Boston College

Doctoral Thesis

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

Author:
Evan Senter

Supervisor:
Dr. Peter Clote

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

in the

Clote Lab
Department of Biology

Thursday 9th July, 2015

Contents

Chapter 1

FFTbor

1.1 Introduction

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

1.1.1 Organization

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

1.2 Formalization of the problem

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

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}, \tag{1.1}$$

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 (1.1). By leveraging complex nth roots of unity in conjunction with the inverse Discrete Fourier Transform the we subvert numeric instability issues observed with both Lagrange interpolation and Gaussian elimination.

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

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

$$\mathbb{B} = \{(\mathrm{A}, \mathrm{U}), (\mathrm{U}, \mathrm{A}), (\mathrm{G}, \mathrm{C}), (\mathrm{C}, \mathrm{G}), (\mathrm{G}, \mathrm{U}), (\mathrm{U}, \mathrm{G})\}. \tag{1.2}$$

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

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

$$d_{\text{BP}}(\mathcal{S},\, \mathcal{T}) = |\mathcal{S} \cup \mathcal{T}| - |\mathcal{S} \cap \mathcal{T}|. \tag{1.3}$$

Given this definition of distance, two structures $\mathcal{S}$ and $\mathcal{T}$ are said to be *k-neighbors* if $d_{\text{BP}}(\mathcal{S},\, \mathcal{T}) = k$. 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. $\mathcal{S}_{[i,j]} = \{(x,y) \,:\, i \le x < y \le j, (x,y) \in \mathcal{S}\}$.

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

$$d_{\text{BP}}(\mathcal{S}_{[i,j]},\, \mathcal{T}_{[i,j]}) = |\{(x,y) : i \le x < y \le j, (x,y) \in \mathcal{S} - \mathcal{T} \text{or} (x,y) \in \mathcal{T} - \mathcal{S}\}| = k. \tag{1.4}$$

1.3 Derivation of the `FFTbor` algorithm

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

$$\mathbf{Z}^k = \mathbf{Z}^k_{1,n} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\text{BP}}(\mathcal{S},\mathcal{S}^*)=k}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{1.5}$$

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

$$\mathbf{Z} = \mathbf{Z}_{1,n} = \sum_{k=0}^{n} \mathbf{Z}^k_{1,n} \tag{1.6}$$

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

$$d_{\mathrm{BP}}(\mathcal{S}^*, \mathcal{S}) \leq |\mathcal{S}^*| + \lfloor \frac{n-\theta}{2} \rfloor \leq n. \tag{1.7}$$

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

$$p(k) = \frac{\mathbf{Z}^k_{1,n}}{\mathbf{Z}_{1,n}}. \tag{1.8}$$

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

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

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

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

$$\mathbf{Z}^0_{i,j} = 1, \text{ for } i \leq j, \tag{1.9}$$

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

$$\mathbf{Z}^k_{i,j} = 0, \text{ for } k > 0, i \leq j \leq i + \theta, \tag{1.10}$$

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

$$\mathbf{Z}^k_{i,j} = \mathbf{Z}^{k-b_0}_{i,j-1} + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\leq r<j}} \sum_{w+w'=k-b(r)} \exp(-E_0(r,j)/RT) \cdot \mathbf{Z}^w_{i,r-1}\mathbf{Z}^{w'}_{r+1,j-1}, \tag{1.11}$$

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

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

$$\mathcal{Z}(x) = \sum_{k=0}^{n} c_k x^k \tag{1.12}$$

where coefficients $c_k = \mathbf{Z}^k_{1,n}$. Moreover, because of equation (1.7) and the fact that the minimum number of unpaired bases in a hairpin loop θ is 3, we know that $c_n = 0$, so that $\mathcal{Z}(x)$ is a polynomial of degree strictly less than n. If we evaluate the polynomial $\mathcal{Z}(x)$ for n distinct values

$$\mathcal{Z}(a_1) = y_1, \ldots, \mathcal{Z}(a_n) = y_n, \tag{1.13}$$

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

$$P_k(x) = \frac{\prod_{i \neq k}(x - x_i)}{\prod_{i \neq k}(x_k - x_i)}. \tag{1.14}$$

