathbf{p}_1$, and generally $\mathbf{p}_{i_1, i_2, \cdots, i_n}$ refers to the i_n^{th} child of $\mathbf{p}_{i_1, i_2, \cdots, i_{n-1}}$. To reference specific nucleotides in the context of their location relative to a helix, we use the opening and closing base pairs (i,j) and (k,l) as landmarks. Thus, $\mathbf{p}_1(l)$ is the index in $\mathcal{S}$ of the right-hand side closing base pair of $\mathbf{p}_1$. We use the notation $\mathbf{t}_i$ to refer to the subtree of $\mathcal{T}$ whose root is $\mathbf{p}_i$.

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 guanine riboswitch from *Bacillus subtilis* subsp. subtilis str. 168 (NC_000964.3 2320197–2320054) with corresponding gene 'off' structure as seen in Figure 2.4.
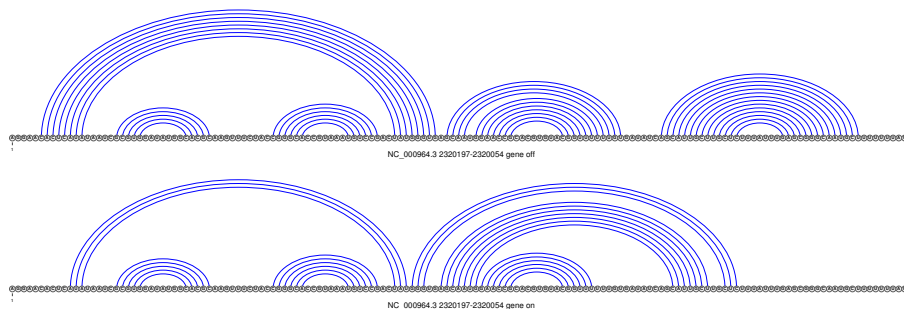
FIGURE 2.4: *(Top)* The xanthine phosphoribosyltransferase (xpt) guanine riboswitch from *B. subtilis* subsp. subtilis str. 168 (NC_000964.3 2320197–2320054), and corresponding gene 'off' structure derived from inline probing studies both in the presence and absence of guanine [41]. *(Bottom)* The experimentally derived gene 'on' structure for *B. subtilis* xpt. These structural diagrams were generated using VARNA [37].

The `RNAshapes` [42] 'type 5' representation for this structure is `[[][]][][]` (note the coalesced left bulge in the hairpin immediately downstream the closing multiloop stem, at helix 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 $p_1(l-1) = $ 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 guanine 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 $\mathcal{S}_{off}$ for a guanine riboswitch is the easier of the two to find computationally, since the terminator stem is exceptionally thermodynamically stable. In the gene 'on' conformation $\mathcal{S}_{on}$, the P1 stem of the multiloop partially dissociates and an anti-terminator stem forms between the region immediately 3' of the P1 stem and what was the left-hand side of the terminator stem. 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:

Structural constraints for both conformations of the guanine riboswitch aptamer:

1. Prohibit base pairing upstream of $\mathbf{p}_1(i)$ and downstream of $\mathbf{p}_2(l)$.

   Do not permit any possible disruptive pairing interactions 5' of the aptamer or 3' of the terminator stem.

2. Force base pairs and unpaired regions in $\mathbf{t}_1$, with the exception of $\mathbf{p}_1$.

   Since the aptamer structure is well conserved and we have the `Infernal`*-provided alignment with the covariance model, force this structure to form as aligned.*

3. Explicitly prohibit formation of $\mathbf{p}_1$ stem, which closes the three-way junction.

   The only exception to above is the closing of the P1 multiloop stem. In our experience, since `RNAfold` *uses soft constraints (meaning that constrained base pairs are only allowed to pair with each other or not at all), in practice we rarely see the P1 stem form as we would like. Instead, restrict it from forming at all, so that it can be added in after the fact without disrupting any other base pairs.*

Constraints exclusive to the gene 'off' structure:

1. Force base pairs and unpaired regions in $\mathbf{t}_2$.

   This simply forces the formation of the terminator stem, as predicted by `TransTermHP`*.*

Constraints exclusive to the gene 'on' structure:

1. Require m nucleotides starting from $\mathbf{p}_1(l + 3)$ to pair to the right, where $m = length(\mathbf{p}_2)$, and require the left-hand side of the $\mathbf{p}_2$ helix to pair to the left.

The formation of the anti-terminator stem involves the partial disruption of the p_1 stem. Though there is no consensus for the length of the anti-terminator stem, experimental data suggests that the left side of the terminator stem ($p_2(i)$–$p_2(k)$) alternatively base pairs to the left, thus forming the anti-terminator hairpin and permitting transcription to proceed [43].

2. Disallow pairing downstream of $p_2(j)$.

   Avoid disruptive pairing downstream of the newly formed anti-terminator stem.

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 [44]. We choose to disable dangles (`-d 0`) and use the Turner 1999 energy model (`-P rna_turner1999.par`) based on visual inspection of the structures output by `RNAfold` with constraints—these flags appear to yield conformations most frequently consistent with the known structures for the *B. subtilis* xpt guanine riboswitch. Experimental evidence using inline probing and crystallographic analysis suggests that the 'on' conformation of the guanine riboswitch has a reduced P1 stem length of 3 base pairs [45, 46]; 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 base pairs] in the gene 'off' [resp. gene 'on'] structure.

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. Our original approach involved using `FoldAlign` [31, 32] alongside the `needleall` tool from EMBOSS [47], to simultaneously select sequences which closely approximate the more thermodynamically stable gene 'off' conformation of the experimentally known *B. subtilis* xpt guanine riboswitch, while minimizing sequence similarity between candidates selected for experimental validation. Due to expense, we elected to instead choose a small number ($n = 2$) of organisms easily available which had multiple promising hits as our experimental candidate pool.

In Figure 2.5, we display a histogram of scores produced by `FoldAlign` for the 503 candidates, when aligned with the *B. subtilis* xpt guanine riboswitch. `FoldAlign` is based off a simplification of Sankoff's algorithm [48], and is a dynamic programming algorithm for simultaneous folding and alignment that runs in $O(n^4)$ time. Because three of the sequences from our pool of 503 candidates have no global alignment with the *B. subtilis* xpt sequence, we have pruned them from our dataset and only consider those remaining 500 sequences for which `FoldAlign` scores are produced. The `FoldAlign` scores produce have a mean of 153.798 with a minimum [resp. maximum] score of -2698 [resp. 1908]. Running `FoldAlign` with the *B. subtilis* xpt sequence aligned with itself produces a theoretical maximum score of 2419.

Of these 500 sequences, 335 have a `FoldAlign` score $s > 0$, representing 227 unique accession numbers, with a distribution of candidates per accession number as shown in Figure 2.6. From the perspective of experimental validation, we have tried to maximize the chance of success per organism by selecting those having multiple candidates within the same genome. Only 25 of the candidate organisms have more than three hits within their genome (only two have five hits). Our approach for selecting the initial two organisms for experimental validation was to take this pool of 25 organisms, sort by descending average score s, and select the first two
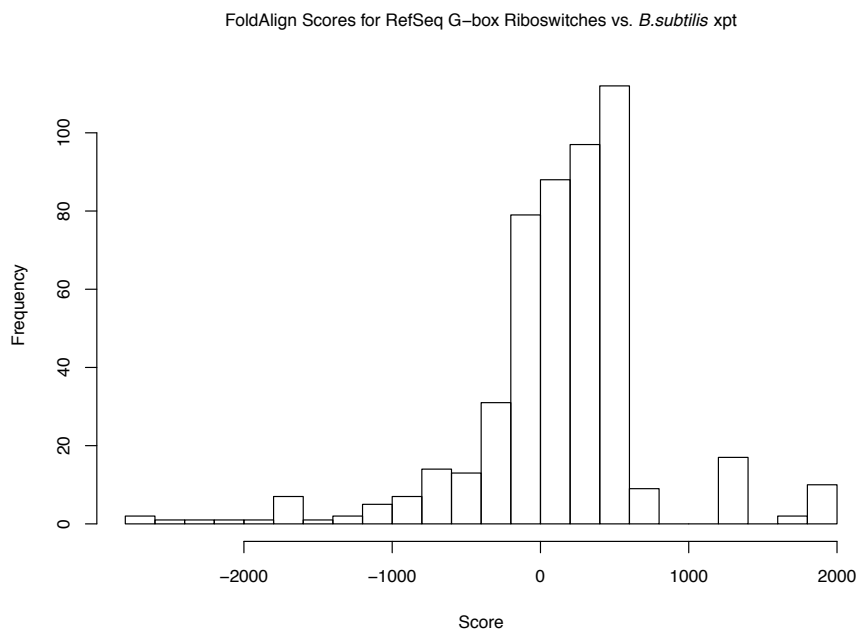
FIGURE 2.5: Histogram displaying the distribution of scores produced by `FoldAlign` 2.1.1 using flags `-global -summary -format commandline` when folding each of the 503 candidates against the *B. subtilis* xpt sequence NC_000964.3 2320197– 2320054. Three of the sequences run against `FoldAlign` (NC_010674.1 1516712– 1516868, NC_010723.1 1487041–1487197, and NC_020291.1 4599412–4599258) have no global alignment with the *B. subtilis* xpt sequence, and thus the histogram represents 500 of the original 503 sequences.

which are available via DSMZ (`https://www.dsmz.de/`), the warehouse for microorganisms used by our collaborators. Prof. Dr. Mario Mörl at Universität Leipzig is presently overseeing Dr. Regula Aregger, a post-doc who is using the SHAPE protocol [49] to validate the computationally predicted structure of these candidates.

Proceeding in this fashion, we have selected *B. megaterium* QM B1551 (NC_014019.1, DSM1804) and *B. megaterium* DSM319 (NC_014103.1, DSM319) for initial validation. These organisms have four candidate guanine riboswitches each, outlined in Table 2.1.
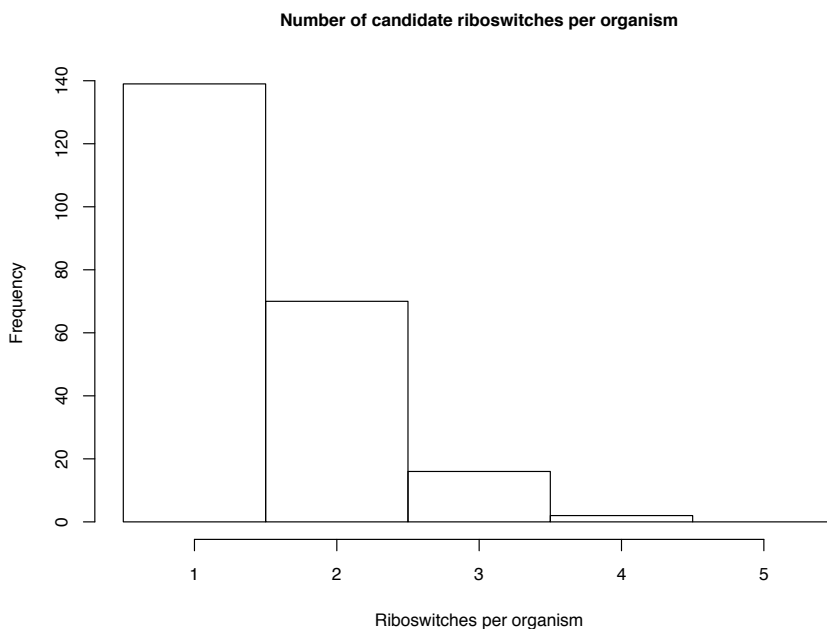
FIGURE 2.6: Histogram displaying the number of candidate riboswitches we observe per organism. From this data, it is clear that the majority of organisms have only one putative riboswitch, however two organisms have five candidates each: *Clostridium botulinum* B str. Eklund 17B (NC_010674.1) and *Clostridium botulinum* E3 str. Alaska E43 (NC_010723.1)

Downstream gene function	*B. megaterium* QM B1551	*B. megaterium* DSM319
xpt	1427313–1427501	1413696–1413884
GMP synthase	231630–231806	230059–230235
guanine permease	233482–233680	231911–232108
N5-carboxyaminoimidazole	240759–240970	239188–239400

TABLE 2.1: The genomic coordinates for the four candidate guanine riboswitches in both *B. megaterium* QM B1551 and *B. megaterium* DSM319. Note that the guanine riboswitches are located upstream of the same genes, and that these two strains of *B. megaterium* are highly similar. These structures are pictured in Section 2.5, plotted using VARNA [37].

2.4 Extending beyond guanine riboswitches

We believe that the `Ribofinder` pipeline allows for the detection of both structural conformations of riboswitches beyond the guanine riboswitch. The investigation of adenine-sensitive

purine riboswitches is a small extension of the existing implementation. As indicated in Section 2.3.1.1, adenine riboswitches have a complimentary uradine residue at the ligand binding site in the J3–1 junction within the aptamer. Beyond differences in ligand specificity, the adenine riboswitch anti-terminator stem is incorporated into the aptamer structure itself, and thus stabilized with the base pairing of the adenine ligand. As a result, the adenine riboswitch permits transcription when bound, unlike the guanine riboswitch. As a result of the extensive overlap between the anti-terminator stem and adenine riboswitch aptamer, the formation of the terminator stem completely dissociates both the P3 and P1 stems [50].

From a computational perspective, these changes are simple to handle within the `Ribofinder` pipeline, and provide some indication to how we believe the framework could be more generally applied in the future. Rather than filter for the discriminatory cytidine residue in the riboswitch aptamer (Section 2.3.1.1) we can only select those hits from `Infernal` having a uridine at the ligand binding site. Structural on and off conformations are known from experimental data for the *B. subtilis* ydhL gene [50] and can be used as templates for the constraint masks used in Section 2.3.2.

In general, we believe those riboswitches using rho-independent transcription termination as a mode of regulation—for which an aptamer alignment exists and some experimental knowledge of the expression platform's structural conformation is known—are well suited for more robust structural prediction using `Ribofinder`.

