RNA STRUCTURE
RNA Basics

- RNA bases A, C, G, U
- Canonical Base Pairs
  - A-U
  - G-C
  - G-U
    - “wobble” pairing
  - Bases can only pair with one other base.

3 Hydrogen Bonds – more stable
RNA Basics

- transfer RNA (tRNA)
- messenger RNA (mRNA)
- ribosomal RNA (rRNA)
- small interfering RNA (siRNA)
- micro RNA (miRNA)
- small nucleolar RNA (snoRNA)
- ..... others

http://www.genetics.wustl.edu/eddy/tRNAscan-SE/
RNA folding

- Prediction of secondary structure of an RNA given its sequence
- General problem is NP-hard due to “difficult” substructures, like pseudoknots
- Most existing algorithms require too much memory ($\geq O(n^2)$), and run time ($\geq O(n^3)$) thus limited to smaller RNA sequences
RNA Structural Levels

**Primary**

AAUCG...CUUCUUCCA

**Secondary**

[Diagram of RNA secondary structure]

**Tertiary**

[3D model of RNA tertiary structure]
RNA families

- **Rfam**: General non-coding RNA database (most of the data is taken from specific databases)

  [http://www.sanger.ac.uk/Software/Rfam/](http://www.sanger.ac.uk/Software/Rfam/)

  Includes many families of non-coding RNAs and functional Motifs, as well as their alignment and their secondary structures
RNA Secondary Structure

- Hairpin loop
- Junction (Multiloop)
- Pseudoknot
- Interior Loop
- Bulge Loop
- Single-Stranded
- Stem
Example: 5S rRNA

**E. coli 5S**
120 bases

**T. thermophilus 5S**
120 bases
Example: E. coli 16S rRNA

1542 bases
Example: E. coli 23S rRNA

2904 bases
Example: HIV

9173 bases

Watts et al., Nature, 2009
Binary Tree Representation of RNA Secondary Structure

- Representation of RNA structure using Binary tree
- Nodes represent
  - Base pair if two bases are shown
  - Loop if base and “gap” (dash) are shown
- Pseudoknots still not represented
- Tree does not permit varying sequences
  - Mismatches
  - Insertions & Deletions

Images – Eddy et al.
Circular Representation

Images – David Mount
Examples of known interactions of RNA secondary structural elements

Pseudoknot

These patterns are excluded from the prediction schemes as their computation is too intensive.

Kissing hairpins

Hairpin-bulge contact
Predicting RNA secondary structure

- Base pair maximization
- Minimum free energy (most common)
  - Fold, Mfold (Zuker & Stiegler)
  - RNAfold (Hofacker)
- Multiple sequence alignment
  - Use known structure of RNA with similar sequence
- Covariance
- Stochastic Context-Free Grammars
Sequence Alignment as a method to determine structure

- Bases pair in order to form backbones and determine the secondary structure.
- Aligning bases based on their ability to pair with each other gives an algorithmic approach to determining the optimal structure.
Simplifying Assumptions

- RNA folds into one minimum free-energy structure.
- There are no knots (base pairs never cross).
- The energy of a particular base pair in a double stranded regions is sequence independent
  - Neighbors do not influence the energy.
- Was solved by dynamic programming, Zuker and Stiegler 1981
Base Pair Maximization

Base Pair Maximization – Dynamic Programming Algorithm

\[ S(i,j) = \max \begin{cases} 
S(i + 1, j - 1) + 1 & \text{[if } i,j \text{ base pair]} \\
S(i + 1, j) \\
S(i, j - 1) \\
\max_{i < k < j} S(i,k) + S(k + 1, j)
\end{cases} \]

\[ S(i,j) \text{ is the folding of the subsequence of the RNA strand from index } i \text{ to index } j \text{ which results in the highest number of base pairs} \]

http://bioalgorithms.info
Base Pair Maximization – Dynamic Programming Algorithm

- Alignment Method
  - Align RNA strand to itself
  - Score increases for feasible base pairs
- Each score independent of overall structure
- Bifurcation adds extra dimension