Since the polynomials $P_k(x)$ can be explicitly computed, it follows that we can compute the coefficients c_k of polynomial $\mathcal{Z}(x)$. As we describe below, the evaluation of $\mathcal{Z}(x)$ for a fixed value of x can be done in time $O(n^3)$ and space $O(n^2)$. It follows that the coefficients $c_k = \mathbf{Z}_{1,n}^k$ can be computed after n evaluations of $\mathcal{Z}(x)$, where the space for each evaluation of $\mathcal{Z}(x)$ is re-used; hence these evaluations can be performed in time $O(n^4)$ and space $O(n^2)$. Finally, Lagrange interpolation is clearly computable in time $O(n^3)$. Although this approach is theoretically sound, there are severe numerical stability issues related to the interpolation method **?**, the choice of values $a_1, \ldots, a_n$ in the interpolation, and floating point arithmetic (round-off error) related to the astronomically large values of the partition functions $\mathbf{Z}_{1,n}^k$, for $0 \leq k < n$. After many unsuccessful approaches including scaling we obtained excellent results by interpolating the polynomial $p(x)$, defined in equation (1.1), rather than the polynomial $\mathcal{Z}(x)$, defined in equation (1.12), and performing interpolation with the Fast Fourier Transform (FFT) **?** where $\alpha_0, \ldots, \alpha_{n-1}$ are chosen to be complex nth roots of unity, $\alpha_k = e^{2\pi ik/n}$. One advantage of the FFT is that interpolation can be performed in $O(n \log n)$ time, rather than the cubic time required by using the Lagrange formula (1.14) or by Gaussian elimination. Fewer numerical operations implies increased numerical stability in our application.

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

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

$$\mathbf{Z}_{1,n}^k = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{S}^*)=k}} e^{\frac{-E_0(\mathcal{S})}{RT}} \tag{1.15}$$

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

$$\mathbf{Z}^k_{i,j} = \mathbf{Z}^{k-b_0}_{i,j-1} \;+ \sum_{\substack{s_r s_j \in \mathbb{B},\\ i \leq r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \sum_{w+w'=k-b(r)} \mathbf{Z}^w_{i,r-1} \mathbf{Z}^{w'}_{r+1,j-1} \right) \tag{1.16}$$

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

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k(i,j)\, x^k \tag{1.17}$$

where

$$c_k(i,j) \;=\; \mathbf{Z}^k_{i,j} \;=\; \sum_{\substack{\mathcal{S} \text{ such that}\\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{S}^*_{[i,j]})=k}} e^{\frac{-E_0(\mathcal{S})}{RT}}. \tag{1.18}$$

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

Theorem 1.1. *Let $s_1, \ldots, s_n$ be a given RNA sequence. For any integers $1 \leq i \leq j \leq n$, let*

$$\mathcal{Z}_{i,j}(x) = \sum_{k=0}^{n} c_k\, x^k \tag{1.19}$$

where

$$c_k(i,j) = \mathbf{Z}^k_{i,j}. \tag{1.20}$$

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

$$\mathcal{Z}_{i,j}(x) = \mathcal{Z}_{i,j-1}(x) \cdot x^{b_0} + \sum_{\substack{s_r s_j \in \mathbb{B}, \\ i \leq r < j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(x) \cdot \mathcal{Z}_{r+1,j-1}(x) \cdot x^{b(r)} \right). \tag{1.21}$$

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

Proof. First, some notation is necessary. Recall that if F is an arbitrary polynomial [resp. analytic] function, then $[x^k]F(x)$ denotes the coefficient of x^k [resp. the kth Taylor coefficient in the Taylor expansion of F]. For instance, in equation (1.1), $[x^k]p(x) = p_k$, and in equation (1.12), $[x^k]\mathcal{Z}(x) = c_k(i,j)$.

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

$$\begin{aligned}
[x^k]\mathcal{Z}_{i,j}(x) &= c_k(i,j) = \mathbf{Z}^k_{i,j} \\
&= \mathbf{Z}^{k-b_0}_{i,j-1} + \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \mathbf{Z}^{k_0}_{i,r-1} \cdot \mathbf{Z}^{k_1}_{r+1,j-1} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot \left\{[x^{k_0}]\mathcal{Z}_{i,r-1}(x)\right\} \cdot \left\{[x^{k_1}]\mathcal{Z}_{r-1,j-1}(x)\right\} \\
&= [x^{k-b_0}]\mathcal{Z}_{i,j-1}(x) \\
&+ \sum_{r=i}^{j-1} \sum_{k_0+k_1=k-b(r)} e^{\frac{-E_0(r,j)}{RT}} \cdot [x^{k_0+k_1}]\mathcal{Z}_{i,r-1}(x)\mathcal{Z}_{r-1,j-1}(x).
\end{aligned} \tag{1.22}$$