2.5 Guanine riboswitches for experimental validation
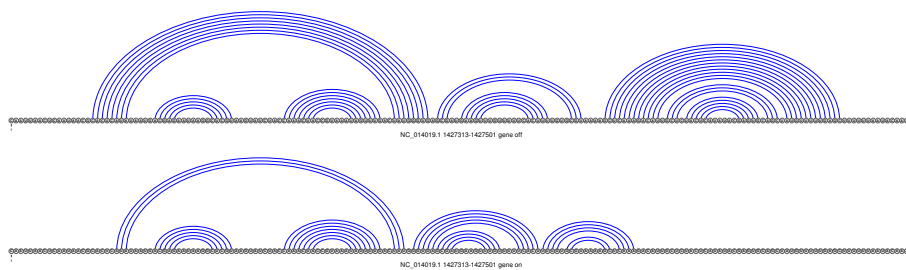
FIGURE 2.7: *Top:* the computationally predicted gene 'off' conformation of sequence NC_014019.1 1427313–1427501, using `RNAfold` from the ViennaRNA 2.1.8 suite, with dangles disabled and the Turner 1999 energies. This sequence is located upstream of the xpt gene in *B. megaterium* QM B1551. *Bottom:* the gene 'on' conformation.
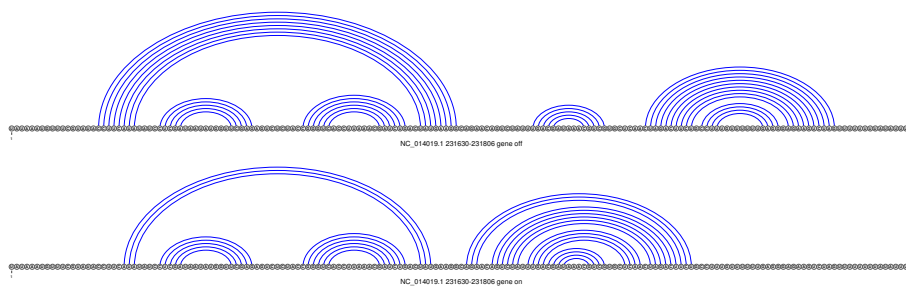
FIGURE 2.8: The computationally predicted riboswitch located upstream of the GMP synthase gene in *B. megaterium* QM B1551 (NC_014019.1 231630–231806). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
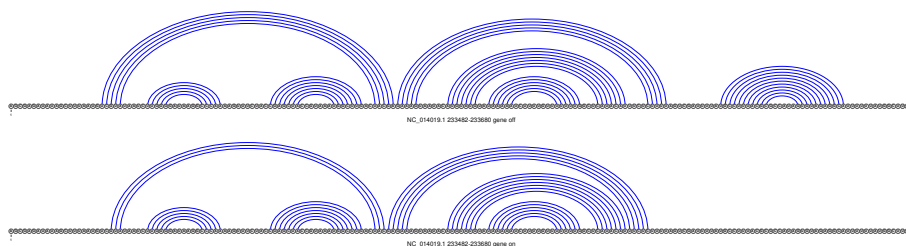
FIGURE 2.9: The computationally predicted riboswitch located upstream of the guanine permease gene in *B. megaterium* QM B1551 (NC_014019.1 233482–233680). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
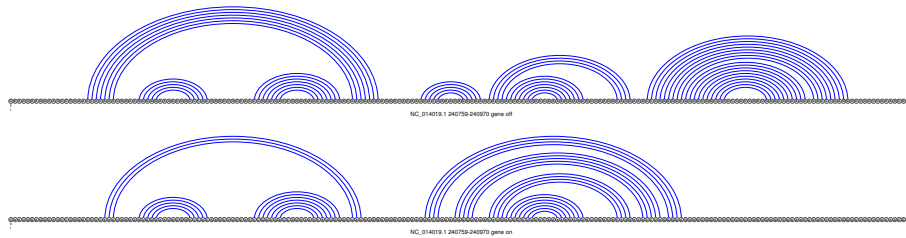
FIGURE 2.10: The computationally predicted riboswitch located upstream of the N5-carboxyaminoimidazole gene in *B. megaterium* QM B1551 (NC_014019.1 240759–240970). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
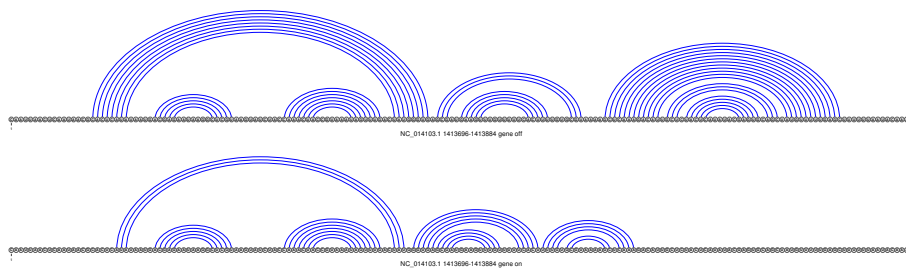
FIGURE 2.11: The computationally predicted riboswitch located upstream of the xpt gene in *B. megaterium* DSM319 (NC_014103.1 1413696–1413884). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
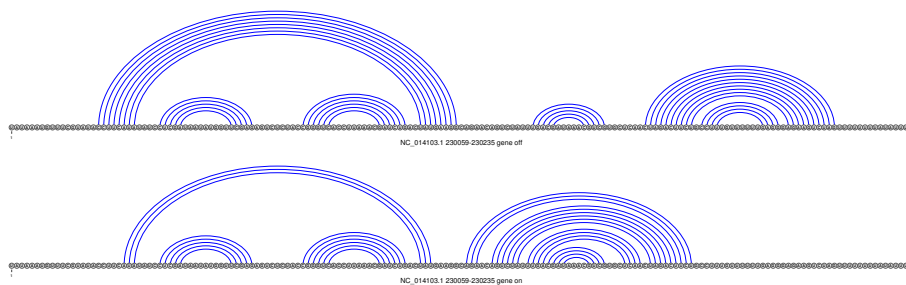
FIGURE 2.12: The computationally predicted riboswitch located upstream of the GMP synthase gene in *B. megaterium* DSM319 (NC_014103.1 230059–230235). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
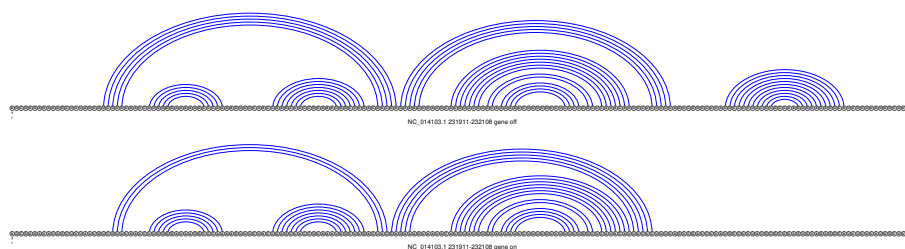
FIGURE 2.13: The computationally predicted riboswitch located upstream of the guanine permease gene in *B. megaterium* DSM319 (NC_014103.1 231911–232108). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.
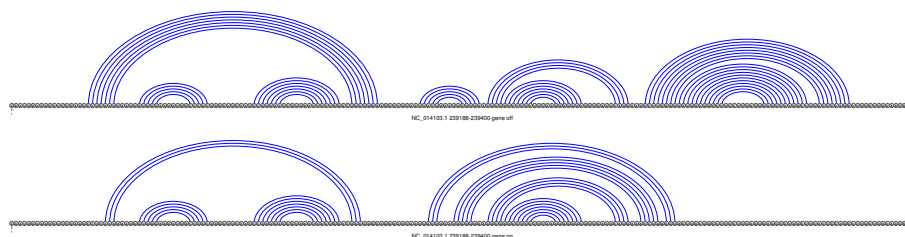
FIGURE 2.14: The computationally predicted riboswitch located upstream of the N5-carboxyaminoimidazole gene in *B. megaterium* DSM319 (NC_014103.1 239188–239400). *Top:* The gene 'off' conformation. *Bottom:* The gene 'on' conformation.

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 which, for a given input RNA sequence **s**, differ by k base pairs. By leverag-
ing 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 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 prob-
lem 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 land-scape is partitioned into discrete bins. We then develop the recursions for the parameterized partition function using the Nussinov-Jacobson energy model, which allows us to highlight the novel aspects of the algorithm. After developing the recursions, we indicate how they can be reformulated as a polynomial whose coefficients $z_k = \mathbf{Z}_{1,n}^k$. We then describe how the Fast Fourier Transform can be employed to efficiently compute the coefficients z_k, finishing our de-scription of the underlying algorithm. Then we proceed to present an application of `FFTbor`, in the area of RNA folding kinetics.

3.2 Background

In [51], a dynamic programming algorithm `RNAbor`—pronounced *RNA neighbor*—was devel-oped which simultaneously computes for each integer k, the Boltzmann probability $p_k = \frac{\mathbf{Z}^k}{\mathbf{Z}}$ of the subensemble of structures whose base pair distance to a given *initial*, or *reference*, structure $\mathcal{S}^*$ is k. [1] `RNAbor` stores the value of the (partial) partition functions $\mathbf{Z}_{i,j}^k$ for all $1 \leq i \leq j \leq n$ and $0 \leq k \leq n$, each of which requires quadratic time to compute. Thus it follows that `RNAbor` runs in time $O(n^5)$ and space $O(n^3)$, which severely limits its applicability to genomic anno-tation. This restriction is somewhat mitigated by the fact that in [52], we showed how to use sampling [53] to efficiently approximate `RNAbor` in cubic time $O(n^3)$ and quadratic space $O(n^2)$, *provided* that the starting structure $\mathcal{S}^*$ is the minimum free energy (MFE) structure. We expect that a more efficient version of `RNAbor` could be used in applications in genomics and synthetic

---

[1] As later explained, $\mathbf{Z}$ denotes the partition function, defined as the sum of all Boltzmann factors $\exp(-E(\mathcal{S})/RT)$, over all secondary structures $\mathcal{S}$ of a given RNA sequence, and R denotes the universal gas constant and T absolute temperature. Similarly $\mathbf{Z}^k$ denotes the sum of all Boltzmann factors of all structures $\mathcal{S}$, whose base pair distance to the initial structure $\mathcal{S}^*$ is exactly k.

biology, to detect potential conformational switches— RNA sequences containing two or more (distinct) metastable structures.

In this chapter, we describe a radically different algorithm, `FFTbor` [54], prounounced *FFT neighbor*, that uses polynomial interpolation to compute the coefficients $p_0, \ldots, p_{n-1}$ of the polynomial defined in equation (3.1), where p_k is defined by $p_k = \frac{Z^k}{Z}$. Due to severe numerical instability issues in both the Lagrange interpolation formula and in Gaussian elimination, we employ the Fast Fourier Transform (FFT) to compute the inverse Discrete Fourier Transform (DFT) 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). This gives rise to an improved version of `RNAbor`, denoted `FFTbor`, which runs in time $O(n^4)$ and space $O(n^2)$.

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 $p_k = \frac{Z^k}{Z}$. We employ the Fast Fourier Transform to compute the inverse Discrete Fourier Transform on values $y_0, \ldots, y_{n-1}$, where $y_k = p(\omega^k)$ and $\omega = e^{2\pi i/n}$ is the principal complex nth root of unity and $p(x)$ is defined in equation (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 $s_i \in \{A, U, G, C\}$, 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

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

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. As a result, sequences known to contain pseudoknots—such as RNase P [55]—are not well suited for `FFTbor`, and whenever possible investigators should take care to consider the likely structure for their sequences before proceeding with computational analysis.

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

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

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 [56] of $\mathcal{S}$, $R = 0.00198$ kcal/(mol·K) denotes the universal gas constant and T denotes absolute temperature. From this, it follows that the full partition function is defined as

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

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

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

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

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

By visualizing the probabilities p_k as a function of k, we generate a coarse-grained view of the one-dimensional energy landscape of **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 $\mathbf{Z}_{1,n}^{k}$

For the rest of the chapter, we consider both **s** as well as the secondary structure $\mathcal{S}^*$ on **s** to be fixed. We now recall the recursions from Freyhult et al. [57] to determine the partition function $\mathbf{Z}_{i,j}^{k}$ with respect to the Nussinov-Jacobson energy E_0 model [58], 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 [57] as well as our current software FFTbor, concern the Turner energy model (described in Section

A.1), consisting of free energy parameters for stacked bases, hairpins, bulges, internal loops and multiloops.

The base case for $Z_{i,j}^k$ is given by

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

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

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

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

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

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

since $b(r)$ is the number of base pairs that differ between $\mathcal{S}^*_{[i,j]}$ and a structure $\mathcal{T}_{[i,j]}$, due to the introduction of the base pair (r,j).

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

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

where coefficients $z_k = \mathbf{Z}^k_{1,n}$. 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 $z_n = 0$, 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 co-efficients z_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 $z_k = \mathbf{Z}^k_{1,n}$ 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 in-terpolation 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 [59], 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 $0 \le k < n$. 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) [60] where $\alpha_0, \ldots, \alpha_{n-1}$ are chosen to be complex nth roots of unity, $\alpha_k = e^{2\pi i k/n}$. One advantage of the FFT is that interpolation can be performed in $O(n \log n)$ time, rather than the cubic time required by using the Lagrange formula shown in equation (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

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

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

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

where

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

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

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

where

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

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. ana-

lytic] function, then $[x^k]F(x)$ denotes the coefficient of x^k [resp. the kth Taylor coefficient in

the Taylor expansion of $F(x)$]. For instance, in equation (3.1), $[x^k]p(x) = p_k$, and in equation

(3.12), $[x^k]\mathcal{Z}(x) = z_k$.

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

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

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:

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

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

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

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.

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

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 [60], 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 $j = 0, \ldots, n-1$.

Since we defined $\mathbf{Y}$ in equation (3.24) by $\mathbf{Y} = (y_0, \ldots, y_{n-1})^{\mathrm{T}}$, where $y_0 = \mathcal{Z}(\alpha_0), \ldots, y_{n-1} = \mathcal{Z}(\alpha_{n-1})$ and $\alpha_k = \omega^k e^{2\pi i k/n}$, it follows that the coefficients $z_k = \mathbf{Z}_{1,n}^k$ in the polynomial $\mathcal{Z}(x) = z_0 + z_1 x + \cdots + z_{n-1} x^{n-1}$ 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 $y_1 = \frac{\mathcal{Z}(\alpha_1)}{Z}, \ldots, y_n = \frac{\mathcal{Z}(\alpha_n)}{Z}$, 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.

Pseudocode for `FFTbor`

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

---

1 **function** FFTBOR($\mathbf{s}$, $\mathcal{S}^*$, m)

2      $n \leftarrow length(\mathbf{s})$

3   **for** $k \leftarrow 0, n-1$ **do**   ▷ Compute all complex nth roots of unity

4          $\omega_k \leftarrow \exp(\frac{2\pi i k}{n})$

5   **end for**

6   **for** $k \leftarrow 0, n-1$ **do**   ▷ Note that $\mathcal{Z}(\omega_0) = Z$

7          $y_k \leftarrow 10^m \cdot \frac{\mathcal{Z}(\omega_k)}{\mathcal{Z}(\omega_0)}$

8   **end for**

9   **for** $k \leftarrow 0, n-1$ **do**   ▷ Compute IDFT from equation (3.27)

10         $a_k \leftarrow \frac{1}{n} \sum_{j=0}^{n-1} y_j\, \omega^{-kj}$

11         $p_k \leftarrow 10^{-m} \cdot \lfloor a_k \rfloor$ ▷ Truncate to m significant digits

12   **end for**

13   **return** $p_0, \ldots, p_{n-1}$ ▷ Return all p_k for $0 \leq k < n$, from equation (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 $p_k = \frac{Z^k}{Z}$. This algorithm operates in $O(n^4)$ time and $O(n^2)$ space, a significant improvement over its predecessor `RNAbor`.

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 $\bar{z}$; i.e. if $z = a + bi$ where $a, b \in \mathbb{R}$ are real numbers and $i = \sqrt{-1}$, then $\bar{z} = a - bi$.

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

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

This comes from the well known fact that $\mathcal{Z}(\overline{\alpha}) = \overline{\mathcal{Z}(\alpha)}$ for any polynomial with real coefficients.

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 $p(k) = \frac{Z^k}{Z}$ 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 [61]. A hypothesis behind theoretical work in biomolecular folding theory in [62] 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 from *B. bacteriovorus* has EMBL accession code BX842649.1 277414–277318 and is composed of the 97 nt sequence ACCUGACGCUAGGGGUGUUGGUG AAUUCACCGACUGAGAAUAACCCUUUGAACCUGAUAGAGAUAAUGCUCGCGCAGGG AAGCAAGAAUAGAAAGAU. The second sequence from the marine metagenome has EMBL accession code AACY022101973.1 389–487 and is composed of the 99 nt sequence UAUAAG UCCAAGGGGUGCCAAUUGGCUGAGAUGGUUUUAACCAAUCCCUUUGAACCUGAUCC GGUUAAUACCGGCGUAGGAAUGGAUUUUCUCUACAGC. Rfam consensus and minimum free energy structures for both sequences are depicted in Figure 3.2. Despite the fact that there is no sequence similarity according to pairwise BLAST [63], 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
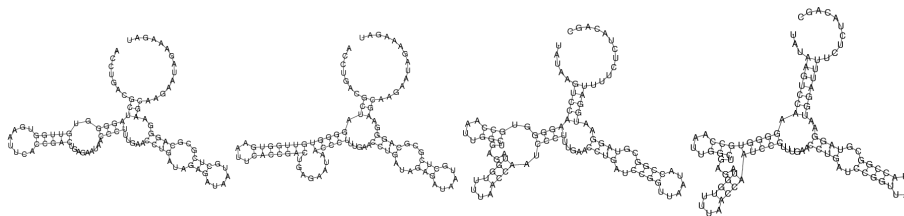
FIGURE 3.2: Rfam consensus structures (Rfam) and minimum free energy (MFE) secondary structures for two thiamine pyrophosphate (TPP) riboswitch aptamers, chosen at random from RF00059 Rfam family seed alignment [64]. Using pairwise BLAST [63], there is no sequence similarity, although the secondary structures are very similar, as shown in this figure. From left to right: *(A)* MFE structure for BX842649.1 277414–277318. *(B)* Rfam consensus structure for BX842649.1 277414–277318. *(C)* MFE structure for AACY022101973.1 389–487. *(D)* Rfam consensus structure for AACY022101973.1 389–487.

slow the kinetics of folding. In contrast, for the second sequence, AACY022101973.1 389–487, the `FFTbor` output suggests that there are no such competing low energy structures, hence the second sequence should fold more quickly than the first.

To test the hypothesis that folding is slower for rugged energy landscapes, we ran the kinetic folding software, `Kinfold` [65], 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—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,
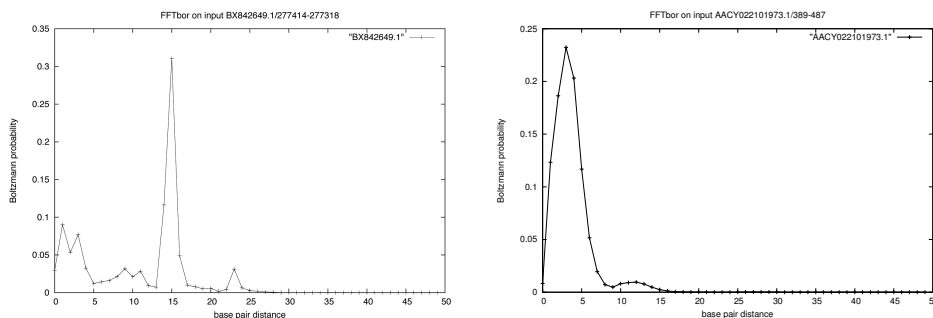
FIGURE 3.3: Output from `FFTbor` on two randomly selected thiamine pyrophosphate riboswitch (TPP) aptamers, taken from the Rfam database [64]. The x-axis represents base pair distance from the minimum free energy structure for each given sequence; the y-axis represents Boltzmann probabilities $p(k) = \frac{Z^k}{Z}$, where 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` [65] simulations clearly show that the second sequence folds at least four times more quickly than the first sequence. See section 3.6 for details.

we found a mean first passage time of 311,075.06 steps for this sequence. The second sequence, AACY022101973.1 389–487, converged within 500,000 steps in 29 out of 30 runs, and we found a mean first passage time of 61,575.69 steps for this sequence. From computational experiments of this type, it is suggestive that `FFTbor` may prove useful 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

$$E[\{d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) : \mathcal{S} \in \mathbb{S}(s_1, \ldots, s_n)\}] = \sum_{\mathcal{S}} P(\mathcal{S}) \cdot d_{\mathrm{BP}}(\mathcal{S}, \mathcal{S}^*) \qquad (3.29)$$

where $\mathbb{S}(s_1, \ldots, s_n)$ denotes the set of secondary structures for $\mathbf{s} = s_1, \ldots, s_n$, $P(\mathcal{S}) = \frac{\exp(-E(\mathcal{S})/RT)}{Z}$ 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 $p(k) = \frac{Z^k}{Z}$, 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 [64]. 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.