Dynamic Programming – possible paths

\[ S(i + 1, j) = \begin{cases} 
0 & \text{if } i = 0 \\
S(i + 1, j - 1) + 1 & \text{if bases can pair, similar to matched alignment} \\
0 & \text{if bases cannot pair, similar to unmatched alignment}
\end{cases} \]
Base Pair Maximization – Dynamic Programming Algorithm

- **Alignment Method**
  - Align RNA strand to itself
  - Score increases for feasible base pairs
- **Each score independent of overall structure**
- **Bifurcation adds extra dimension**

Reminder:
For all \( k \)

\[
S(i,k) + S(k + 1, j)
\]

Bifurcation – add values for all \( k \)

**Images – Sean Eddy**
Base Pair Maximization - Drawbacks

- Base pair maximization will not necessarily lead to the most stable structure
  - May create structure with many interior loops or hairpins which are energetically unfavorable

- Comparable to aligning sequences with scattered matches – not biologically reasonable
Energy Minimization

- Thermodynamic Stability
  - Estimated using experimental techniques
  - Theory: Most Stable is the Most likely
- No Pseudoknots due to algorithm limitations
- Uses Dynamic Programming alignment technique
- Attempts to maximize the score taking into account thermodynamics
- MFOLD and ViennaRNA
Free energy of a structure is the sum of all interactions energies

Free Energy\(E\) = \(E(CG) + E(CG) + \ldots\)

Each interaction energy can be calculated thermodynamically
Why is MFE secondary structure prediction hard?

- MFE structure can be found by calculating free energy of all possible structures
- BUT the number of potential structures grows exponentially with the number, n, of bases
RNA folding with Dynamic programming (Zuker and Stiegler)

- $W(i,j)$: MFE structure of substrand from $i$ to $j$
Assume a function $W(i,j)$ which is the MFE for the sequence starting at $i$ and ending at $j$ ($i<j$)

Define scores, for example base pair (CG) = -1 non-pair (CA) = 1 (we want a negative score)

Consider 4 possibilities:
- $i,j$ are a base pair, added to the structure for $i+1..j-1$
- $i$ is unpaired, added to the structure for $i+1..j$
- $j$ is unpaired, added to the structure for $i..j-1$
- $i,j$ are paired, but not to each other;

Choose the minimal energy
Energy Minimization Results

- All loops must have at least 3 bases in them
  Equivalent to having 3 base pairs between all arcs

Exception: Location where the beginning and end of RNA come together in circularized representation

Images – David Mount
Trouble with Pseudoknots

- Pseudoknots cause a breakdown in the Dynamic Programming Algorithm.
- In order to form a pseudoknot, checks must be made to ensure base is not already paired – this breaks down the recurrence relations.

Images – David Mount
Sequence dependent free-energy
Nearest Neighbor Model

Energy is influenced by the previous base pair
(not by the base pairs further down).
Sequence dependent free-energy values of the base pairs

These energies are estimated experimentally from small synthetic RNAs.

Example values:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GC</td>
<td>GC</td>
<td>GC</td>
<td>GC</td>
<td>GC</td>
<td></td>
</tr>
<tr>
<td>AU</td>
<td>GC</td>
<td>CG</td>
<td>UA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.3</td>
<td>-2.9</td>
<td>-3.4</td>
<td>-2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adding Complexity to Energy Calculations

- **Stacking energy** - Assign negative energies to these *between base pair* regions.
  - Energy is influenced by the previous base pair (not by the base pairs further down).
  - These energies are estimated experimentally from small synthetic RNAs.

- **Positive energy** - added for destabilizing regions such as bulges, loops, etc.

