# Weekly Report

Zhuoran Qiao

August 10, 2018

## 1 Introduction

**1** Developed a genetic algorithm based approach to simulate kinetics of co-transcriptional folding.

**2** Made initial tests on effect of folding rate on $p_{unbound}$ during transcription.

## 2 Progress

### 2.1 Framework

To quantitatively predict folding dynamics coupled with transcription, we developed a genetic algorithm based approach. Our method is built on two following assumptions:

**1** All populated RNA secondary structures (SS) are linkage of locally optimal or sub-optimal structures at various folding sites;

**2** Global structural rearrangement of a partial RNA segment is permitted only if it's folding to the optimal SS on that segment.

Formally, we denote a domain $D_{A,\,B}$ as a segment between base $A$ and $B$ that all contacts on that segment are local. For simplicity, we denote **foldon** as domains with optimal secondary structures: $D^{foldon}_{A,\,B}$ = MFE(sequence[A,B]). Note that '.' is a trival example of foldon. Our assumption 1 can be rewritten as

$$D_{A,\,B} = D^{foldon}_{A,i_1} \oplus D^{foldon}_{i_1,i_2} \oplus ... \oplus D^{foldon}_{i_n,B} \tag{1}$$

Where $\oplus$ represents a link operation. Note that all structural information of $D_{A,\,B}$ is encoded by the sequential representation $[A,\, i_1,\, ...,\, i_n,\, B]$; as a foldon is also a linkage of smaller foldons, there could be multiple way to represent $D_{A,\,B}$. Here we introduce **Irreducible Foldon Representation** (IFR) to be the sequential representations for which linkage of every adjacent foldons is not another foldon: $\forall k$, $D^{foldon}_{i_k, i_{k+1}} \oplus D^{foldon}_{i_{k+1}, i_{k+2}} \neq D^{foldon}_{i_k, i_{k+2}}$. Then the sufficient and necessary condition for structural rearrangement is

$$\begin{gathered}
\langle D^u_{A,\,B} | \hat{\mathbf{T}} | D^v_{A,\,B} \rangle \neq 0 \text{ if and only if } \exists\, i,\, j \text{ satisfies} \\
i,\, j \in D^u_{A,\,B}.\text{IFR},\ i,\, j \in D^v_{A,\,B}.\text{IFR}; \\
D^u_{A,\,i} = D^v_{A,\,i},\ D^u_{j,\,B} = D^v_{j,\,B}; \\
D^u_{i,\,j} = D^{foldon}_{i,\,j} \text{ or } D^v_{i,\,j} = D^{foldon}_{i,\,j}. \\
\text{Then } \langle D^u_{A,\,B} | \hat{\mathbf{T}} | D^v_{A,\,B} \rangle = \langle D^u_{i,\,j} | \hat{\mathbf{T}} | D^v_{i,\,j} \rangle.
\end{gathered}$$

## 2.2 Algorithm procedure

During every iterative elongation step, an active species pool of strands with unique SS and diffrent population is updated. New candidate strands $D^{Candidate}_{0,\,L+\Delta L}$ with length $L + \Delta L$ are generated by a recombination process: for every old strand $D^{Strand}_{0,\,\text{L}}$, all indices in its IFR is identified as possible rearrangement site, then its child strands is generated by linking partial domains $D^{Strand}_{0,\,\text{Site}}$ with a foldon $D^{foldon}_{\text{Site}, L+\Delta L}$ that terminated at $L + \Delta L$.

We assume that elongation will not change the inital population distribution of secondary structures: child strands with the exact parental SS on $[0, L]$ ($D^{child}_{0,L+\Delta L} = D^{strand}_{0,L} \oplus D^{foldon}_{L,L+\Delta L}$) will also inherit the population of their parents.

After structual generation the rate matrix among all candidate strands within the new active species pool is calculated (see part 2.4). Then the population distribution of strands after elongation is computed by propagate the chemical master equation.

For the sake of computational efficiency, we introduce a cutoff $N$ as the size limit of the active species pool. After each elongation step, we impose a selection sweep on all active strands; species with top $N$ fitness is reserved. In the current edition, we simply used population as the fitness function. Population of remaining strands within the active pool is renormalized after selection.

Pseudocodes of the procedure are as follows (Algorithm 1):

## 2.3 Secondary structure generation

## 2.4 Folding pathway identification & Rate calculation

**Algorithm 1** Co-transcriptional folding elongation procedure

```
1: Initalize ActivePool
2: while sequence length > current length do
3:     OldPool ← ActivePool
4:     renew ActivePool
5:     Current length ← Current length + $dL$
6:     dt ← dL / Transcription rate
7:     for left boundary ∈ {0, dL, 2dL, ..., Current length - dL} do        ▷ Get all new foldons
8:         $D^{foldon}_{\text{left boundary, Current length}}$ ← numpy.mfe(sequence[left boundary, Current length])
9:     end for
10:    for Strand ∈ OldPool do                                              ▷ Recombination
11:        for Site ∈ Strand.IFR do
12:            $D^{Candidate}_{\text{0,Current length}} \leftarrow D^{Strand}_{\text{0, Site}} \oplus D^{foldon}_{\text{Site, Current length}}$
13:            if $D^{Candidate}_{\text{0,Current length}}$ ∈ ActivePool then
14:                update $D^{Candidate}_{\text{0,Current length}}$.IFR
15:            else
16:                add $D^{Candidate}_{\text{0,Current length}}$ to ActivePool
17:            end if
18:            if site = Current length $- dL$ then
19:                $\langle \text{ActivePool.}\mathbf{population}\,|D^{Candidate}_{\text{0,Current length}}\rangle \leftarrow \langle \text{OldPool.}\mathbf{population}\,|D^{Strand}_{\text{0, Site}}\rangle$
20:            end if
21:        end for
22:    end for
23:    for $D^{\text{u}}_{\text{0,Current length}} \neq D^{\text{v}}_{\text{0,Current length}}$ ∈ ActivePool do     ▷ Calculate new rate matrix
24:        calculate $D^{u}_{\text{rearrange}}$, $D^{v}_{\text{rearrange}}$          ▷ Find all helices involved in rearrangement
25:        $\langle D^{u}_{\text{rearrange}}|\hat{\mathbf{T}}|D^{v}_{\text{rearrange}}\rangle \leftarrow k_0 \exp\left(-\frac{1}{RT}(\Delta G^{Stack}_{u} + \Delta G^{Loop}_{v})\right)$
26:    end for
27:    $\langle \text{ActivePool.}\mathbf{population}\,| \leftarrow \langle \text{ActivePool.}\mathbf{population}\,|\exp\left(\hat{\mathbf{T}}\right)$        ▷ Master equation
28:    reserve top $N$ populated strands in ActivePool                  ▷ Selection
29:    renormalize $\langle \text{ActivePool.}\mathbf{population}\,|$
30: end while
```