Surprisingly, we found that there is a significant correlation of 0.4847 with one-tailed p-value of 0.0002 between the standard deviation of the FFTbor output (when starting from the empty structure) and logarithm base 10 of the mean first passage time. 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. The standard deviation of this probability distribution corresponds to a notion of the width of the distribution. It is possible that those sequences having distributions tightly centered around the mean have faster folding times than those with a wider distribution due to other local minima causing the RNA to get trapped while folding.

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 from *B. bacteriovorus* and AACY022101973.1 389–487 from the marine metagenome, starting from the empty reference structure $\mathcal{S}^* = \varnothing$. 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 $\sigma_1 = 2.2253$ [resp. $\sigma_2 = 1.9857$], and the Kinfold MFPT is 311,075.06 [resp. 61,575.69] for the TPP riboswitch aptamer BX842649.1 277414–277318 [resp. AACY022101973.1 389–487]. This anecdotal evidence supports the hypothesis that small standard deviation in FFTbor distribution is correlated with fast folding.

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 [66], 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 BX842649.1 are respectively $\mu_1 = 19.93$ and $\sigma_1 = 2.88$, while those for randomized AACY022101973.1 are $\mu_2 = 24.39$

	μ	σ	σ/μ	n	MFE	$\log_{10}(\text{MFPT})$
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	
$\log_{10}(\text{MFPT})$	-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 [64]. 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 ($\log_{10}(\text{MFPT})$), 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 $\log_{10}(\text{MFPT})$ 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); and *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 -0.0363), although *vi)* MFE is correlated with MFPT (correlation 0.3990) — 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 0.0002.

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 AACY022101973.1, only 15 out of 30 runs converged within 500,000 steps, and
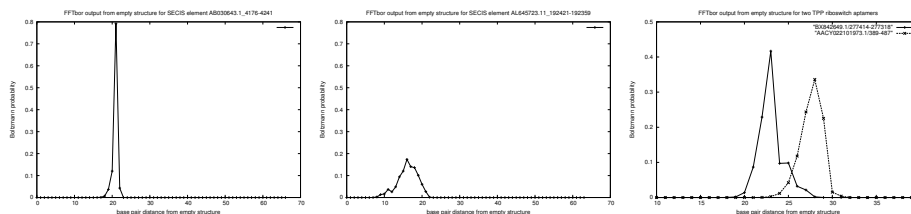
FIGURE 3.4: This figure represents the graphical output of `FFTbor`, when the empty structure is chosen as initial structure $\mathcal{S}^*$. The x-axis represents the number of base pairs per structure, taken over the ensemble of all secondary structures for the given RNA sequence; the y-axis represents Boltzmann probability $p(k) = \frac{\mathbf{Z}^k}{\mathbf{Z}}$, where $\mathbf{Z}$ is the partition function for all secondary structures having exactly k base pairs. *(Left)* For the *M. musculus* 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 0.7276, while the logarithm base 10 of the mean first passage time ($\log_{10}$(MFPT)) is 4.75. *(Center)* For the *M. musculus* seleno-cysteine (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 sec-ondary structures, is 2.6794, 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 out-put of `FFTbor` for two TPP riboswitch aptamers: the 97 nt sequence BX842649.1 277414–277318 from *B. bacteriovorus* and the 99 nt sequence AACY022101973.1 389–487 from the marine metagenome, 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 2.2253 [resp. 1.9857], and the `Kinfold` MFPT is 311,075.06 [resp. 61,575.69] for the TPP riboswitch aptamer BX842649.1 277414–277318 [resp. AACY022101973.1 389–487]. 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 Table 3.1.

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 anecdo-tal 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 $O(n^2)$. While `FFTbor` saves an order of magnitude in performance and memory, `RNAbor` has the benefit of producing suboptimal structures MFE_k for all $0 \leq k \leq n$, whose free energy is minimal across all structures having base pair distance k from input structures $\mathcal{S}^*$. 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` [51] 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.
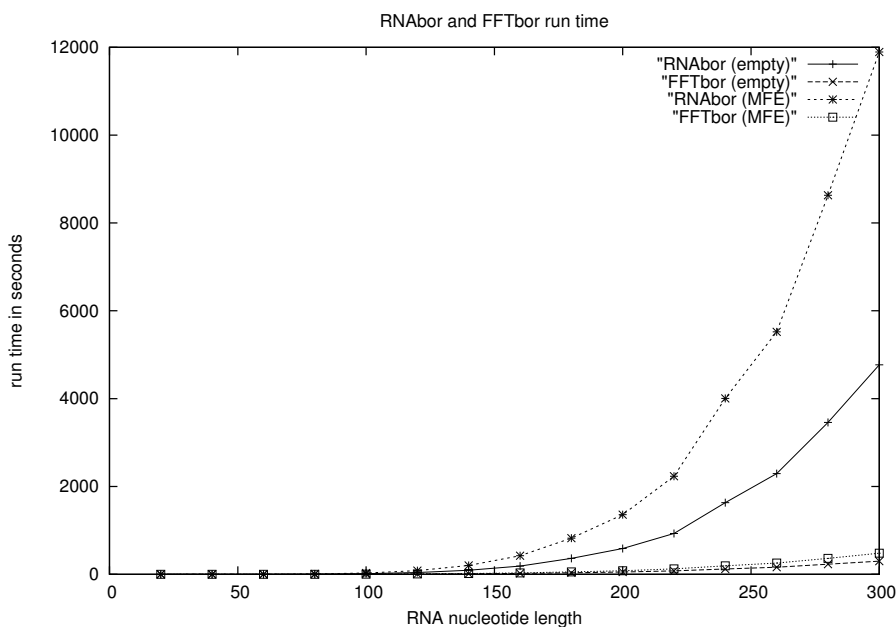
FIGURE 3.5: Run times in seconds for RNAbor and FFTbor, on random RNA of length 20,40,60,. . . ,300 in step size of 20 nt. Each algorithm was run with the empty initial structure $\mathcal{S}^*$, see rows RNAbor (empty), FFTbor (empty), and with the minimum free energy structure as the initial structure $\mathcal{S}^*$, 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 data in Table 3.2 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. Figure 3.6 presents similar data for running `FFTbor` on 2,3,6,4,12,15,20 cores.

n	Single core	Two cores
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

TABLE 3.2: Table showing parallel run times in seconds for `FFTbor`, using OpenMP—
`http://openmp.org/`. For each sequence length 200, . . . ,500, five random RNAs
were generated using equal probability for each nucleotide A,C,G,U. Run time in sec-
onds, plus or minus one standard deviation, are given for a 24-core AMD Opteron
6172 running at 2.10GHz with 64GB RAM, with only 1 [resp. 2] cores used.
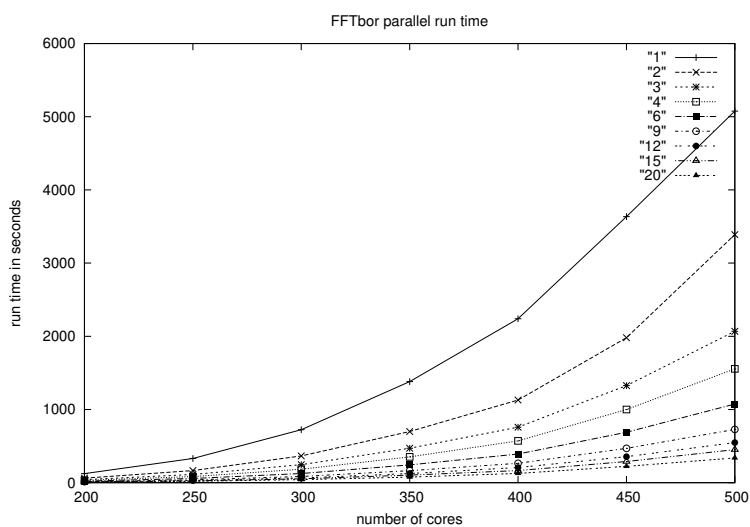
FIGURE 3.6: Graph showing parallel run time of `FFTbor` as a function of sequence
length, running on an AMD Opteron 6172 running at 2.10GHz with 64GB RAM, using
respectively 1,2,3,4,6,9,12,15,20 cores.

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 **s**. `FFTbor2D` computes the two-dimensional coarse-grained energy landscape for **s** given two compatible input secondary structures $\mathcal{A}$ and $\mathcal{B}$, where position (x,y) on the discrete energy landscape corresponds to the Boltzmann probability for those structures $\mathcal{S}$ which have $d_{\mathrm{BP}}(\mathcal{S}, \mathcal{A}) = x$ 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 improvements over the straight-forward use of the Fast Fourier Transform to compute $P(x)$, a parity condition and complex conjugates, which together 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

RNA folding pathways play an important role in biological processes. For instance, in the *hok/sok* (host-killing/suppression of killing) system [67], the transition between two metastable RNA structures determines the fate of a cell as follows. The *hok* gene of *E. coli* and other bacteria codes a small (52 amino acid) toxin causing irreversible damage to the cell membrane. It has been shown that *hok*-mRNA is constitutively expressed from a weak promoter, while the

rapidly degraded *sok*-RNA is constitutively expressed from a strong promoter. The *hok*-mRNA is initially inactive, since a foldback sequesters the Shine-Dalgarno sequence; however, slow exonucleolytic processing digests the last $\approx$ 40 nt of the 3′ end of *hok*-mRNA, transforming the molecule into its active form in which the Shine-Dalgarno sequence is no longer sequestered. If R1 plasmids of *E. coli* are present in sufficient copy number, then a portion of the 64 nt *sok*-RNA, which is complementary to *hok*-mRNA leader region, binds to the active conformation of *hok*-mRNA, thus causing degradation of the complex by RNase III [67]. If plasmids are not present in sufficient copy number, then the cell is killed by *hok* toxin, thus ensuring fitness of the daughter cells.

In the case of spliced leader (SL) RNA from certain trypanosomes and nematodes, a portion of the 5′ exon is donated to another mRNA by trans-splicing. Intermediate structures appear to be important in the process of splicing, as shown by LeCuyer and Crothers [68], who performed stopped-flow rapid-mixing and temperature-jump measurements of the kinetics for the structural transition between two low energy structures of SL RNA from *Leptomonas collosoma*. Conformational switches are thought not only to play a role in such trans-splicing, but as well in transcriptional and translational regulation, protein synthesis, and mRNA splicing.

For these reasons, substantial experimental and computational work has been done on folding pathways. In [69], `RNA2Dfold`—a dynamic programming method that operates in $O(n^7)$ time and $O(n^4)$ space—is presented, a generalization of `RNAbor` [51] that computes a 2D projection of the energy landscape, discretized by base pair distance to two input structures. Our work presented here, called `FFTbor2D`, produces a similar energy landscape in $O(n^5)$ time and $O(n^2)$ space—efficient enough for the analysis of folding pathways of large RNA sequences not tractable using `RNA2Dfold`.

4.3 Derivation of the `FFTbor2D` algorithm

For expository clarity, we describe `FFTbor2D` [70] and all recursions in terms of the Nussinov energy model [58] (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 [56], 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 chapter, 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 [71], Lorenz et al. generalized the dynamic programming recursions of our

earlier work [51], to yield recursions for the partition function $Z_{i,j}^{x,y}$ in equation (4.1). In the context of the Nussinov model, $Z_{i,j}^{x,y}$ is equal to

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

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

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

where $E(S)$ denotes the free energy of 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 z_{rn+s} of $\mathcal{Z}(x)$, for $0 \le r,s < n$. Due to technical difficulties concerning numerical robustness observered while working on the FFTbor software (Chapter 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

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

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 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 $1 \le i < j \le n$, let*

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

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

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

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

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

$\mathcal{B}_{[k+1,j-1]} \cup \{(k,j)\})$.

The proof is given in Section B.2.

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` [71]. 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 $0 \leq i \leq n^2 - 1$, we can compute and save only the values $\mathcal{Z}(\rho_0), \ldots, \mathcal{Z}(\rho_{n^2-1})$, each time re-using the $O(n^2)$ space for the next computation of $\mathcal{Z}(\rho_i)$. It follows that the computation resources used to determine the (column) vector

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

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

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

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 $p_{rn+s} = \frac{Z_{1,n}^{r,s}}{Z}$. 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 $y \in \{d_0 - x + 2k : k = 0, \ldots, x\}$.*

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

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

Pseudocode for `FFTbor2D`

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

---

1 **function** FFTBOR2D($\mathbf{s}, \mathcal{A}, \mathcal{B}, m$)

2      $n \leftarrow length(\mathbf{s})$

3   **for** $k \leftarrow 0, n^2 - 1$ **do**   ▷ Compute all complex n^2-roots of unity

4          $\omega_k \leftarrow \exp(\frac{2\pi i k}{n^2})$

5   **end for**

6   **for** $k \leftarrow 0, n^2 - 1$ **do**   ▷ Note that $\mathcal{Z}(\omega_0) = \mathbf{Z}$

7          $y_k \leftarrow 2^m \cdot \frac{\mathcal{Z}(\omega_k)}{\mathcal{Z}(\omega_0)}$

8   **end for**

9   **for** $k \leftarrow 0, n^2 - 1$ **do**   ▷ Compute IDFT from equation (4.26)

10          $a_k \leftarrow \frac{1}{n} \sum_{j=0}^{n^2-1} a_j\, \omega^{-kj}$

11          $p_k \leftarrow 2^{-m} \cdot \lfloor a_k \rfloor$ ▷ Truncate to m significant digits

12   **end for**

13   **return** $p_0, \ldots, p_{n^2-1}$ ▷ Return all p_k for $0 \le k < n^2$

14 **end function**

FIGURE 4.1: Pseudocode to compute the m most significant digits for probabilities $p_{rn+s} = \frac{Z^{r,s}_{1,n}}{Z}$. 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 $O(n^2 \log n)$.

---

first noticed in [71], as well as the triangle inequality $d_{BP}(\mathcal{A}, \mathcal{S}) + d_{BP}(\mathcal{S}, \mathcal{B}) \ge d_{BP}(\mathcal{A}, \mathcal{B})$ for base

pair distance, probably folklore. Lorenz et al. [71] exploited the parity condition and the triangle

inequality by using sparse matrix methods to improve on the efficiency of the naïve implemen-

tation of the $O(n^7)$ time and $O(n^4)$ space algorithm to compute the partition function, $Z^{r,s}_{1,n}$, and

minimum free energy structure, $MFE^{r,s}_{1,n}$, over all structures having base pair distance r to $\mathcal{A}$ and

$\mathcal{S}$ to $\mathcal{B}$.

Lemma 4.3. *If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (4.3), then for any complex nth 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*

$$\nu^k = -(\nu^{-(M_0-k)}) = -\overline{\nu^{M_0-k}}. \tag{4.15}$$

Lemma 4.2 is proved by induction; Lemma 4.3 is proved in Section 3.2 by a computation involving binomial coefficients; Lemma 4.4 is proved in Section B.3 by the parity observation above, resulting from Lemma 4.2; Lemma 4.5 is proved in Section B.4, relying on Euler's formula and trigonometric addition formulas.

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 *1)* all coefficients $z_1, z_3, z_5, \ldots$ of odd parity are zero; or *2)* 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 $\mathcal{Y}(\exp(\frac{2\pi i k}{M/2})) = \mathcal{Y}(\exp(\frac{4\pi i k}{M})) = \mathcal{Z}(\exp(\frac{2\pi i k}{M}))$, hence we can use the previous recursions from equation (4.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi i k}{M}))$. Instead of performing M evaluations of $\mathcal{Z}(x)$ at M-roots of unity, this requires only $M_0 = M/2$ evaluations of $\mathcal{Y}(u)$ 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 $\mathcal{Z}(x) = \sum_{k=0}^{M_0-1} z_{2k+1} x^{2k+1}$. 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 $\mathcal{Y}(\exp(\frac{2\pi i k}{M/2})) = \mathcal{Y}(\exp(\frac{4\pi k i}{M})) = \exp(\frac{-2\pi i k}{M}) \cdot \mathcal{Z}(\exp(\frac{2\pi i k}{M}))$, employing the previous recursions from equation (4.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi i k}{M}))$. 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 $\exp(\frac{-2\pi i k}{M})$ 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 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

$$\mathcal{Y}(\omega^k) = \mathcal{Y}(v^{2k})$$

$$= \begin{cases} \mathcal{Z}(v^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ \frac{1}{v^k} \cdot \mathcal{Z}(v^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases}$$

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

$$= \begin{cases} \mathcal{Z}(\overline{v^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ v^{-k} \cdot \mathcal{Z}(\overline{v^{(M_0-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases}$$

$$= \begin{cases} \overline{\mathcal{Z}(v^{(M_0-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 0 \bmod 2 \\ -v^{-k} \cdot \overline{\mathcal{Z}(v^{(M_0-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) = 1 \bmod 2 \end{cases}$$

Line 1 follows by definition, since $\omega = v^2$; 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

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

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

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

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 $v = \exp(\frac{2\pi i}{M})$ be the principal M-root of unity, and $\omega = v^2 = \exp(\frac{2\pi i}{M/2}) = \exp(\frac{2\pi \cdot 2i}{M})$ 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 $\omega^{i \cdot j} = v^{2i \cdot j}$; i.e.