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

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

$$\mathcal{Z}_{i,j}(\alpha) = \mathcal{Z}_{i,j-1}(\alpha) \cdot \alpha^{b_0} + \sum_{\substack{(s_r,s_j)\in\mathbb{B},\\ i\le r<j}} \left(e^{\frac{-E_0(r,j)}{RT}} \cdot \mathcal{Z}_{i,r-1}(\alpha) \cdot \mathcal{Z}_{r+1,j-1}(\alpha) \cdot \alpha^{b(r)} \right). \quad (1.23)$$

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

$$\mathbf{Y} = (y_0, \ldots, y_{n-1})^{\mathrm{T}} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} \quad (1.24)$$

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 $O(n^2)$.

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

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

$$V_n = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \ldots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{pmatrix} \quad (1.25)$$

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

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = V_n \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} \tag{1.26}$$

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

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \, \omega^{-kj} \tag{1.27}$$

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

Since we defined $\mathbf{Y}$ in equation (1.24) 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 $\alpha_k = \omega^k e^{2\pi ik/n}$, it follows that the coefficients $c_k = \mathbf{Z}^k_{1,n}$ in the polynomial $\mathcal{Z}(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ defined in equation (1.12) can be computed, at least in principle, by using the Fast Fourier Transform. It turns out, however, that the values of $\mathbf{Z}^k_{1,n}$ are so astronomically large, that the ensuing numerical instability makes even this approach infeasible for values of n that exceed 56 (data not shown). Nevertheless, our approach can be modified as follows. Define $\mathbf{Y}$ by $\mathbf{Y} = (y_1, \ldots, y_n)^{\mathrm{T}}$, where $y_1 = \frac{\mathcal{Z}(\alpha_1)}{\mathcal{Z}(x)}, \ldots, y_n = \frac{\mathcal{Z}(\alpha_n)}{\mathcal{Z}(x)}$, and $\mathbf{Z}$ is the partition function defined in equation (1.6). Using the Fast Fourier Transform to compute the inverse Discrete Fourier Transform, it follows from equation (1.27) 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 (1.1). 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.