- More than one structure can be predicted
Mfold

- Positive energy - added for destabilizing regions such as bulges, loops, etc.
- More than one structure can be predicted
Free energy computation

\[ \Delta G = -4.6 \text{ KCAL/MOL} \]
Mfold

- Positive energy - added for destabilizing regions such as bulges, loops, etc.
- More than one structure can be predicted
More than one structure can be predicted for the same RNA

RNA fold prediction based on Multiple Alignment

Information from multiple sequence alignment (MSA) can help to predict the probability of positions \( i,j \) to be base-paired.
Compensatory Substitutions

Mutations that maintain the secondary structure can help predict the fold.
RNA secondary structure can be revealed by identification of compensatory mutations.
Insight from Multiple Alignment

Information from multiple sequence alignment (MSA) can help to predict the probability of positions i,j to be base-paired.

- Conservation – no additional information
- Consistent mutations (GC → GU) – support stem
- Inconsistent mutations – do not support stem.
- Compensatory mutations – support stem.
**RNAalifold**

- Predicts the consensus secondary structure for a set of aligned RNA sequences by using modified dynamic programming algorithm that add alignment information to the standard energy model
- Improvement in prediction accuracy
STOCHASTIC CONTEXT-FREE GRAMMARS
RNA folding can be represented as context-free grammars
Chomsky hierarchy

- Unrestricted grammars
  - Context-sensitive grammars
    - Context-free grammars
      - Regular grammars

  (equivalent to Turing machines & recursively enumerable sets)

  (equivalent to finite automata & HMM’s)

  (equivalent to linear bounded TMs)

B. Majoros
A **context-free grammar** is a generative model denoted by a 4-tuple:

\[ G = (V, \sum, S, R) \]

where:

- \( \sum \) is a **terminal alphabet**, (e.g., \( \{a, c, g, u\} \) )
- \( V \) is a **nonterminal alphabet**, (e.g., \( \{A, B, C, D, E, ...\} \) )
- \( S \in V \) is a special **start symbol**, and
- \( R \) is a set of rewriting rules called **productions**.

Productions in \( R \) are rules of the form:

\[ X \rightarrow \lambda \]

where \( X \in V, \lambda \in (V \cup \sum)^* \)
Context “freeness”

The “context-freeness” is imposed by the requirement that the l.h.s of each production rule may contain only a **single** symbol, and that symbol must be a **nonterminal**:

\[ X \rightarrow \lambda \]

Thus, a CFG **cannot** specify **context-sensitive** rules such as:

\[ wXz \rightarrow w\lambda z \]
Suppose a CFG $G$ has generated a terminal string $w \in \sum^*$. A derivation $S \Rightarrow ^* w$ denotes a possible derivation for generating $w$.

A derivation (or parse) consists of a series of applications of productions from $R$, beginning with the start symbol $S$ and ending with the terminal string $w$:

$$S \Rightarrow s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow \cdots \Rightarrow w$$

where $s_i \in (V \cup \sum)^*$.

We’ll concentrate of leftmost derivations where the leftmost nonterminal is always replaced first.
A CFG for an RNA

RNA hairpin with 3 bp stem and a 4-base loop (GAAA or GCAA)

S -> aXu | cXg | gXc | uXa
X -> aYu | cYg | gYc | uYa
Y -> aZu | cZg | gZc | uZa
Z -> gaaa | gcaa

R. Shamir & R. Sharan
Parse trees

- A representation of a parse of a string by a CFG
- **Root** – start nonterminal S
- **Leaves** – terminal symbols in the given string
- **Internal nodes** - nonterminals
- The children of an internal node are the productions of that nonterminal (left-to-right order

---

R. Shamir & R. Sharan
A stochastic context-free grammar (SCFG) is a CFG plus a probability distribution on productions:

\[ G = (V, \sum, S, R, P_p) \]

where \( P_p : R \setminus \) \( \lambda \), and probabilities are normalized at the level of each l.h.s. symbol \( X \):

\[ \forall [ \sum P_p(X \rightarrow \lambda) = 1 ] \]

\[ X \in V \; X \rightarrow \lambda \]