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

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

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

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

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

Since we defined $\mathbf{Y}$ in equation (4.12) by $\mathbf{Y} = (y_0, \dots, y_{M_0-1})^{\mathrm{T}}$, where $y_0 = \mathcal{Z}(\alpha_0), \dots, y_{M_0-1} = \mathcal{Z}(\alpha_{M_0-1})$ and $\alpha_k = \omega^k \exp(\frac{k \cdot 2\pi i}{M_0})$, it follows that the coefficients $z_k = \mathbf{Z}_{1,n}^{\pi_1(k), \pi_2(k)}$ in the polynomial $\mathcal{Z}(x) = z_0 + z_1 x + \dots + z_M x^M$ defined in equation (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 $\mathbf{Z} = \sum_{\mathcal{S}} \exp(-E(\mathcal{S})/RT)$ 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 $p(x,y) = \frac{Z_{1,n}^{x,y}}{Z}$. 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 $p_{r \cdot (N+1)+s} = Z_{1,n}^{r,s}/Z$

INPUT: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, secondary structures $\mathcal{A}, \mathcal{B}$ of $\mathbf{s}$, integer m

OUTPUT: $p_{r \cdot (N+1)+s} = Z_{1,n}^{r,s}/Z$ to $\lfloor \log_{10}(2^m) \rfloor$ significant digits for $r,s = 0, \ldots, N$

---

1 **function** FFTBOR2D IMPROVED($\mathbf{s}, \mathcal{A}, \mathcal{B}, m$)

2     $n \leftarrow length(\mathbf{s})$

3     $N \leftarrow n + (n \bmod 2)$

4     $M \leftarrow (N + 1)^2 + ((N + 1)^2 \bmod 4)$

5     $M_0 \leftarrow \frac{M}{2}$

6   **for** $k \leftarrow 0, (N + 1)^2 - 1$ **do**   ▷ Note that $k \leftarrow \pi_1(k) \cdot M + \pi_2(k)$

7         $\pi_1(k) \leftarrow \lfloor \frac{k}{N+1} \rfloor$

8         $\pi_2(k) \leftarrow k \bmod (N + 1)$

9   **end for**

10   **for** $k \leftarrow 0, M - 1$ **do**   ▷ Compute all complex M and M_0-roots of unity

11       $v_k \leftarrow \exp(\frac{2\pi i k}{M})$

12   **if** $k < M_0$ **then**

13          $\omega_k \leftarrow \exp(\frac{2\pi i k}{M_0})$

14   **end if**

15   **end for**

16   **for** $k \leftarrow 0, M_0 - 1$ **do**

17   **if** $d_{\text{BP}}(\mathcal{A}, \mathcal{B}) \bmod 2 = 0$ **then**   ▷ From equation (4.22)

18   **if** $k \leq \frac{M_0}{2}$ **then**

19            $y_k \leftarrow \mathcal{Z}(v^k)$

20   **else**

21            $y_k \leftarrow \overline{\mathcal{Z}(v^{M_0-k})}$

22   **end if**

23   **else**   ▷ From equation (4.23)

24   **if** $k \leq \frac{M_0}{2}$ **then**

25            $y_k \leftarrow v^{-k} \cdot \mathcal{Z}(v^k)$

26   **else**

27            $y_k \leftarrow -v^{-k} \cdot \overline{\mathcal{Z}(v^{M_0-k})}$

28   **end if**

29   **end if**

30   **end for**

31 **for** $k \leftarrow 0, M_0 - 1$ **do**   ▷ Note that $\mathcal{Z}(v_0) = \mathbf{Z}$

32      $y_k \leftarrow 2^m \cdot \frac{y_k}{\mathcal{Z}(v_0)}$

33 **end for**

34 **for** $k \leftarrow 0, M_0 - 1$ **do**   ▷ Compute IDFT from equation (4.26)

35      $a_k \leftarrow \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \omega^{-kj}$

36 **end for**

37 **for** $k \leftarrow 0, M - 1$ **do**   ▷ Change the polynomial back to degree $M - 1$

38   **if** $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}) \bmod 2 = 0$ **then**

39   **if** $k \bmod 2 = 0$ **then**

40              $p_{\pi_1(k) \cdot (N+1) + \pi_2(k)} \leftarrow a_{k/2}$

41   **else**

42              $p_{\pi_1(k) \cdot (N+1) + \pi_2(k)} \leftarrow 0$

43   **end if**

44   **else**

45   **if** $k \bmod 2 = 0$ **then**

46              $p_{\pi_1(k) \cdot (N+1) + \pi_2(k)} \leftarrow 0$

47   **else**

48              $p_{\pi_1(k) \cdot (N+1) + \pi_2(k)} \leftarrow a_{\frac{k-1}{2}}$

49   **end if**

50   **end if**

51 **end for**

52 **for** $k \leftarrow 0, (N + 1)^2 - 1$ **do**   ▷ Truncate to m significant digits

53      $p_k \leftarrow 2^{-m} \cdot \lfloor p_k \rfloor$

54 **end for**

55 **return** $p_0, \ldots, p_{(N+1)^2 - 1}$ ▷ Return all $p(r,s) = p_{r \cdot (N+1) + s} = \frac{Z_{1,n}^{r,s}}{Z}$

56 **end function**

FIGURE 4.2: Pseudocode to compute the m most significant digits for probabilities $p_{r \cdot (N+1)+s} = \frac{Z_{1,n}^{r,s}}{Z}$. 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 **s**, the metastable structure $\mathcal{A}$ was taken to be the MFE structure of **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 [51] 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 $\mathbf{Z}$, for spliced leader RNA from *L. collosoma* of length 56 nt, `RNA2Dfold -d0` obtains a value of -9.660 kcal/mol, while `FFTbor2D` obtains -9.661 kcal/mol; similarly, for attenuator RNA of length 73 nt, `RNA2Dfold -d0` obtains a value of -22.172 kcal/mol, while `FFTbor2D` obtains -22.173 kcal/mol. Note that the straightforward calculation of the partition function, following McCaskill's algorithm [24] 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 the range 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.

Time benchmarking (each point is the average of 100 sequences)
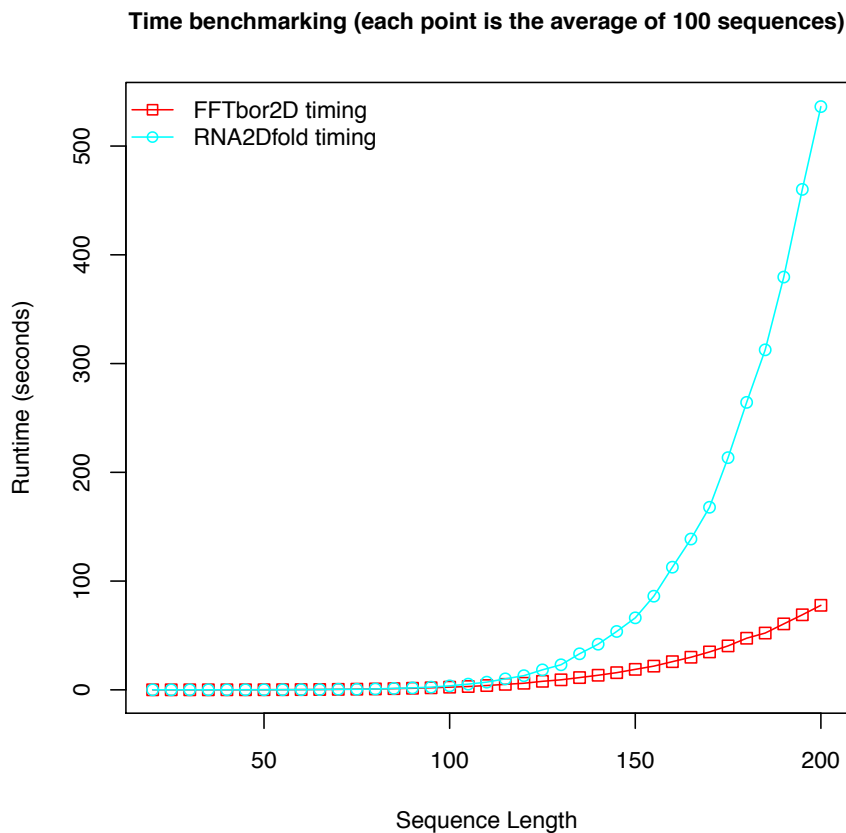
FIGURE 4.3: Run time in seconds for `RNA2Dfold` and `FFTbor2D` on random RNA sequences of length 20–200 nt, where sequence generation and choice of metastable structures $\mathcal{A}, \mathcal{B}$ is described in the text. Beyond a length of approximately 80 nt, `FFTbor2D` is demonstrably faster.

An important advantage of `RNA2Dfold` over `FFTbor2D` is that the former can additionally compute the structures $M_{x,y}$ having minimum free energy over all structures that are x-neighbors of metastable $\mathcal{A}$ and simultaneously y-neighbors of metastable $\mathcal{B}$. (There is a similar advantage of `RNAbor` [51] over the faster `FFTbor` [54].) As well, `RNA2Dfold` directly computes the partition function values $\mathbf{Z}^{x,y}$, while `FFTbor2D` estimates $\mathbf{Z}^{x,y}$ by computing $p(x,y) \cdot \mathbf{Z}$. This difference entails a significant loss of precision, when depicting the energy landscape.
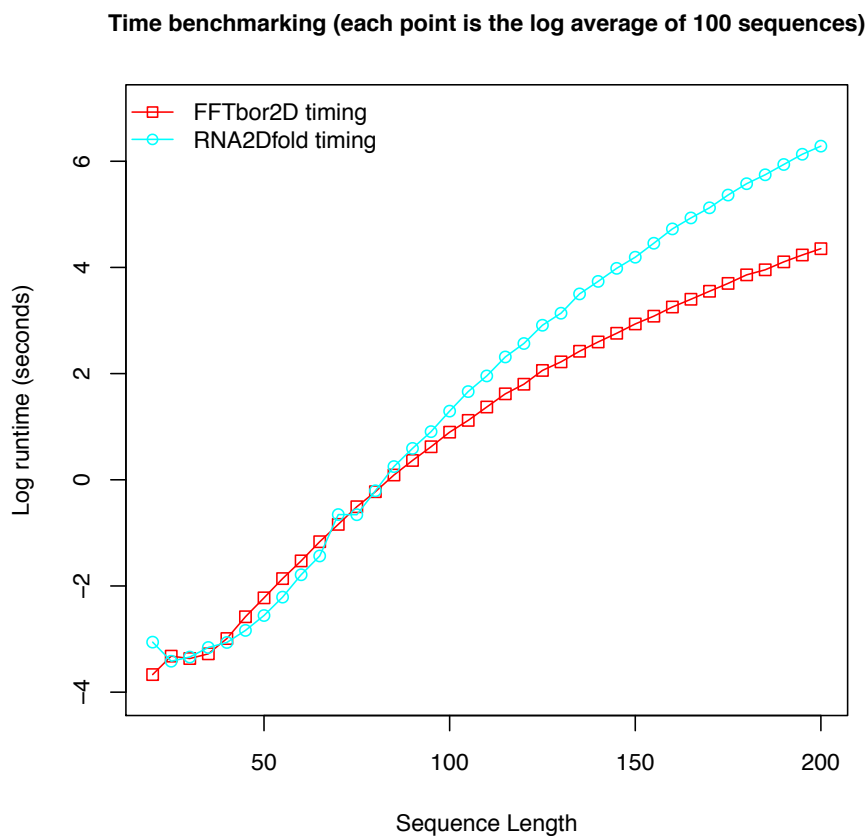
FIGURE 4.4: Logarithm of run time in seconds for `RNA2Dfold` and `FFTbor2D` on random RNA sequences of length less than 200 nt, for same data as that in Figure 4.3. By taking $\log_{10}$ of the run times, crossover points are apparent, where `FFTbor2D` is faster than `RNA2Dfold`. For very small sequences, `RNA2Dfold` is faster, though since both programs converge in a fraction of a second, this difference is of no practical consequence.

Standard deviation of runtimes (sample of 100 sequences per point)

Time benchmarking (each point is the min (resp. max) of 100 sequences)
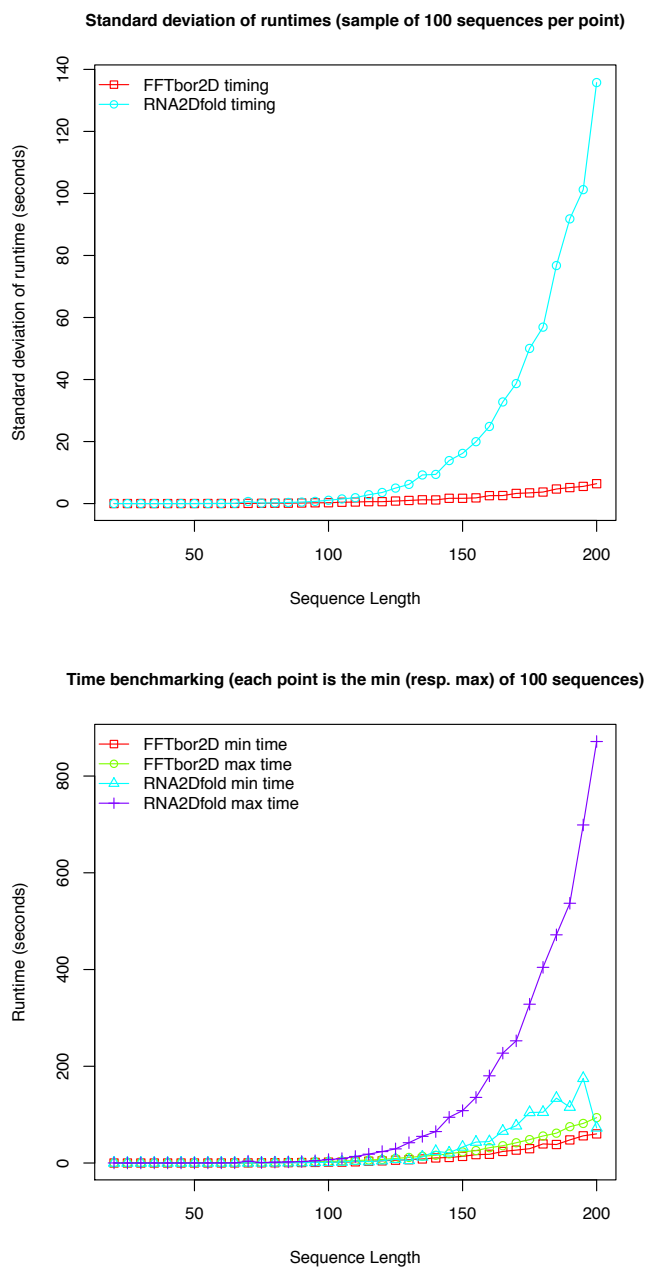
FIGURE 4.5: *(Top)* Standard deviation of run times of `RNA2Dfold` and `FFTbor2D` as a function of sequence length *n*. *(Bottom)* Minimum and maximum run times for `RNA2Dfold` and `FFTbor2D`. For each collection of 100 random sequences of length *n*, the minimum and maximum run time for a sequence of 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.
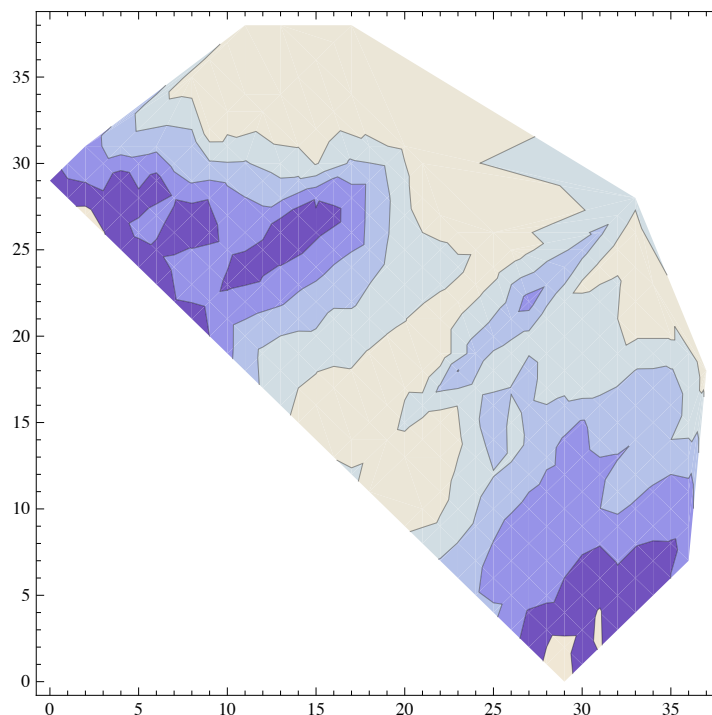
FIGURE 4.6: 2D projection of energy landscape for Spliced Leader (SL) RNA from *L. collosoma*, having sequence and metastable secondary structures:

$\mathbf{s}$ = 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.
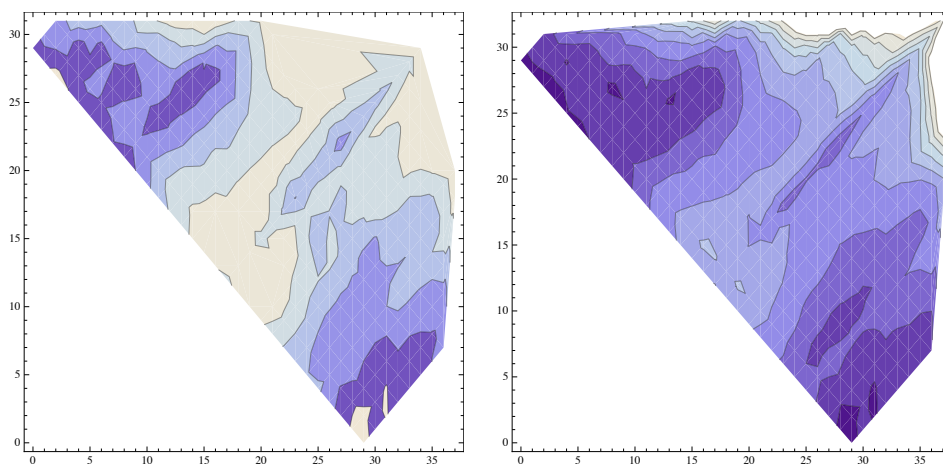
FIGURE 4.7: 2D projection of energy landscape for Spliced Leader (SL) RNA from *L. collosoma*, as in Figure 4.6, except that in the right panel, ensemble free energy $-RT \log \mathbf{Z}^{x,y}$ is computed from the values of $\mathbf{Z}^{x,y}$ output by `RNA2Dfold`, while in the left panel, ensemble 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

Remarkable results in RNA synthetic biology have recently been obtained by various groups. In [72], small conditional RNAs have been engineered to silence a gene Y by using the RNA interference machinery, only if a gene X is transcribed. In [73] a novel theophylline riboswitch has been computationally designed to transcriptionally regulate a gene in *E. coli*, and in [74] a purely computational approach was used to design functionally active hammerhead ribozymes. Computational design of synthetic RNA molecules invariably uses some form of thermodynamics-based algorithm; indeed, `NUPACK-Design` [75] was used to design small conditional RNAs [72], Vienna RNA Package [76] was used in the design of the synthetic theophylline riboswitch, and the `RNAiFold` inverse folding software [77, 78] was used to design the synthetic hammerhead ribozymes. The next step in the computational design of synthetic RNA molecules is to control the kinetics of folding—such control could be important in engineering conformational

switches. Kinetics is already used as a design feature for synthetic design in proteins [79, 80].

In this chapter we introduce a new software suite, called `Hermes` [81], which efficiently computes RNA secondary structure folding kinetics by using a coarse-grained method to model RNA transitions that add or remove a single base pair. Since our motivation in developing `Hermes` is to provide a new tool to aid in engineering synthetic RNA molecules with desired kinetic properties, `Hermes` does not model *co-transcriptional folding*, but only the *refolding* of RNA sequences.

There is a rich history of both experimental and computational work on RNA folding pathways and kinetics. Experimental approaches to determine the kinetics of RNA folding include temperature jump experiments [68], using fluorophores [82], using mechanical tension at the single molecule level [83], etc. and will not be further discussed. Computational approaches to folding kinetics commonly model stepwise transitions between secondary structures, involving the addition or removal of a single base pair, as first considered in [84]. Nevertheless, it should be noted that there are a number of methods that concern the addition or removal of an entire helical region, e.g. [85, 86].

Most computational approaches involve either *1)* algorithms to determine optimal or near-optimal folding pathways; *2)* explicit solutions of the master equation; or *3)* repeated simulations to fold an initially empty secondary structure to the target minimum free energy (MFE) structure. Examples of methods to determine optimal or near-optimal folding pathways include the greedy approach of Morgan and Higgs [87], the exact, optimal, exponential time program `barriers` [84], the program `Findpath` [88], which uses bounded look-ahead breadth-first search, a genetic algorithm [89], the program `RNAtabupath` that uses local search to determine

near-optimal folding pathways, the Basin Hopping Graph (BHG, [90]) which uses a greedy algorithm to determine local minima and approximate their barrier energies, etc. The program `barriers` [84] is the only current method guaranteed to produce optimal folding pathways. Since it has been shown that the problem of determining optimal folding pathways is NP-complete [91], it is now understandable why `barriers` can take exponential time to converge, depending on the RNA sequence. For this reason, near-optimal solutions provided by heuristic methods, such as `RNAtabupath`, are very useful.

Methods that employ the master equation include `Treekin` [92], which uses the programs `RNAsubopt` [93] and `barriers` [84] to determine *macrostates*, defined as basins of attraction near a locally optimal structure. The resulting coarse-grained Markov chain is then sufficiently small to allow explicit solution of the master equation. In [94], a moderate number of RNA structures were sampled according to different strategies, from which a robotic motion planning graph was defined to connect each sampled structure to its k nearest sampled neighbors. Again, the resulting coarse-grained Markov chain is sufficiently small for an explicit solution of the master equation to be given.

We now come to simulation approaches to estimate RNA folding kinetics. The program `Kinfold` [65, 95] is an implementation of Gillespie's algorithm [96], directly related to the master equation, hence is considered by many to be the *gold standard* for RNA kinetics. A recent extension of the `Kinfold` algorithm was reported in [97]. `KFOLD` [98] also implements Gillespie's algorithm, but leverages memoization of transition rates to operate much more efficiently than `Kinfold`. Meanwhile `Kinefold` [99] uses stochastic simulations of the nucleation and dissociation of helical regions to predict secondary structure and folding pathways. In contrast to

the previously mentioned methods, both `RNAKinetics` [100] and `Kinwalker` [101] model co-transcriptional folding, known to be necessary when simulating *in vivo* folding of long RNA molecules [102]. As well, `Kinefold` can simulate both refolding and co-transcriptional folding pathways. Finally, unlike all the previous simulation results, which depend on thermodynamic free energy parameters [103], the program `Oxfold` [104] performs kinetic folding of RNA using stochastic context-free grammars and evolutionary information.

In contrast to the previous methods, `Hermes` computes the mean first passage time (MFPT) and equilibrium time for a coarse-grained Markov chain consisting of the ensemble of all secondary structures having base pair distance x [resp. y] from reference structures $\mathcal{A}$ [resp. $\mathcal{B}$] of a given RNA sequence. Mean first passage time (MFPT) is computed exactly by matrix inversion, and equilibrium time is computed using spectral decomposition of the rate matrix for the coarse-grained master equation.

5.3   Traditional approaches for kinetics

To better understand the underlying algorithms behind the software, we describe two traditional approaches in kinetics, mean first passage time and equilibrium time.

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 tth 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 $length(\rho)$ times the probability of path ρ, where $length(\rho_0,\ldots,\rho_n)$ is defined to be n. In other words, $MFPT = \sum_\rho Pr[\rho] \cdot length(\rho)$, 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 $(I - P_{x_\infty}^-)^{-1} \cdot \mathbf{e}$ [105].

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 [106].

The Metropolis Monte Carlo algorithm [107] 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.

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

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 $\mathbf{Z} = \sum_{k=1}^{n} \exp(-E(k)/RT)$ is a normalization constant [106, 108]. 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 [109], 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.

5.3.2   Equilibrium time

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

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      $t \leftarrow 0$

3      $x \leftarrow x_0$

4   **while** $x \neq x_\infty$ AND $t < T_{\max}$ **do**

5         $y \in N_x$

6   **if** $E(y) < E(x)$ **then**   ▷ Greedy move

7            $x \leftarrow y$

8   **else**   ▷ Metropolis move

9            $z \in (0,1)$

10   **if** $z < \exp(-\frac{E(y)-E(x)}{RT})$ **then**

11               $x \leftarrow y$

12   **end if**

13   **end if**

14         $t \leftarrow t + 1$

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.

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

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

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

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

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

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

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.

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

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 jth row of T is the jth 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 [110], 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 $p_k^* = \frac{\exp(-E(k)/RT)}{Z}$. 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 [96] 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` [65, 95].

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      $t \leftarrow 0$

3      $x \leftarrow x_0$

4   **while** $x \neq x_\infty$ AND $t < T_{\max}$ **do**

5         $\Phi \leftarrow 0$ ▷ Φ is the *flux* out of x

6   **for** $y \in N_x$ **do**

7            $r_{x,y} \leftarrow \min\left(1, \exp(-\frac{E(y)-E(x)}{RT})\right)$

8            $\Phi \leftarrow \Phi + r_{x,y}$

9   **end for**

10   **for** $y \in N_x$ **do**

11           $r_{x,y} \leftarrow \frac{r_{x,y}}{\Phi}$

12   **end for**

13        $z_1 \in (0,1)$

14        $t \leftarrow t + (-\frac{1}{\Phi}\ln(z_1))$ ▷ Update t to reflect time spent in state x

15        $z_2 \in (0,1)$

16        $s \leftarrow 0$

17   **for** $y \in N_x$ **do**   ▷ Use roulette wheel to select new state for x

18            $s \leftarrow s + r_{x,y}$

19   **if** $z_2 \leq s$ **then**

20               $x \leftarrow y$

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.

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 Fourier Transform [111] ($\geq$

3.3.4) `http://www.fftw.org/`, Gnu Scientific Library GSL ($\geq$ 1.15) `http://www.gnu.org/`

`software/gsl/`, Vienna RNA Package [30] ($\geq$ 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.
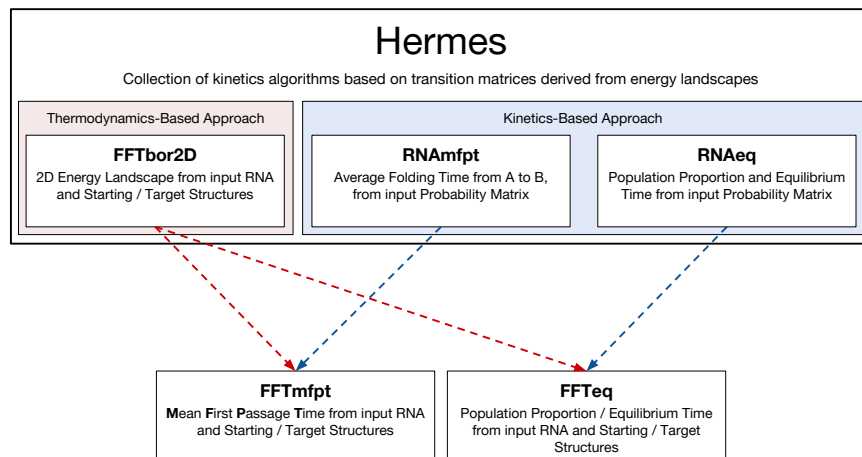
FIGURE 5.3: Overall organization of `Hermes`. `FFTbor2D`, `RNAmfpt`, and `RNAeq` are three distinct software packages we have developed, which compile into both standalone executables and archive files, providing an API that allow novel applications development using source from each of the packages, 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 [64] 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.

5.4.1 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 [105]—see Section 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 **s** and secondary structure x, let $N(x)$ denote the set of neighboring secondary structures of **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 **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 [105] 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` [54] 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

$$Q_1 = \{(x,y) : 0 \leq x,y \leq n, (x + y \bmod 2) = (d_0 \bmod 2),$$

$$(d_0 \leq x + y), (x \leq d_0 + y), (y \leq d_0 + x)\}. \tag{5.19}$$

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

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

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

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

As we report in Section 5.5.1, given an RNA sequence **s**, if $\mathcal{A}$ is the empty structure and $\mathcal{B}$ the MFE structure of **s**, then `FFTmfpt` output is well correlated with the exact MFPT in folding the empty structure to the MFE structure, where transitions between structures involve the addition or removal of a single base pair.

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` [112], 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. Indices

(x,y) in the rate matrix correspond to all strucutres having base pair distance x [resp. y] to input

structure $\mathcal{A}$ [resp. $\mathcal{B}$], 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

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

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. 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 is described in equation (5.14). 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.

FFTbor2D folding kinetics for L. collosoma spliced leader RNA
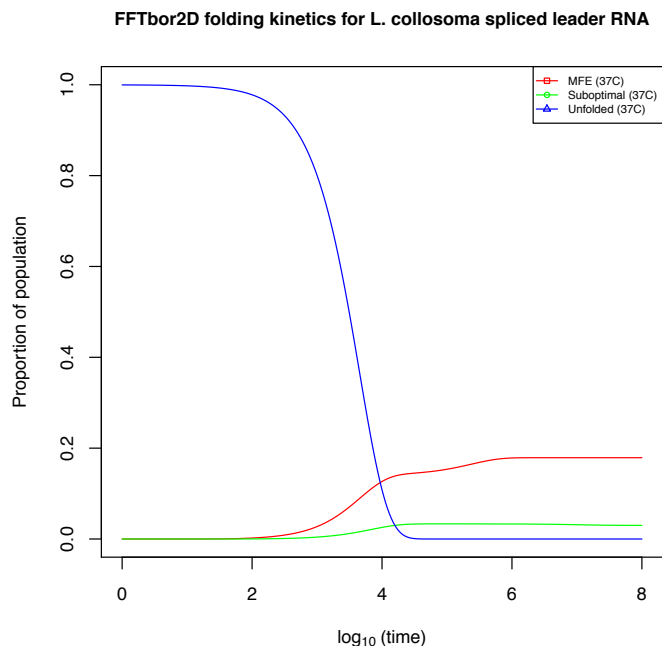
FIGURE 5.4: Population occupancy curves computed with `FFTeq` for the 56 nt conformational switch *L. collosoma* spliced leader RNA, with sequence `AACUAAAACAAUUUUUGAAGAACAGUUUCUGUACUUCAUUGGUAUGUAGA` GACUUC. The dot bracket format for the MFE structure, as computed by version 2.1.7 of `RNAfold -d0`, 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 0.1789 should equal the Boltzmann probability 0.1791, 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 0.0300 for the (x_1,y_1) be approximately the Boltzmann probability 0.0301 of the alternate structure.

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` [92]. Our first approximation of equilibrium time involved using a sliding window approach, where we compute the smallest t_0, such that for $t \in \{t_0 + t_\Delta, t_0 + 2t_\Delta, t_0 + 3t_\Delta, \ldots, t_0 + (w - 1) \cdot t_\Delta\}$, the absolute difference $|p(t)[x_\infty] - p(t_0)[x_\infty]| < \epsilon$, for $\epsilon = 10^{-4}$ and step size t_Δ across a window of length w. In systems where there are local energy minima, this 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 a window of size w, having step size t_Δ

---

1 **function** ESTIMATEEQUILIBRIUM($x_0, x_\infty, T_{\min}, T_{\max}, t_\Delta, \delta, \epsilon, w$)

2     $t \leftarrow$ SOFTBOUND($x_\infty, T_{\min}, T_{\max}, t_\Delta, \delta, s$)

3   **if** WINDOWEQ($x_\infty, t, t_\Delta, w, \epsilon$) **then**

4   **while** $t > T_{\min}$ AND WINDOWEQ($x_\infty, t, t_\Delta, w, \epsilon$) **do**

5             $t \leftarrow t - t_\Delta$

6   **end while**

7   **end if**

8   **while** $t < T_{\max}$ AND NOT WINDOWEQ($x_\infty, t, t_\Delta, w, \epsilon$) **do**

9         $t \leftarrow t + t_\Delta$

10   **end while**

11   **if** $t = T_{\min}$ **then**   ▷ x doesn't leave equilibrium after $T_{\min}$

12   **return** $-\infty$

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     $i_\infty \leftarrow p_i(10^{T_{\max}})$ ▷ From equation (5.15)

21     $t \leftarrow T_{\min} + \frac{T_{\max}-T_{\min}}{2}$

22     $t' \leftarrow T_{\max}$

23   **while** $|t - t'| > t_\Delta$ **do**   ▷ Binary search for time t within δ of $T_{\max}$

24         $t_{\text{temp}} \leftarrow t$

25   **if** $|p_i(10^t) - p_i(10^{t'})| > \delta$ **then**

26             $t \leftarrow t + s * \frac{|t-t'|}{2}$ ▷ s is 1 [resp. −1] for the right [resp. left] bound

27   **else**

28             $t \leftarrow t - s * \frac{|t-t'|}{2}$

29   **end if**

30         $t' \leftarrow t_{\text{temp}}$

31   **end while**

32   **return** t

33 **end function**

34 **function** WINDOWEQ(x, t, t_Δ, w, ϵ)   ▷ Is x within ϵ over $[t, t + (w - 1) \cdot t_\Delta]$

35   **for** $i \leftarrow 1, w - 1$ **do**

36   **if** $|p_x(10^{t+i \cdot t_\Delta}) - p_x(10^t)| > \epsilon$ **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 $t \in \{t_0+t_\Delta, t_0+2t_\Delta, t_0+3t_\Delta, \ldots, t_0+(w-1)\cdot t_\Delta\}$, 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 root-mean-square deviation (RMSD) between $p_{x_\infty}(t_0)$ and $\frac{\exp(-E(x_\infty)/RT)}{Z}$ is 0.0041, where RMSD is defined as

$$\sqrt{\frac{\sum_{i=1}^{n}(\widehat{x_i} - x_i)^2}{n}}. \tag{5.27}$$

Visual inspection of the estimated equilibrium times (Figure 5.6) are much better than the naïve approach used by `treekin`, which uses a simple sliding window for all $x \in Q$. We also allow defining the equilibrium time to be the smallest t_0, such that for $t \in \{t_0 + t_\Delta, t_0 + 2t_\Delta, t_0 + 3t_\Delta, \ldots, t_0 + (w-1) \cdot t_\Delta\}$, 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.
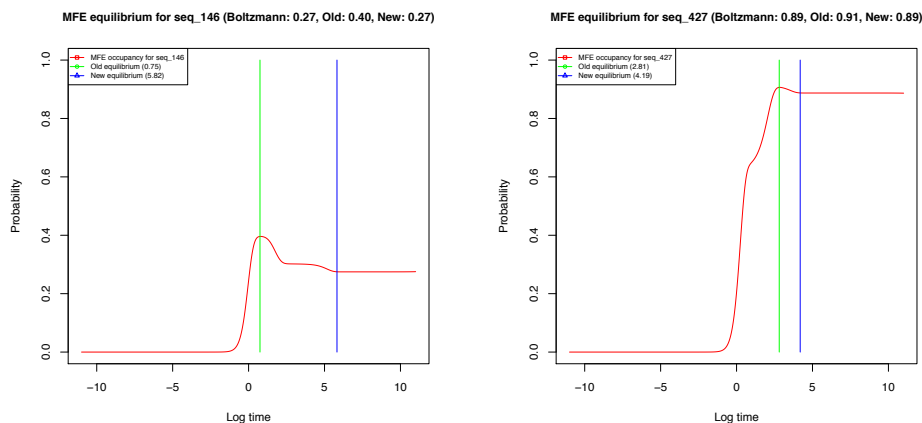
FIGURE 5.6: Example of the differences in using the simple sliding window approach (green) versus the approach outlined in Algorithm 5.5 (blue). There are two representative sequences shown from the benchmarking dataset described in Section 5.5, sequence #146 *(Left)* and #427 *(Right)*. Note that the $p(t)$ for the approach from Algorithm 5.5 is much closer to the computed Boltzmann probability—the root-mean-square deviation (RMSD, equation 5.27) across the dataset of 1,000 sequences is 0.0041 [resp. 0.0491] for the approach described in Algorithm 5.5 [resp. simple sliding window].