Algorithm 1 Pseudocode for `FFTbor`

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

function FFTBOR($\mathbf{s}$, $\mathcal{S}^*$, $m$)
    $n \leftarrow length(\mathbf{s})$
    for $k \leftarrow 0, n-1$ do
        $\omega_k \leftarrow \exp(\frac{2\pi ik}{n})$        ▷ Pre-compute all complex $n$th root of unity, where $i = \sqrt{-1}$
    end for
    for $k \leftarrow 0, n-1$ do
        $y_k \leftarrow 10^m \cdot \mathcal{Z}(\omega_k)/\mathbf{Z}$        ▷ Note that $\mathcal{Z}(\omega_0) = \mathbf{Z}$
    end for
    for $k \leftarrow 0, n-1$ do
        $p_k \leftarrow \frac{\sum_{j=0}^{n-1} a_j \omega^{-kj}}{n}$        ▷ Where $a_j$ comes from equation (1.27)
        $p_k \leftarrow \lfloor 10^m \cdot p_k \rfloor \cdot \frac{1}{10^m}$        ▷ Truncate to $m$ significant digits
    end for
    return $p_0, \ldots, p_{n-1}$        ▷ Return all $p_k$ for $0 \leq k < n$, from equation (1.8)
end function
```

1.4 Benchmarking and performance considerations

In this subsection, we show that we need only evaluate the polynomial $\mathcal{Z}(x)$, as defined in equation (1.12), for $n/2$ of the complex nth roots of unity. It is first necessary to recall the definition of complex conjugate. Recall that the complex conjugate of z is denoted by $\overline{z}$; i.e. if $z = a + bi$ where $a, b \in \mathbb{R}$ are real numbers and $i = \sqrt{-1}$, then $\overline{z} = a - bi$.

Lemma 1.2. *If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (1.12), 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*

$$\mathcal{Z}(a - bi) = A - Bi. \tag{1.28}$$

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

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

$$(a+bi)^m = \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (bi)^k \tag{1.29}$$

$$(bi)^k = \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ -ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases} \tag{1.30}$$

$$(a-bi)^m = \sum_{k=0}^{m} \binom{m}{k} a^{m-k} \cdot (-bi)^k \tag{1.31}$$

$$(-bi)^k = \begin{cases} b^k & \text{if } k \equiv 0 \bmod 4 \\ -ib^k & \text{if } k \equiv 1 \bmod 4 \\ -b^k & \text{if } k \equiv 2 \bmod 4 \\ ib^k & \text{if } k \equiv 3 \bmod 4 \end{cases} \tag{1.32}$$

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

□

Lemma 1.2 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`.

1.5 Coarse-grained kinetics with `FFTbor`

1.6 Riboswitch detection with `FFTbor`

Chapter 2

FFTbor2D

2.1 Introduction

In this chapter, we present the `FFTbor2D` algorithm and accompanying software. `FFTbor2D`, like `FFTbor` described in Chapter 1, is an algorithm which computes the paramerized partition function for an input RNA sequence `s`. `FFTbor2D` computes the two-dimensional coarse energy landscape for `s` given two compatible input secondary structures $\mathcal{A}$ and $\mathcal{B}$, where position (x, y) on the discrete energy landscape corresponds to the Boltzmann probability for those structures $\mathcal{S}$ which have $d_{\mathrm{BP}}((, \mathcal{S}), \mathcal{A}) = x$ and $d_{\mathrm{BP}}((, \mathcal{S}), \mathcal{B}) = y$ (where $d_{\mathrm{BP}}(, \mathrm{i})$ s as defined in equation 1.3). 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 **??**.

2.1.1 Organization

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

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

2.2 Derivation of the `FFTbor2D` algorithm

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

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

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

$$\mathbf{Z}_{1,n}^{x,y} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=x,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=y}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{2.1}$$

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

For any secondary structure $\mathcal{S}$ of $\mathbf{s}$, and any values $1 \leq i \leq j \leq n$, the restriction $\mathcal{S}_{[i,j]}$ is defined to be the collection of base pairs of $\mathcal{S}$, lying within interval $[i,j]$; i.e. $\mathcal{S}_{[i,j]} = \{(k,\ell) : i \leq k < \ell \leq j\}$. In **?**, Lorenz et al. generalized the dynamic programming recursions of our earlier work **?**, to yield recursions for the partition function $\mathbf{Z}_{i,j}^{x,y}$ in equation (2.1). In the context of the Nussinov model, $\mathbf{Z}_{i,j}^{x,y}$ is equal to