Thus, we can compute the probability of a single derivation \( S \Rightarrow^* w \) by multiplying the probabilities for all productions used in the derivation:

\[ \prod_i P(X_i \rightarrow \lambda_i) \]

We can sum over all possible (leftmost) derivations of a given string \( w \) to get the probability that \( G \) will generate \( x \) at random:

\[ P(x \mid G) = \sum_j P(S \Rightarrow_j^* x \mid G). \]
An example

As an example, consider \( G = (V_G, \Sigma, S, R_G, P_G) \), for \( V_G = \{S, L, N\} \), \( \Sigma = \{a, c, g, t\} \), and \( R_G \) the set consisting of:

\[
S \rightarrow a \ S \ u \mid u \ S \ a \mid c \ S \ g \mid g \ S \ c \mid L \quad (P=0.2)
\]

\[
L \rightarrow N \ N \ N \ N \ N \ \quad (P=1.0)
\]

\[
N \rightarrow a \mid c \mid g \mid u \quad (P=0.25)
\]

Then the probability of the sequence \textcolor{red}{acguacguacgu} is given by:

\[
P(\text{acguacguacgu}) =
\]

\[
P( S \Rightarrow aSu \Rightarrow acSgu \Rightarrow acgSgcu \Rightarrow acguScgcu \Rightarrow acguLaacgu \Rightarrow acguNNNNacgu \Rightarrow acguaNNNNacgu \Rightarrow acguacNNNacgu \Rightarrow acguacgNNacgu \Rightarrow acguacaguacgu) =
\]

\[
0.2 \times 0.2 \times 0.2 \times 0.2 \times 1 \times 0.25 \times 0.25 \times 0.25 \times 0.25 = 1.25 \times 10^{-6}
\]

because this sequence has only one possible (leftmost) derivation under grammar \( G \).
Structure using SFCG

- Grammar rules with associated probabilities

\[
S \rightarrow aSu \mid cSg \mid aS \mid uS \mid \ldots \mid Su \mid SS \mid \varepsilon
\]

\[
P = 0.21 \quad 0.15 \quad 0.11 \quad 0.08 \quad 0.03 \quad 0.22 \quad 0.02
\]

- We select the set of transformations that highest probability of generating the input sequence. This set gives us our structure.

- Let’s generate a structure for the sequence acuguaucuag

```plaintext
acuyggggg

- (((())))}
```
Chomsky Normal Form

A CNF grammar is one in which all productions are of the form:

\[ X \rightarrow YZ \]

or:

\[ X \rightarrow a \]

Non-CNF:

\[
\begin{align*}
S & \rightarrow as t | tSa | cs g | gSc | L \\
L & \rightarrow N N N N \\
N & \rightarrow a | c | g | u
\end{align*}
\]

CNF:

\[
\begin{align*}
S & \rightarrow AS_T | TS_A | CS_G | GS_C | NL_1 \\
S_A & \rightarrow SA \\
S_T & \rightarrow ST \\
S_C & \rightarrow SC \\
S_G & \rightarrow SG \\
L_1 & \rightarrow NL_2 \\
L_2 & \rightarrow NN \\
N & \rightarrow a | c | g | u \\
A & \rightarrow a \\
C & \rightarrow c \\
G & \rightarrow g \\
T & \rightarrow u
\end{align*}
\]
Two questions for a CFG:

1) Can a grammar $G$ derive string $w$?
2) If so, what series of productions would be used during the derivation? (there may be multiple answers)

Additional questions for an SCFG:

1) What is the probability that $G$ derives string $w$?
2) What is the most probable derivation of $w$ via $G$?
Parsing CFG

- CYK Algorithm (Cocke-Younger-Kasami)
  - Dynamic Programming method

- Modified CYK for SCFG
  - “Inside algorithm”
  - Training similar to HMM
    - If parses are known for training data sequences, simply count the number of times for each production, calculate probabilities (labeled sequence training for HMM)
    - If parses are not known, apply an EM algorithm called “Inside-Outside” (“forward-backward” for HMM)