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 [64].

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
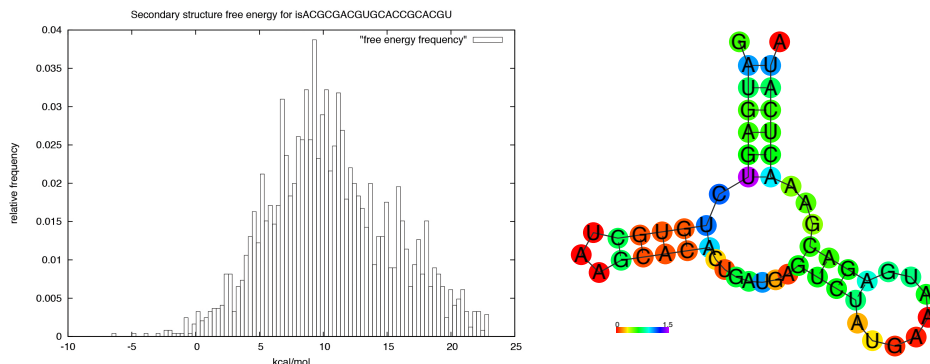
FIGURE 5.7: *(Left)* Histogram of free energies of secondary structures of `ACGCGACGUGCACCGCACGU`, which range from −6.5 to +25 kcal/mol, with mean of 10.7 kcal/mol. *(Right)* Minimum free energy structure of the 54 nt Peach Latent Mosaic Viroid (PLMVd) AJ005312.1 282–335, which is identical to the consensus structure from Rfam 11.0 [64]. `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 [74]. 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`.

sorted by increasing folding time mean. Standard deviation exceeded the mean in 83.9% of the 1,000 cases, indicating the enormous variation between separate `Kinfold` runs, even for 20 nt RNA sequences having at most 2,500 secondary structures. Elementary considerations from statistics indicate that for our benchmarking set of 20-mers, the minimum sample size $n = \left(\frac{z_{\alpha/2} \cdot \sigma}{E}\right)^2$ ranges from 937,712 to 23,289,310 to have a confidence level of 95% that the average of n `Kinfold` runs differs from the real folding time by at most 100 steps. 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.

With this in mind, it is natural to turn to coarse-grained models, as done by Wolfinger et al. [92] and by Tang et al. [113]. The software of Tang et al. appears not to be available. Concerning the method of Wolfinger et al. (called `BarriersEq` in our benchmarking), there is now a web server available, which runs `RNAsubopt` [112] to generate all secondary structures within a user-specified energy range, then runs `barriers` [84] to generate basins of attraction around a user-specified number of locally optimal structures, and then runs `treekin` on the output of `barriers`. The program `treekin` performs some of the same operations as `Hermes`, by computing population occupation frequencies by spectral decomposition. Nevertheless it would require a user to write scripts and perform several manual steps, in order to determine the equilibrium time for an input RNA sequence, with respect to the macrostate Markov process of [92]. In addition, because `barriers` computes basins of attraction by utilizing the output of `RNAsubopt`, estimating kinetics for the refolding of an RNA molecule from the empty structure requires exhaustive enumeration of all suboptimal structures having non-positive free energy.

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 `RNAeq`, using equation (5.25); *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 using equation (5.25); and finally *6)* `RNA2Dfold` with and without the Hastings modification using equations (5.23) and (5.24).
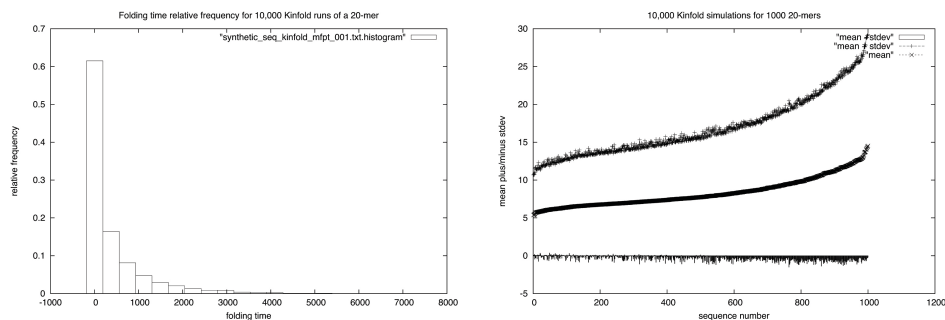
FIGURE 5.8: *(Left)* Histogram of `Kinfold` folding times for 20-mer `CCGAUUGGCGAAAGGCCACC`. The mean [resp. standard deviation] of 10,000 runs of `Kinfold` for this 20-mer is 538.37 [resp. 755.65]. Note the close fit to the exponential distribution, *(Right)* Mean minus standard deviation ($\mu - \sigma$), 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.

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 [92] 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.
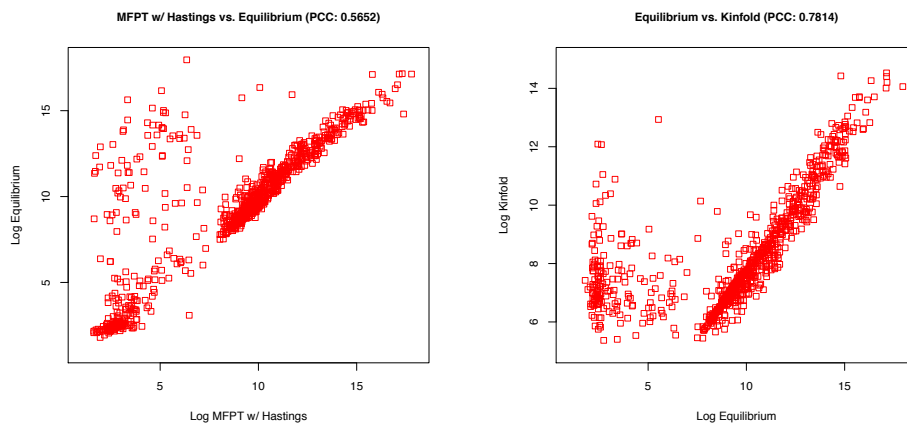
FIGURE 5.9: Scatter plots of the natural logarithm of times from `RNAmfpt` versus `RNAeq` *(Left)* and for `Kinfold` versus `RNAeq` *(Right)*.
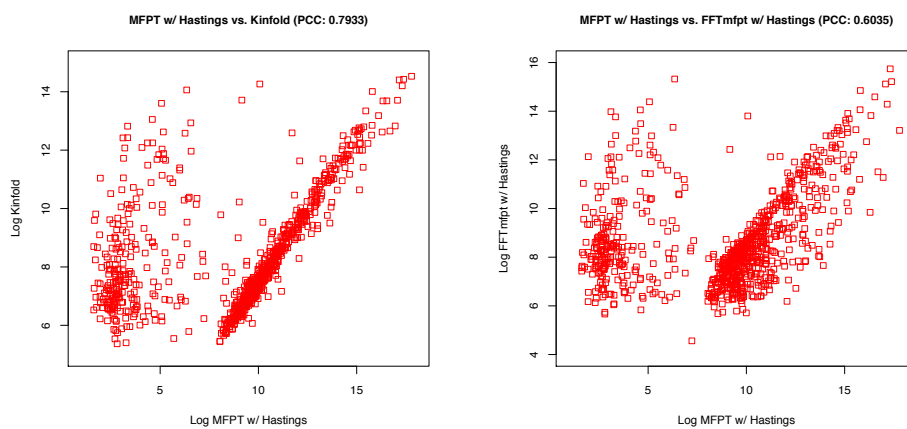

FIGURE 5.10: Scatter plots of the natural logarithm of times from `RNAmfpt` versus `Kinfold` *(Left)* and for `RNAmfpt` versus `FFTmfpt` *(Right)*.
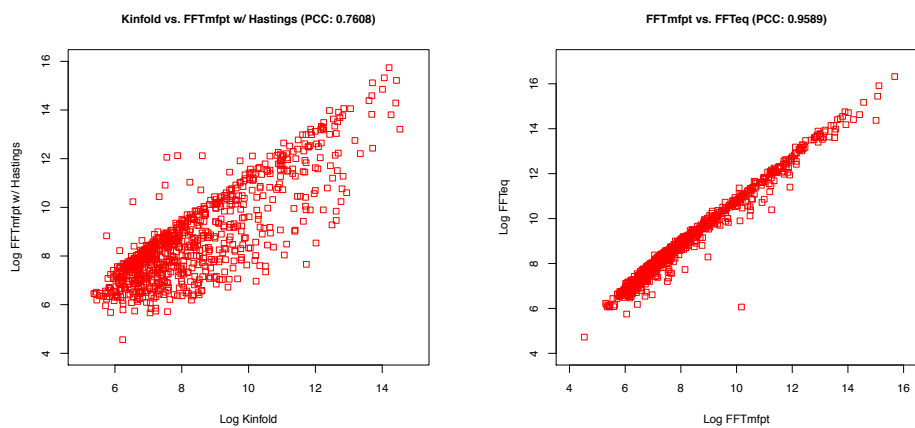
FIGURE 5.11: Scatter plots of the natural logarithm of times from `Kinfold` versus `FFTmfpt` *(Left)* and for `FFTmfpt` versus `FFTeq` *(Right)*.

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 $P(x,y) = \sum_{S} \frac{\exp(-E(S)/RT)}{Z}$, 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 $P(x) = \sum_{S} \frac{\exp(-E(S)/RT)}{Z}$, 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

Over the course of this thesis, we have described a collection of tools aimed at facilitating the computational analysis of RNA. The work in Chapter 2 aims to address the open problem of determining full riboswitch sequences, along with their 'on' and 'off' conformational states in an efficient and generalized manner. Riboswitches are an imporant regulatory motif that can influence transcription by the introduction of an intrinsic terminator, an extended stem-loop followed by a series of uracil residues which cause RNA polymerase to prematurely terminate transcription [114, 115]. Alternately, riboswitches can regulate translation, generally by sequestration of the ribosome binding site in response to the presence or absence of a corresponding ligand [116]. This degree of control makes riboswitches well suited for the regulation of metabolic pathways, and are generally found within bacterial genomes. While the aptamer portion of the riboswitch necessarily exhibits high sequence and structural conservation—necessary to retain a high binding affinity for the corresponding ligand—the downstream expression platform is much more variable, and difficult to detect using covariance models such as `Infernal` [27]. For this reason, databases such as Rfam [117] only contain alignments for the conserved

aptamer domain; rarely enough to understand the structural characteristics of the riboswitch 'states' in the greater context of the mRNA. `Ribofinder` aims to provide a holistic description of riboswitches by coupling powerful aptamer-detection tools (such as `Infernal`) with software designed to detect intrinsic terminators (such as `TransTermHP`), and use experimentally-derived structural information from the literature in conjunction with constrained folding methods to predict full gene 'on' and 'off' conformations in a general manner.

The remainder of the thesis follows a natural progression. In Chapter 3 we present the algorithm `FFTbor`, an efficient approach to compute the *parameterized* Boltzmann probabilities $p_k = \frac{Z^k}{Z}$ of those sequences having base pair distance k from an arbitrary input structure $\mathcal{S}^*$. By using the Fast Fourier Transform along with complex nth roots of unity, `FFTbor` operates in $O(n^4)$ speed, an order of magnitude faster than its predecessor. This approach of applying the Fast Fourier Transform to compute parameterized partition functions was put to use in [118] as well, where we develop programs `FFThairpin` [resp. `FFTmultiloop`] to describe the probability distribution of structures having exactly k hairpins [resp. multiloops]. Though not discussed here, by training a support vector machine (SVM) on the probability distributions output by these programs we were able to classify Rfam families using five-fold cross validation with excellent 'area under curve' (AUC) values.

Chapter 4 uses the Fast Fourier Transform to compute the 2D probability distribution, where given two compatible input secondary structures $\mathcal{A}$ and $\mathcal{B}$, position (x,y) on the discrete energy landscape corresponds to the Boltzmann probability for those structures $\mathcal{S}$ which have base pair distance x [resp. y] from $\mathcal{A}$ [resp. $\mathcal{B}$]. Compared against its closest relative `RNA2Dfold`, `FFTbor2D` is two orders of magnitude faster, operating in $O(n^5)$ speed and only $O(n^2)$ space.

This opens the door for using `FFTbor2D` in a high-throughput fashion, where the probability landscapes can be used as a starting point for kinetic analysis.

With `Hermes` in Chapter 5, we do just that; representing the output of `FFTbor2D` as a graph, we can create a corresponding Markov chain [resp. Markov process] to estimate mean first passage time [resp. equilibrium time] with correlation to the mathematically-exact values as shown in Table 5.1. The speed of these approximate tools within `Hermes`—namely `FFTmfpt` and `FFTeq`—present what we believe to be the first suite of kinetic analysis tools for RNA sequences that are suitable for high throughput usage, something we believe to be of interest in the field of synthetic design.

Finally, though not discussed in the body of this thesis, we have also developed a collection of packages for working with command-line tools for computational RNA analysis in a more streamlined fashion. Two of these programs, `wrnap` and `rbfam` are among the first of their type for the Ruby programming langage, a popular interpreted language that enjoys wide adoption in the web development community. In an era of increasingly cloud-focused software, we believe that researchers will need a robust set of tools for working with their data in a web-oriented environment, something `wrnap` and `rbfam` aim to provide.

Appendix A

FFTbor Appendix

A.1 Full recursions for $\mathcal{Z}_{i,j}(x)$ for the Turner energy model

To compute $\mathcal{Z}(x) = \mathcal{Z}_{1,n}(x)$ given input structure $\mathcal{S}$, we use the recursions

$$\mathcal{Z}_{i,j}(x) = \mathcal{Z}_{i,j-1}(x) \cdot x^{d_0} + \sum_{\substack{(s_k, s_j) \in \mathbb{B}, \\ i \le k < j}} \left(\mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}\mathcal{B}_{k,j}(x) \cdot e^{\frac{-E_d}{RT}} \cdot x^{d_1} \right), \tag{A.1}$$

where $d_0 = 1$ if j is base paired in $\mathcal{S}_{[i,j]}$ and 0 otherwise, $d_1 = d_{\mathrm{BP}}(\mathcal{S}_{[i,j]}, \mathcal{S}_{[i,k-1]} \cup \mathcal{S}_{[k,j]})$, and E_d is the energy contribution due to dangling ends (energy contributions from single bases stacking on adjacent base pairs) and closing AU base pairs (since a non GC base pair closing a stem has a destabilizing effect). The sum is taken over all possible base pairs (k,j) with $i \le k < j$.