$$\mathbf{Z}_{i,j-1}^{x-\alpha_0,y-\beta_0} + \sum_{\substack{s_k s_j \in \mathbb{B}, \\ i \leq k < j}} \left(e^{\frac{-E_0(k,j)}{RT}} \sum_{u+u'=x-\alpha(k)} \sum_{v+v'=y-\beta(k)} \mathbf{Z}_{i,k-1}^{u,v} \cdot \mathbf{Z}_{k+1,j-1}^{u',v'} \right) \tag{2.2}$$

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

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

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

$$\mathcal{Z}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{2.3}$$

where (constant) coefficients

$$z_{rn+s} = \mathbf{Z}_{1,n}^{r,s} = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}} \tag{2.4}$$

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

$$\mathcal{Z}(a_0) = y_0, \ldots, \mathcal{Z}(a_{n^2-1}) = y_{n^2-1}, \tag{2.5}$$

then Lagrange polynomial interpolation (equation 1.14) guarantees that we can determine the coefficients c_{rn+s} of $\mathcal{Z}(x)$, for $0 \leq r, s < n$. Due to technical difficulties concerning numerical robustness observered while working on the `FFTbor` software (Chapter 1), 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 (2.2), can be used to compute the *polynomial* $\mathcal{Z}_{i,j}(x)$ defined by

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} z_{rn+s}(i,j) \cdot x^{rn+s} \ = \sum_{k=0}^{n^2-1} z_k(i,j) \cdot x^k \tag{2.6}$$

where

$$z_{rn+s}(i,j) \ = \ \mathbf{Z}^{r,s}_{i,j} \ = \sum_{\substack{\mathcal{S} \text{ such that} \\ d_{\mathrm{BP}}(\mathcal{S},\mathcal{A})=r,\, d_{\mathrm{BP}}(\mathcal{S},\mathcal{B})=s}} e^{\frac{-E(\mathcal{S})}{RT}}. \tag{2.7}$$

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

Theorem 2.1. *Let $s_1, \ldots, s_n$ be a given RNA sequence. For any integers $1 \leq i < j \leq n$, let*

$$\mathcal{Z}_{i,j}(x) = \sum_{r=0}^{n-1}\sum_{s=0}^{n-1} z_{rn+s} x^{rn+s} \tag{2.8}$$

where

$$z_{rn+s}(i,j) = \mathbf{Z}^{r,s}_{i,j}. \tag{2.9}$$

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

$$\begin{aligned} &\mathcal{Z}_{i,j-1}(x) \cdot x^{\alpha_0 n+\beta_0} + \\ &\sum_{\substack{s_k s_j \in \mathbb{B}, \\ i \leq k < j}} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(x) \cdot \mathcal{Z}_{k+1,j-1}(x) \cdot x^{\alpha(k)n+\beta(k)} \right) \end{aligned} \tag{2.10}$$

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 supplementary information.

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

$$\mathcal{Z}_{i,j-1}(\rho) \cdot \rho^{\alpha_0 n+\beta_0} + \sum_{\substack{(s_k,s_j)\in\mathbb{B},\\ i\leq k<j}} \left(e^{\frac{-E_0(k,j)}{RT}} \cdot \mathcal{Z}_{i,k-1}(\rho) \cdot \mathcal{Z}_{k+1,j-1}(\rho) \cdot \rho^{\alpha(k)n+\beta(k)} \right) \tag{2.11}$$

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

$$\mathbf{Y} = (y_0, \ldots, y_{n^2-1})^T = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n^2-1} \end{pmatrix} \tag{2.12}$$

where $y_0 = \mathcal{Z}(\alpha_0), \ldots, y_{n^2-1} = \mathcal{Z}(\alpha_{n^2-1})$ are thus quintic time $O(n^5)$ and quadratic space $O(n^2)$.

2.2.3 Polynomial interpolation

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

$$\mathbf{p}(x) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} \mathbf{p}_{rn+s} x^{r\cdot n+s} = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} \frac{z_{rn+s}}{Z} x^{r\cdot n+s}, \tag{2.13}$$

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 2.2.7. The following pseudocode describes how to compute the m most significant digits for probabilities $p_{rn+s} = \frac{\mathbf{Z}^{r,s}_{1,n}}{\mathbf{Z}}$. It is well-known that the FFT requires $O(N \log N)$

time to solve the inverse discrete Fourier transform for a polynomial of degree N. In our case, $N = n^2$, and so line 6 involving the FFT requires time $O(n^2 \log n)$.

The pseudocode for the algorithm to compute $\mathbf{p}(x)$ is given in Figure 1. In the next section, we explain a highly non-trivial improvement of this algorithm to reduce time by a factor of 4.

Algorithm for `FFTbor2D`
Input: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, and distinct secondary structures $\mathcal{A}, \mathcal{B}$ of $\mathbf{s}$, and integer m.
Output: Probabilities $p_{rn+s} = p(r,s) = \mathbf{Z}^{r,s}_{1,n}/\mathbf{Z}$ to m significant digits for $x, y = 0, \ldots, n-1$.
Let i denote $\sqrt{-1}$, $\alpha = \exp(\frac{2\pi i}{n^2})$ and $\alpha^k = \exp(\frac{2\pi i k}{n^2})$.

Figure 2.1: Pseudocode to compute the m most significant digits for probabilities $p_{rn+s} = \frac{\mathbf{Z}^{r,s}_{1,n}}{\mathbf{Z}}$. In our implementation, due to numerical stability issues in the FFT engine, precision parameter m has an upper bound of 8 – only the $m = 8$ most significant digits are computed with `FFTbor2D`. (Note that the software actually usesbase 2 precision parameter, with maximum of 27, where $2^{27} \approx 10^8$.) It is well-known that the FFT requires $O(N \log N)$ time to solve the inverse discrete Fourier transform for a polynomial of degree N. In our case, $N = n^2$, and so the FFT requires time $O(n^2 \log n)$.

2.2.4 Speed-up by factor of 4

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. $\{e^{2\pi i k/n} : k = 0, \ldots, n-1\}$ is a set of pairwise distinct nth roots of unity. For notational reasons below, we will write 'n-root of unity' instead of 'nth root of unity'. We have the following.

Lemma 1: 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 $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B},)$. If $x = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{S},)$ and $y = d_{\mathrm{BP}}(\mathcal{S}, \mathcal{B},)$, then $y \in \{d_0 - x + 2k : k = 0, \ldots, x\}$.

It follows that if $x = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{S},)$ and $y = d_{\mathrm{BP}}(\mathcal{S}, \mathcal{B},)$, 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

$$d_{\mathrm{BP}}(\mathcal{A}, \mathcal{S}, +) d_{\mathrm{BP}}(\mathcal{S}, \mathcal{B}, \equiv) d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, \bmod) 2 \tag{2.14}$$

first noticed in **?**, as well as the triangle inequality $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{S}, +)d_{\mathrm{BP}}(\mathcal{S}, \mathcal{B}, \geq)d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B},)$ for base pair distance, probably folklore. Lorenz et al. **?** exploited the parity condition and the triangle inequality by using sparse matrix methods to improve on the efficiency of the naive implementation of the $O(n^7)$ time and $O(n^4)$ space algorithm to compute the partition function, $\mathbf{Z}^{r,s}_{1,n}$, 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 $\mathcal{B}$. The following lemma is not difficult to establish.

Lemma 2: If $\mathcal{Z}(x)$ is the complex polynomial defined in equation (**??**), then for any complex nth root of unity α, it is the case that $\mathcal{Z}(\overline{\alpha}) = \overline{\mathcal{Z}(\alpha)}$.

Lemma 3: Let $\mathcal{Z}(x)$ be defined by equation (**??**), 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 $emZof{-\alpha} = emZof\alpha$, while if the distance is odd, then $emZof{-\alpha} = -emZof\alpha$.

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

$$\nu^k = -(\nu^{-(M/2-k)}) = -\overline{\nu^{M/2-k}}. \tag{2.15}$$

Lemma 1 is proved by simple induction; Lemma 2 is proved by a computation involving binomial coefficients; Lemma 3 is immediate by the parity observation above, resulting from Lemma 1; Lemma 4 is elementary, relying on Euler's formula and trigonometric addition formulas. Details proofs of Lemmas 2,3,4 can be found in supplementary information.

Lemma 1 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 2 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 1 and Lemma 2, along with a precomputation of powers of the complex roots of unity, lead to a large performance speed-up in our implementation of `FFTbor2D`– indeed, by a factor of 4 or more. Though the intuitive idea of how to obtain this speedup by a factor of four may be apparent, the technical details leading to the pseudocode of `FFTbor2D`, presented in Figure 2, are rather tricky. These details are presented in the next two subsections, which can be skipped by the reader wishing to move on to the algorithm itself.

2.2.5 Time reduction due to Lemma 1

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

$$\begin{aligned} \mathcal{Z}(x) &= \sum_{r=0}^{N} \sum_{s=0}^{N} z_{rN+s} x^{r \cdot N+s} && (2.16) \\ &= \sum_{k=0}^{(N+1)^2-1} z_{\pi_1(k)\cdot(N+1)+\pi_2(k)} x^{\pi_1(k)\cdot(N+1)+\pi_2(k)} && (2.17) \\ &= \sum_{k=0}^{(N+1)^2-1} z_k x^k && (2.18) \end{aligned}$$