We compute $\mathcal{Z}\mathcal{B}(x)$ using the recursion

$$\mathcal{ZB}_{i,j}(x) = e^{\frac{-EH(i,j)}{RT}} \cdot x^{d_2}$$

$$+ \sum_{\substack{(s_k,s_l)\in\mathbb{B}, \\ i<k<l<j}} \mathcal{ZB}_{k,l}(x) \cdot e^{\frac{-EI(i,j,k,l)}{RT}} \cdot x^{d_3} \tag{A.2}$$

$$+ \sum_{\substack{(s_k,s_l)\in\mathbb{B}, \\ i<k<l<j}} \left(\mathcal{ZM}_{i+1,k-1}(x) \cdot \mathcal{ZB}_{k,l}(x) \cdot e^{\frac{-(a+b+c(j-l-1))}{RT}} \cdot x^{d_4} \right)$$

where $d_2 = d_{\text{BP}}(\mathcal{S}_{[i,j]}, \{(i,j)\})$, $EH(i,j)$ is the energy of the hairpin loop with closing base pair

(i,j), $EI(i,j,k,l)$ is the energy of the stack, bulge or interior loop with the closing base pair (i,j)

and the interior base pair (k,l), $d_3 = d_{\text{BP}}(\mathcal{S}_{[i,j]}, \mathcal{S}_{[k,l]} \cup \{(i,j)\})$, and $d_4 = d_{\text{BP}}(\mathcal{S}_{[i,j]}, \mathcal{S}_{[i+1,k-1]} \cup$

$\mathcal{S}_{[k,l]} \cup \{(i,j)\})$. The first term in the recursion takes care of the case where (i,j) is the only

base pair in $[i,j]$, i.e. (i,j) closes a hairpin loop. The second term handles the case where there

is an interior loop (or a bulge or a stack) closed by (i,j) and (k,l). The third term takes care

of all the structures where (i,j) closes a multiloop. To reduce complexity of the algorithm, the

interior and bulge loop size can be limited to a maximum size of L (taken by default to be 30),

by requiring that $l > j - L$ in the above recursion.

The final recursion, for computing $\mathcal{ZM}(x)$, is

$$\mathcal{ZM}_{i,j}(x) = \mathcal{ZM}_{i,j-1}(x) \cdot e^{\frac{-c}{RT}} \cdot x^{d_0}$$

$$+ \sum_{\substack{(s_k,s_j)\in\mathbb{B}, \\ i\leq k<j}} \left(\mathcal{ZB}_{k,j}(x) \cdot e^{\frac{-(b+c(k-i))}{RT}} \cdot x^{d_5} \right. \tag{A.3}$$

$$+ \mathcal{ZM}_{i,k-1}(x) \cdot \mathcal{ZB}_{k,j}(x) \cdot e^{\frac{-b}{RT}} \cdot x^{d_6} \bigg)$$

where $d_5 = d_{\text{BP}}(\mathcal{S}_{[i,j]}, \mathcal{S}_{[k,j]})$ and $d_6 = d_{\text{BP}}(\mathcal{S}_{[i,j]}, \mathcal{S}_{[i,k-1]} \cup \mathcal{S}_{[k,j]})$. Note that since $\mathcal{ZM}_{i,j}(x)$

computes the partition function contribution under the assumption that $[i,j]$ is part of a mul-

tiloop, there will be exactly one stem-loop structure in this region (the $\mathcal{ZB}(x)$ term) or more

than one (the $\mathcal{ZM}(x)-\mathcal{ZB}(x)$ term). Justification of recursions (A.1), (A.2), and (A.3) follow

by induction, as in the proof of Theorem 3.1.

Appendix B

FFTbor2D Appendix

B.1 Full recursions for $\mathcal{Z}_{i,j}(x)$ for the Turner energy model

To compute $\mathcal{Z}(x) = \mathcal{Z}_{1,n}(x)$ given input structures $\mathcal{A}, \mathcal{B}$, we use the recursions

$$\mathcal{Z}_{i,j}(x) = \mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n + \beta_0} + \sum_{\substack{(s_k, s_j) \in \mathbb{B}, \\ i \leq k < j}} \left(\mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}\mathcal{B}_{k,j}(x) \cdot e^{\frac{-E_d}{RT}} \cdot x^{\alpha_1 n + \beta_1} \right), \qquad \text{(B.1)}$$

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, $\alpha_1 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i,k-1]} \cup \mathcal{A}_{[k,j]})$, $\beta_1 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k,j]})$, and E_d is the energy contribution due to dangling ends (energy contributions from single bases stacking on adjacent base pairs) and closing AU base pairs (since a non GC base pair closing a stem has a destabilizing effect). The sum is taken over all possible base pairs (k,j) with $i \leq k < j$.

We compute $\mathcal{Z}\mathcal{B}(x)$ using the recursion

$$\mathcal{ZB}_{i,j}(x) = e^{\frac{-EH(i,j)}{RT}} \cdot x^{\alpha_2 n + \beta_2}$$

$$+ \sum_{\substack{(s_k, s_l) \in \mathbb{B}, \\ i < k < l < j}} \mathcal{ZB}_{k,l}(x) \cdot e^{\frac{-EI(i,j,k,l)}{RT}} \cdot x^{\alpha_3 n + \beta_3} \tag{B.2}$$

$$+ \sum_{\substack{(s_k, s_l) \in \mathbb{B}, \\ i < k < l < j}} \left(\mathcal{ZM}_{i+1,k-1}(x) \cdot \mathcal{ZB}_{k,l}(x) \cdot e^{\frac{-(a+b+c(j-l-1))}{RT}} \cdot x^{\alpha_4 n + \beta_4} \right)$$

where $EH(i,j)$ is the energy of the hairpin loop with closing base pair (i,j), $\alpha_2 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \{(i,j)\})$, $\beta_2 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \{(i,j)\})$, $EI(i,j,k,l)$ is the energy of the stack, bulge or interior loop with the closing base pair (i,j) and the interior base pair (k,l), $\alpha_3 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[k,l]} \cup \{(i,j)\})$, $\beta_3 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[k,l]} \cup \{(i,j)\})$, $\alpha_4 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i+1,k-1]} \cup \mathcal{A}_{[k,l]} \cup \{(i,j)\})$, and $\beta_4 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i+1,k-1]} \cup \mathcal{B}_{[k,l]} \cup \{(i,j)\})$. The first term in the recursion takes care of the case where (i,j) is the only base pair in $[i,j]$, i.e. (i,j) closes a hairpin loop. The second term handles the case where there is an interior loop (or a bulge or a stack) closed by (i,j) and (k,l). The third term takes care of all the structures where (i,j) closes a multiloop. To reduce complexity of the algorithm, the interior and bulge loop size can be limited to a maximum size of L (taken by default to be 30), by requiring that $l > j - L$ in the above recursion.

The final recursion, for computing $\mathcal{ZM}(x)$, is

$$\mathcal{ZM}_{i,j}(x) = \mathcal{ZM}_{i,j-1}(x) \cdot e^{\frac{-c}{RT}} \cdot x^{\alpha_0 n + \beta_0}$$

$$+ \sum_{\substack{(s_k, s_j) \in \mathbb{B}, \\ i \le k < j}} \left(\mathcal{ZB}_{k,j}(x) \cdot e^{\frac{-(b+c(k-i))}{RT}} \cdot x^{\alpha_5 n + \beta_5} \right. \tag{B.3}$$

$$\left. + \mathcal{ZM}_{i,k-1}(x) \cdot \mathcal{ZB}_{k,j}(x) \cdot e^{\frac{-b}{RT}} \cdot x^{\alpha_6 n + \beta_6} \right)$$

where $\alpha_5 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[k,j]})$, $\beta_5 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[k,j]})$, $\alpha_6 = d_{\mathrm{BP}}(\mathcal{A}_{[i,j]}, \mathcal{A}_{[i,k-1]} \cup \mathcal{A}_{[k,j]})$, and $\beta_6 = d_{\mathrm{BP}}(\mathcal{B}_{[i,j]}, \mathcal{B}_{[i,k-1]} \cup \mathcal{B}_{[k,j]})$. Note that since $\mathcal{ZM}_{i,j}(x)$ computes the partition function

contribution under the assumption that $[i,j]$ is part of a multiloop, there will be exactly one stem-loop structure in this region (the $\mathcal{ZB}(x)$ term) or more than one (the $\mathcal{ZM}(x)-\mathcal{ZB}(x)$ term). Justification of recursions (B.1), (B.2), and (B.3) follow by induction, as in the proof of Theorem 4.1.

B.2   Proof for Theorem 4.1

Proof. Recall that if F is an arbitrary polynomial [resp. analytic] function, then $[x^{rn+s}]F(x)$ denotes the coefficient of monomial x^{rn+s} in the Taylor expansion of $F(x)$. For instance, in equation (4.13), $[x^{rn+s}]p(x) = p_{rn+s}$, and in equation (4.3), $[x^{rn+s}]\mathcal{Z}(x) = z_{rn+s}$.

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

$$[x^{rn+s}]\mathcal{Z}_{i,j}(x) = z_{rn+s}(i,j) = \mathbf{Z}_{i,j}^{rn+s}$$

$$= \mathbf{Z}_{i,j-1}^{(r-\alpha_0)n+(s-\beta_0)}$$

$$+ \sum_{k=i}^{j-1} \sum_{u_0+u_1=r-\alpha(k)} \sum_{v_0+v_1=s-\beta(k)} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathbf{Z}_{i,k-1}^{u_0 n+v_0} \cdot \mathbf{Z}_{k+1,j-1}^{u_1 n+v_1} \right)$$

$$= [x^{(r-\alpha_0)n+(s-\beta_0)}]\mathcal{Z}_{i,j-1}(x)$$

$$+ \sum_{k=i}^{j-1} \sum_{u_0+u_1=r-\alpha(k)} \sum_{v_0+v_1=s-\beta(k)} \left(e^{\frac{-E_0(k,j)}{RT}} \right.$$

$$\cdot \left([x^{u_0 n+v_0}]\mathcal{Z}_{i,k-1}(x)) \cdot ([x^{u_1 n+v_1}]\mathcal{Z}_{k+1,j-1}(x)) \right)$$

$$= [x^{(r-\alpha_0)n+(s-\beta_0)}]\mathcal{Z}_{i,j-1}(x)$$

$$+ \sum_{k=i}^{j-1} \sum_{u_0+u_1=r-\alpha(k)} \sum_{v_0+v_1=s-\beta(k)} \left(e^{\frac{-E_0(k,j)}{RT}} \right. \tag{B.4}$$

$$\cdot [x^{(u_0+u_1)n+(v_0+v_1)}]\left(\mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \right) \right)$$

$$= [x^{(r-\alpha_0)n+(s-\beta_0)}]\mathcal{Z}_{i,j-1}(x)$$

$$+ \sum_{k=i}^{j-1} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot [x^{(r-\alpha(k))n+(s-\beta(k))}]\left(\mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \right) \right)$$

$$= [x^{rn+s}]\left(\mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n+\beta_0} \right)$$

$$+ \sum_{k=i}^{j-1} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot [x^{rn+s}]\left(\mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \cdot x^{\alpha(k)n+\beta(k)} \right) \right)$$

$$= [x^{rn+s}]\left(\mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n+\beta_0} \right.$$

$$+ \sum_{k=i}^{j-1} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \cdot x^{\alpha(k)n+\beta(k)} \right) \right)$$

$\square$

B.3 Proof for Lemma 4.4

Proof. The lemma states that if the base pair distance d_0 between reference structures $\mathcal{A}, \mathcal{B}$ is even, then $\mathcal{Z}(-\alpha) = \mathcal{Z}(\alpha)$, while if the distance is odd, then $\mathcal{Z}(-\alpha) = -\mathcal{Z}(\alpha)$. Suppose first that d_0 is even. By Lemma 4.2, $\mathcal{Z}(x) = z_0 + z_2 x^2 + z_4 x^4 + \cdots + z_{M-2} x^{M-2)}$, and so $\mathcal{Z}(-\alpha) = \mathcal{Z}(\alpha)$. Suppose now that d_0 is odd. By Lemma 4.2, $Z(x) = z_1 x^1 + z_3 x^3 + z^5 x^5 \cdots + z_{M-1} x^{M-1}$, and so $\mathcal{Z}(-\alpha) = -\mathcal{Z}(\alpha)$. □

B.4 Proof for Lemma 4.5

Proof. Recall Euler's formula in complex analysis: $\exp(ix) = \cos(x) + i\sin(x)$. As well, recall that $\sin(\pi) = 0$, $\cos(\pi) = -1$, and the trigonometric addition formulas:

$$\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$$

$$\sin(\alpha - \beta) = \sin(\alpha)\cos(\beta) - \sin(\beta)\cos(\alpha).$$

$$(B.5)$$

Then we have

$$v^{M_0-k} = \exp\left(\frac{2\pi i(M_0 - k)}{M}\right)$$

$$= \cos\left(\frac{2\pi(M_0 - k)}{M}\right) + i\sin\left(\frac{2\pi(M_0 - k)}{M}\right)$$

$$= \cos\left(\pi - \frac{2\pi k}{M}\right) + i\sin\left(\pi - \frac{2\pi k}{M}\right)$$

$$= \left[\cos(\pi)\cos\left(\frac{2\pi k}{M}\right) + \sin(\pi)\sin\left(\frac{2\pi k}{M}\right)\right]$$

$$+ i\left[\sin(\pi)\cos\left(\frac{2\pi k}{M}\right) - \sin\left(\frac{2\pi k}{M}\right)\cos(\pi)\right] \tag{B.6}$$

$$= -\cos\left(\frac{2\pi k}{M}\right) + i\sin\left(\frac{2\pi k}{M}\right)$$

$$= -1\left[\cos\left(\frac{2\pi k}{M}\right) - i\sin\left(\frac{2\pi k}{M}\right)\right]$$

$$= -1 \cdot \overline{\cos\left(\frac{2\pi k}{M}\right) + i\sin\left(\frac{2\pi k}{M}\right)}$$

$$= -1 \cdot \overline{\exp\left(\frac{2\pi i k}{M}\right)} = -\overline{v^k}.$$

It follows that $v^{M_0-k} = -\overline{v^k}$, so $v^k = \overline{-v^{(M_0-k)}} = -\overline{v^{(M_0-k)}}$.

This completes the proof of the lemma. □

Bibliography

[1] F. Crick, "Central dogma of molecular biology," *Nature*, 1970.

[2] J. M. Coffin, S. H. Hughes, H. E. Varmus, and A. B. Rabson, "Synthesis and processing of viral RNA," 1997.

[3] J. Berretta and A. Morillon, "Pervasive transcription constitutes a new level of eukaryotic genome regulation," *EMBO reports*, 2009.

[4] M. B. Clark, P. P. Amaral, F. J. Schlesinger, M. E. Dinger, R. J. Taft, J. L. Rinn, C. P. Ponting, P. F. Stadler, K. V. Morris, A. Morillon, J. S. Rozowsky, M. B. Gerstein, C. Wahlestedt, Y. Hayashizaki, P. Carninci, T. R. Gingeras, and J. S. Mattick, "The reality of pervasive transcription." *PLoS biology*, vol. 9, pp. e1000625–discussion e1001102, July 2011.

[5] T. H. Jensen, A. Jacquier, and D. Libri, "Dealing with pervasive transcription," *Molecular cell*, 2013.

[6] F. F. Costa, "Non-coding RNAs: new players in eukaryotic biology," *Gene*, 2005.

[7] A. Fire, S. Q. Xu, M. K. Montgomery, S. A. Kostas, and S. E. Driver, "Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans," *nature*, 1998.

[8] A. P. McCaffrey, L. Meuse, T. Pham, and D. S. Conklin, "Gene expression: RNA interference in adult mice," *Nature*, 2002.

[9] G. J. Hannon, "RNA interference," *Nature*, 2002.

[10] L. He and G. J. Hannon, "MicroRNAs: small RNAs with a big role in gene regulation," *Nature Reviews Genetics*, 2004.

[11] E. Nudler and A. S. Mironov, "The riboswitch control of bacterial metabolism," *Trends in biochemical sciences*, 2004.

[12] M. Mandal and R. R. Breaker, "Gene regulation by riboswitches," *Nature Reviews Molecular Cell Biology*, 2004.

[13] K. Kruger, P. J. Grabowski, A. J. Zaug, J. Sands, D. E. Gottschling, and T. R. Cech, "Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena.," *Cell*, vol. 31, pp. 147–157, Nov. 1982.

[14] T. R. Cech, "Self-splicing of group I introns," *Annual review of biochemistry*, 1990.

[15] E. A. Doherty and J. A. Doudna, "Ribozyme structures and mechanisms," *Annual review of biophysics and …*, 2001.

[16] W. Gilbert, "Origin of life: The RNA world," *Nature*, 1986.

[17] P. Yakovchuk, E. Protozanova, and M. D. Frank-Kamenetskii, "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix.," *Nucleic acids research*, vol. 34, no. 2, pp. 564–574, 2006.

[18] J. R. Fresco, B. M. Alberts, and P. Doty, "Some molecular details of the secondary structure of ribonucleic acid.," *Nature*, vol. 188, pp. 98–101, Oct. 1960.

[19] M. S. Waterman, "Secondary structure of single-stranded nucleic acids," *Adv math suppl studies*, 1978.

[20] R. Nussinov, G. Pieczenik, and J. R. Griggs, "Algorithms for loop matchings," *SIAM Journal on Applied …*, 1978.

[21] *On the theory of dynamic programming*, 1952.

[22] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.," *Nucleic acids research*, vol. 9, pp. 133–148, Jan. 1981.