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 2-dimensional array.

Consider the coefficients of the polynomial

$$\mathcal{Z}(x) = \sum_{r=0}^{N} \sum_{s=0}^{N} z_{rN+s} x^{rN+s} = \sum_{k=0}^{(N+1)^2-1} z_k x^k. \tag{2.19}$$

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

Case 1: All coefficients z_k of odd parity in equation (2.19) are zero.

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

$$Y(u) = \sum_{k=0}^{M_0-1} b_k u^k = \sum_{k=0}^{M_0-1} z_{2k} x^{2k}. \tag{2.20}$$

Since $Y(x^2) = Z(x)$, we have $Y(\exp(\frac{2\pi ki}{M/2})) = Y(\exp(\frac{4\pi ki}{M})) = Z(\exp(\frac{2\pi ki}{M}))$, hence we use the previous recursions (2.10) to evaluate $Z(\exp(\frac{2\pi ki}{M})$. Instead of performing M evaluations of $Z(x)$ at M-roots of unity, this requires only $M_0 = M/2$ evaluations of $Y(u)$ at M_0-roots of unity; i.e. only half the number of evaluations of $Z(x)$ are necessary to obtain the coefficients of $Y(x)$. But then, we immediately obtain the full polynomial $\mathcal{Z}(x)$, since its coefficients of odd parity are zero.

Case 2: All coefficients z_k of even parity in equation (2.19) are zero.

In this case, $z_0, z_2, z_4, \ldots$ are zero, so $\mathcal{Z}(x) = \sum_{k=0}^{M/2-1} z_{2k+1} x^{2k+1}$. But then $\mathcal{Z}(x) = x \cdot Y(u)$, where $Y(u) = \sum_{k=0}^{M_0-1} b_k u^k$, where we have made a variable change $u = x^2$, and coefficient changes $b_k = z_{2k+1}$. Similarly to Case 1, we can interpolate the M_0 coefficients of the polynomial $Y(u) = \sum_{k=0}^{M_0-1} b_k u^k$ by evaluating M_0 many complex M_0-roots of unity. Since $\mathcal{Z}(x) = x \cdot Y(x^2)$, $Y(x^2) = x^{-1} \cdot \mathcal{Z}(x)$, so $Y(\exp(\frac{2\pi ki}{M/2})) = Y(\exp(\frac{4\pi ki}{M})) = \exp(\frac{-2\pi ki}{M}) \cdot \mathcal{Z}(\exp(\frac{2\pi ki}{M}))$, employing the previous recursions (2.10) to evaluate $\mathcal{Z}(\exp(\frac{2\pi ki}{M}))$. Note, that unlike the Case 1, since $Z(x) = x \cdot Y(x^2)$, we have $Y(x^2) = \frac{\mathcal{Z}(x)}{x}$, which explains the presence of additional factor $\exp(\frac{-2\pi ki}{M}))$ in Case 2. Thus, instead of performing M evaluations of $\mathcal{Z}(x)$ at M-roots of unity, we perform only $M_0 = \frac{M}{2}$ evaluations of $Y(u)$ at M_0-roots of unity; i.e. only half the number of evaluations of $\mathcal{Z}(x)$ are necessary to obtain the coefficients of $Y(x)$. But then, we immediately obtain the full polynomial $\mathcal{Z}(x)$, since $\mathcal{Z}(x) = x \cdot Y(x^2)$, and the coefficients of $\mathcal{Z}(x)$ of even parity are zero.

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

$$Y(x^2) = \mathcal{Z}(x) \tag{2.21}$$

while if the parity is odd, then

$$Y(x^2) = \frac{1}{x} \cdot \mathcal{Z}(x). \tag{2.22}$$

2.2.6 Time reduction due to Lemma 2

As before, let M be the the least number evenly divisible by 4, which is greater than or equal to $(N+1)^2$, let $\nu = \exp(\frac{2\pi i}{M})$ and $\omega = \nu^2 = \exp(\frac{2\pi i}{M})^2 = \exp(\frac{2\pi i}{M/2})$. Clearly, ν is a principal complex M-root of unity, while ω is a principal complex $\frac{M}{2}$-root of unity. Evaluate $Z(\alpha)$ for each $\frac{M}{2}$-root of unity that belongs to the first quadrant, and apply Lemma 2 to infer the values of $Z(\alpha)$ for each $\frac{M}{2}$-root of unity that belongs to the fourth quadrant. More precisely, we compute $Z(\nu^k)$, for $k = 0, \ldots, \frac{M}{4}$, and by Lemmas 2,3,4 infer that for $k = \frac{M}{4}+1, \ldots, \frac{M}{2}-1$, we have $Z(\nu^k) = -1^{d_0} \cdot \overline{Z(\nu^{\frac{M}{2}-k})}$, where $d_0 = d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B},)$. This is justified in the following.

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

$$\begin{aligned}
Y(\omega^k) &= Y(\nu^{2k}) \\
&= \begin{cases} Z(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)0 \bmod 2 \\ \frac{1}{\nu^k} \cdot Z(\nu^k) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)1 \bmod 2 \end{cases} \\
&= \begin{cases} Z(-\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)0 \bmod 2 \\ \nu^{-k} \cdot Z(-\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)1 \bmod 2 \end{cases} \\
&= \begin{cases} Z(\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)0 \bmod 2 \\ \nu^{-k} \cdot -Z(\overline{\nu^{(\frac{M}{2}-k)}}) & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)1 \bmod 2 \end{cases} \\
&= \begin{cases} \overline{Z(\nu^{(\frac{M}{2}-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)0 \bmod 2 \\ -\nu^{-k} \cdot \overline{Z(\nu^{(\frac{M}{2}-k)})} & \text{if } d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B}, =)1 \bmod 2 \end{cases}
\end{aligned} \tag{2.23}$$

Line 1 follows by definition, since $\omega = \nu^2$; line 2 follows by equations (2.21) and (2.22); line 3 follows by Lemma 4; line 4 follows by Lemma 3. Thus if $d_{\mathrm{BP}}(\mathcal{A}, \mathcal{B},)$ is even, then

$$y_k = Y(\omega^k) = \begin{cases} Z(\nu^k) & \text{for } k = 0, \ldots, \frac{M}{4} \\ \\ \overline{Z(\nu^{\frac{M}{2}-k})} & \text{for } k = \frac{M}{4}+1, \ldots, \frac{M}{2}-1. \end{cases} \tag{2.24}$$

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

$$y_k = Y(\omega^k) = \begin{cases} \nu^{-k} \cdot Z(\nu^k) & \text{for } k = 0, \ldots, \frac{M}{4} \\ \\ -\nu^{-k} \cdot \overline{Z(\nu^{\frac{M}{2}-k})} & \text{for } k = \frac{M}{4}+1, \ldots, \frac{M}{2}-1. \end{cases} \tag{2.25}$$

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

Improved Algorithm for `FFTbor2D`
Input: RNA sequence $\mathbf{s} = s_1, \ldots, s_n$, and distinct secondary structures $\mathcal{A}, \mathcal{B}$ of $\mathbf{s}$, and integer m.
Output: Probabilities $p(x,y) = \mathbf{Z}^{x,y}_{1,n}/\mathbf{Z}$ to m significant digits for $x, y = 0, \ldots, n-1$. Let N be the least even number greater than or equal to n, M be the least number evenly divisible by 4, which is greater than or equal to $(N+1)^2$, $M_0 = M/2$, $\nu = \exp(\frac{2\pi i}{M})$, $\omega = \nu^2 = \exp(\frac{2\pi i}{M_0})$. For $0 \leq k < M^2$, let $\pi_1(k) = \lfloor \frac{k}{M} \rfloor$, $\pi_2(k) = k - M \cdot \pi_1(k) = k \bmod M$, and note that $k = \pi_1(k) \cdot M + \pi_2(k)$.

Figure 2.2: Pseudocode to compute the m most significant digits for probabilities $p_k = \frac{z_k}{\mathbf{Z}} = \frac{\mathbf{Z}^{\pi_1(k),\pi_2(k)}_{1,n}}{\mathbf{Z}}$. Our program, `FFTbor2D`, supports values of $m = 1, \ldots, 8$ for the precision parameter m. (Note that the software actually usesbase 2 precision parameter, with maximum of 27, where $2^{27} \approx 10^8$.)

2.2.7 Using the fast Fourier transform

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

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

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

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

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

$$a_j = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \omega^{-kj} = \frac{1}{M_0} \sum_{k=0}^{M_0-1} y_k \nu^{-2kj} \tag{2.26}$$

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

Since we defined $\mathbf{Y}$ in (2.12) by $\mathbf{Y} = (y_0, \ldots, y_{M_0-1})^T$, where $y_0 = \mathcal{Z}(\alpha_0), \ldots, y_{M_0-1} = \mathcal{Z}(\alpha_{M_0-1})$ and $\alpha_k = \omega^k \exp(\frac{k \cdot 2\pi i}{M_0})$, it follows that the coefficients $z_k = \mathbf{Z}_{1,n}^{\pi_1(k),\pi_2(k)}$ in the polynomial $\mathcal{Z}(x) = z_0 + z_1 x + \cdots + z_M x^M$ defined in (**??**) can be computed, at least in principle, by using the FFT. However, since the values of z_k are astronomically large, numerical instability makes even this approach infeasible for moderate values of n. Nevertheless, we apply this approach to compute the m most significant digits of $\frac{\mathbf{Z}_{1,n}^{\pi_1(k),\pi_2(k)}}{\mathbf{Z}}$, where the partition function $\mathbf{Z} = \sum_S \exp(-E(\mathcal{S})/RT)$ satisfies $\mathbf{Z} = \sum_{x,y} \mathbf{Z}_{1,n}^{x,y}$. This leads to numerical stability, allowing `FFTbor2D` to compute the m most significant digits of $p(x,y) = \frac{\mathbf{Z}_{1,n}^{x,y}}{\mathbf{Z}}$. Pseudocode for the complete algorithm, `FFTbor`, is given in Figure 2.

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

2.3 Acceleration of the `FFTbor2D` algorithm

2.3.1 Optimization due to parity condition

2.3.2 Optimization due to complex conjugates

2.4 Benchmarking and performance considerations

2.5 Applications of the `FFTbor2D` algorithm