[23] M. Zuker, "On finding all suboptimal foldings of an RNA molecule," *Science (New York, N.Y.)*, vol. 244, pp. 48–52, Apr. 1989.

[24] J. McCaskill, "The equilibrium partition function and base pair binding probabilities for RNA secondary structure," *Biopolymers*, vol. 29, pp. 1105–1119, 1990.

[25] D. Matthews, J. Sabina, M. Zuker, and D. Turner, "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure," *J. Mol. Biol.*, vol. 288, pp. 911–940, 1999.

[26] D. H. Turner and D. H. Mathews, "NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure," *Nucleic Acids Research*, 2009.

[27] E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy, "Infernal 1.0: inference of RNA alignments," *Bioinformatics*, 2009.

[28] E. P. Nawrocki and S. R. Eddy, "Infernal 1.1: 100-fold faster RNA homology searches," *Bioinformatics*, vol. 29, pp. 2933–2935, Oct. 2013.

[29] M. D. Ermolaeva, H. G. Khalak, O. White, H. O. Smith, and S. L. Salzberg, "Prediction of transcription terminators in bacterial genomes," *Journal of molecular biology*, vol. 301, pp. 27–33, Aug. 2000.

[30] 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.*, vol. 6, p. 26, 2011.

[31] J. Gorodkin, L. J. Heyer, and G. D. Stormo, "Finding the most significant common sequence and structure motifs in a set of RNA sequences.," *Nucleic Acids Research*, vol. 25, pp. 3724–3732, Sept. 1997.

[32] J. H. Havgaard, E. Torarinsson, and J. Gorodkin, "Fast Pairwise Structural RNA Alignments by Pruning of the Dynamical Programming Matrix," *PLoS computational biology*, vol. 3, no. 10, p. e193, 2007.

[33] P. Bengert and T. Dandekar, "Riboswitch finder–a tool for identification of riboswitch RNAs," *Nucleic Acids Research*, vol. 32, pp. W154–W159, July 2004.

[34] T.-H. Chang, H.-D. Huang, L.-C. Wu, C.-T. Yeh, B.-J. Liu, and J.-T. Horng, "Computational identification of riboswitches based on RNA conserved functional sequences and conformations.," *RNA (New York, N.Y.)*, vol. 15, pp. 1426–1430, July 2009.

[35] C. Abreu-Goodger and E. Merino, "RibEx: a web server for locating riboswitches and other conserved bacterial regulatory elements.," *Nucleic Acids Research*, vol. 33, pp. W690–2, July 2005.

[36] Z. Yao, Z. Weinberg, and W. L. Ruzzo, "CMfinder–a covariance model based RNA motif finding algorithm," *Bioinformatics*, vol. 22, pp. 445–452, Feb. 2006.

[37] K. Darty, A. Denise, and Y. Ponty, "VARNA: Interactive drawing and editing of the RNA secondary structure.," *Bioinformatics (Oxford, England)*, vol. 25, pp. 1974–1975, Aug. 2009.

[38] É. Fusy and P. Clote, "Combinatorics of locally optimal RNA secondary structures," *Journal of mathematical biology*, vol. 68, pp. 341–375, Dec. 2012.

[39] B. Voß, R. Giegerich, and M. Rehmsmeier, "Complete probabilistic analysis of RNA shapes.," *BMC biology*, vol. 4, no. 1, p. 5, 2006.

[40] W. A. Lorenz, Y. Ponty, and P. Clote, "Asymptotics of RNA Shapes," *Journal of Computational Biology*, vol. 15, no. 1, pp. 31–63, 2008.

[41] M. Mandal, B. Boese, J. E. Barrick, W. C. Winkler, and R. R. Breaker, "Riboswitches Control Fundamental Biochemical Pathways in Bacillus subtilis and Other Bacteria," *Cell*, vol. 113, pp. 577–586, May 2003.

[42] S. Janssen and R. Giegerich, "The RNA shapes studio.," *Bioinformatics (Oxford, England)*, vol. 31, pp. 423–425, Feb. 2015.

[43] M. Mandal and R. R. Breaker, "Gene regulation by riboswitches," *Nature Reviews Molecular Cell Biology*, vol. 5, pp. 451–463, June 2004.

[44] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner, "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure," *Journal of molecular biology*, vol. 288, pp. 911–940, May 1999.

[45] M. Mandal, B. Boese, J. Barrick, W. Winkler, and R. Breaker, "Riboswitches control fundamental biochemical pathways in Bacillus subtilis and other bacteria," *Cell*, vol. 113(5), pp. 577–586, 2003.

[46] A. Serganov, Y.-R. Yuan, O. Pikovskaya, A. Polonskaia, L. Malinina, A. T. Phan, C. Höbartner, R. Micura, R. R. Breaker, and D. J. Patel, "Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs.," *Chemistry & biology*, vol. 11, pp. 1729–1741, Dec. 2004.

[47] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: the European molecular biology open software suite," *Trends in Genetics*, 2000.

[48] D. Sankoff, "Simultaneous solution of the RNA folding, alignment and protosequence problems," *SIAM Journal on Applied Mathematics*, 1985.

[49] K. A. Wilkinson, E. J. Merino, and K. M. Weeks, "Selective 2-hydroxyl acylation analyzed by primer extension (SHAPE): quantitative RNA structure analysis at single nucleotide resolution," *Nature protocols*, 2006.

[50] M. Mandal and R. R. Breaker, "Adenine riboswitches and gene activation by disruption of a transcription terminator," *Nature structural & molecular biology*, vol. 11, pp. 29–35, Dec. 2003.

[51] E. Freyhult, V. Moulton, and P. Clote, "Boltzmann probability of RNA structural neighbors and riboswitch detection.," *Bioinformatics*, vol. 23, pp. 2054–2062, August 2007.

[52] P. Clote, F. Lou, and W. Lorenz, "Maximum expected accuracy structural neighbors of an RNA secondary structure," *BMC Bioinformatics*, vol. 13, p. S6, April 2012.

[53] Y. Ding and C. E. Lawrence, "A statistical sampling algorithm for RNA secondary structure prediction.," *Nucleic. Acids. Res.*, vol. 31, pp. 7280–7301, 2003.

[54] 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.*, vol. 7, no. 12, p. e50506, 2012.

[55] E. S. Haas, D. P. Morse, J. W. Brown, F. J. Schmidt, and N. R. Pace, "Long-range structure in ribonuclease P RNA.," *Science (New York, N.Y.)*, vol. 254, pp. 853–856, Nov. 1991.

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

[57] E. Freyhult, V. Moulton, and P. Gardner, "Predicting RNA structure using mutual information." *Appl. Bioinformatics*, vol. 4, no. 1, pp. 53–59, 2005.

[58] 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*, vol. 77, no. 11, pp. 6309–6313, 1980.

[59] N. Higham, "The numerical stability of barycentric Lagrange interpolation," *IMA J. Numer. Anal.*, vol. 24, pp. 547–556, 2004.

[60] T. Cormen, C. Leiserson, and R. Rivest, *Algorithms*. McGraw-Hill, 1990. 1028 pages.

[61] P. G. Wolynes, "Energy landscapes and solved protein-folding problems." *Philos. Transact. A. Math. Phys. Eng. Sci.*, vol. 363, pp. 453–464, February 2005.

[62] 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.*, vol. 21, pp. 167–195, March 1995.

[63] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool.," *J. Mol. Biol.*, vol. 215, pp. 403–410, October 1990.

[64] 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.*, vol. 39, pp. D141–D145, January 2011.

[65] C. Flamm, W. Fontana, I. Hofacker, and P. Schuster, "RNA folding at elementary step resolution," *RNA*, vol. 6, pp. 325–338, 2000.

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

[67] K. Gerdes, A. P. Gultyaev, T. Franch, K. Pedersen, and N. D. Mikkelsen, "Antisense RNA-regulated programmed cell death.," *Annu. Rev. Genet.*, vol. 31, pp. 1–31, 1997.

[68] K. Lecuyer and D. Crothers, "The Leptomonas collosoma spliced leader RNA can switch between two alternate structural forms," *Biochemistry*, vol. 32(20), pp. 5301–5311, 1993.

[69] R. Lorenz, C. Flamm, and I. L. Hofacker, "2D Projections of RNA Folding Landscapes.," *GCB*, 2009.

[70] E. Senter, I. Dotu, and P. Clote, "RNA folding pathways and kinetics using 2D energy landscapes," *J Math Biol*, 2014.

[71] R. Lorenz, C. Flamm, and I. Hofacker, "2D projections of RNA folding landscapes," in *German Conference on Bioinformatics 2009* (I. Grosse, S. Neumann, S. Posch, F. Schreiber, and P. Stadler, eds.), vol. 157 of *Lecture Notes in Informatics*, pp. 11–20, 2009.

[72] L. M. Hochrein, M. Schwarzkopf, M. Shahgholi, P. Yin, and N. A. Pierce, "Conditional Dicer substrate formation via shape and sequence transduction with small conditional RNAs.," *J. Am. Chem. Soc.*, vol. 135, pp. 17322–17330, November 2013.

[73] M. Wachsmuth, S. Findeiss, N. Weissheimer, P. F. Stadler, and M. Morl, "De novo design of a synthetic riboswitch that regulates transcription termination.," *Nucleic. Acids. Res.*, vol. 41, pp. 2541–2551, February 2013.

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

[75] J. N. Zadeh, B. R. Wolfe, and N. A. Pierce, "Nucleic acid sequence design via efficient ensemble defect optimization.," *J. Comput. Chem.*, vol. 32, pp. 439–452, February 2011.

[76] A. Gruber, R. Lorenz, S. Bernhart, R. Neubock, and I. Hofacker, "The vienna rna web-suite," *Nucleic Acids Research*, vol. 36, pp. 70–74, 2008.

[77] J. Garcia-Martin, P. Clote, and I. Dotu, "RNAiFold: A constraint programming algorithm for RNA inverse folding and molecular design," *J Bioinform Comput Biol*, vol. 11, no. 2, p. 1350001, 2013. in press.

[78] J. A. Garcia-Martin, P. Clote, and I. Dotu, "RNAiFold: a web server for RNA inverse folding and molecular design.," *Nucleic. Acids. Res.*, vol. 41, pp. W465–W470, July 2013.

[79] A. Bujotzek, M. Shan, R. Haag, and M. Weber, "Towards a rational spacer design for bivalent inhibition of estrogen receptor.," *J. Comput. Aided. Mol. Des.*, vol. 25, pp. 253–262, March 2011.

[80] C. Fasting, C. A. Schalley, M. Weber, O. Seitz, S. Hecht, B. Koksch, J. Dernedde, C. Graf, E. W. Knapp, and R. Haag, "Multivalency as a chemical organization and action principle.," *Angew. Chem. Int. Ed. Engl.*, vol. 51, pp. 10472–10498, October 2012.

[81] E. Senter and P. Clote, "Fast, Approximate Kinetics of RNA Folding," *Journal of computational biology : a journal of computational molecular cell biology*, vol. 22, pp. 124–144, Feb. 2015.

[82] C. Hobartner and R. Micura, "Bistable secondary structures of small RNAs and their structural probing by comparative imino proton NMR spectroscopy.," *J. Mol. Biol.*, vol. 325, pp. 421–431, January 2003.

[83] J. R. Vieregg and I. Tinoco, "Modelling RNA folding under mechanical tension.," *Mol. Phys.*, vol. 104, pp. 1343–1352, April 2006.

[84] C. Flamm, I. Hofacker, P. Stadler, and M. Wolfinger, "Barrier trees of degenerate landscapes," *Z. Phys. Chem.*, vol. 216, pp. 155–173, 2002.

[85] J. Huang and B. Voss, "Analysing RNA-kinetics based on folding space abstraction.," *BMC. Bioinformatics*, vol. 15, p. 60, 2014.

[86] P. Zhao, W. Zhang, and S. J. Chen, "Cotranscriptional folding kinetics of ribonucleic acid secondary structures.," *J. Chem. Phys.*, vol. 135, p. 245101, December 2011.

[87] S. Morgan and P. Higgs, "Barrier heights between ground states in a model of RNA secondary structure," *J. Phys. A: Math. Gen.*, vol. 31, pp. 3153–3170, 1998.

[88] C. Flamm, I. L. Hofacker, S. Maurer-Stroh, P. F. Stadler, and M. Zehl, "Design of multistable RNA molecules.," *RNA.*, vol. 7, pp. 254–265, February 2001.

[89] B. A. Shapiro, D. Bengali, W. Kasprzak, and J. C. Wu, "RNA folding pathway functional intermediates: their prediction and analysis.," *J. Mol. Biol.*, vol. 312, pp. 27–44, September 2001.

[90] M. Kuchařík, I. L. Hofacker, P. F. Stadler, and J. Qin, "Basin Hopping Graph: a computational framework to characterize RNA folding landscapes.," *Bioinformatics*, vol. 30, pp. 2009–2017, July 2014.

[91] C. Thachuk, J. Manuch, A. Rafiey, L. A. Mathieson, L. Stacho, and A. Condon, "An algorithm for the energy barrier problem without pseudoknots and temporary arcs.," *Pac Symp Biocomput.*, vol. 0, no. O, p. O, 2010:108-19.

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

[93] S. Wuchty, W. Fontana, I. Hofacker, and P. Schuster, "Complete suboptimal folding of RNA and the stability of secondary structures," *Biopolymers*, vol. 49, pp. 145–164, 1999.

[94] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato, "Using motion planning to study RNA folding kinetics.," *J. Comput. Biol.*, vol. 12, no. 6, pp. 862–881, 2005.

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

[96] D. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J Comp Phys*, vol. 22, no. 403, pp. 403–434, 1976.

[97] I. Aviram, I. Veltman, A. Churkin, and D. Barash, "Efficient procedures for the numerical simulation of mid-size RNA kinetics.," *Algorithms. Mol. Biol.*, vol. 7, no. 1, p. 24, 2012.

[98] E. C. Dykeman, "An implementation of the Gillespie algorithm for RNA kinetics with logarithmic time update," *Nucleic Acids Research*, vol. 43, pp. 5708–5715, July 2015.

[99] A. Xayaphoummine, T. Bucher, and H. Isambert, "Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots.," *Nucleic. Acids. Res.*, vol. 33, pp. W605–W610, July 2005.

[100] L. V. Danilova, D. D. Pervouchine, A. V. Favorov, and A. A. Mironov, "RNAKinetics: a web server that models secondary structure kinetics of an elongating RNA.," *J. Bioinform. Comput. Biol.*, vol. 4, pp. 589–596, April 2006.

[101] M. Geis, C. Flamm, M. T. Wolfinger, A. Tanzer, I. L. Hofacker, M. Middendorf, C. Mandl, P. F. Stadler, and C. Thurner, "Folding kinetics of large RNAs.," *J. Mol. Biol.*, vol. 379, pp. 160–173, May 2008.

[102] D. Lai, J. R. Proctor, and I. M. Meyer, "On the importance of cotranscriptional RNA structure formation.," *RNA.*, vol. 19, pp. 1461–1473, November 2013.

[103] D. H. Turner and D. H. Mathews, "NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure.," *Nucleic. Acids. Res.*, vol. 38, pp. D280–D282, January 2010.

[104] J. W. Anderson, P. A. Haas, L. A. Mathieson, V. Volynkin, R. Lyngso, P. Tataru, and J. Hein, "Oxfold: kinetic folding of RNA using stochastic context-free grammars and evolutionary information.," *Bioinformatics*, vol. 29, pp. 704–710, March 2013.

[105] C. Meyer, "The role of the group inverse in the theory of finite Markov chains," *SIAM Rev.*, vol. 17, no. 46, pp. 443–464, 1975.

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

[107] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1092, 1953.

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

[109] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, pp. 223–230, 1973.

[110] J. N. Franklin, *Matrix Theory.* Dover Publications, Mineola, New York, 2000. 292 pages.

[111] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

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

[113] X. Tang, S. Thomas, L. Tapia, D. P. Giedroc, and N. M. Amato, "Simulating RNA folding kinetics on approximated energy landscapes.," *J. Mol. Biol.*, vol. 381, pp. 1055–1067, September 2008.

[114] I. Gusarov and E. Nudler, "The mechanism of intrinsic transcription termination.," *Molecular cell*, vol. 3, pp. 495–504, Apr. 1999.

[115] W. S. Yarnell and J. W. Roberts, "Mechanism of intrinsic transcription termination and antitermination.," *Science (New York, N.Y.)*, vol. 284, pp. 611–615, Apr. 1999.

[116] J. E. Barrick and R. R. Breaker, "The distributions, mechanisms, and structures of metabolite-binding riboswitches.," *Genome Biology*, vol. 8, no. 11, p. R239, 2007.

[117] E. P. Nawrocki, S. W. Burge, A. Bateman, J. Daub, R. Y. Eberhardt, S. R. Eddy, E. W. Floden, P. P. Gardner, T. A. Jones, J. Tate, and R. D. Finn, "Rfam 12.0: updates to the RNA families database.," *Nucleic Acids Research*, vol. 43, pp. D130–7, Jan. 2015.

[118] Y. Ding, W. A. Lorenz, I. Dotu, E. Senter, and P. Clote, "Computing the Probability of RNA Hairpin and Multiloop Formation," *Journal of computational biology : a journal of computational molecular cell biology*, vol. 21, pp. 201–218, Mar. 2014.