CMSC423: Bioinformatic Algorithms, Databases and Tools

RNA folding
RNA folding

- Function of RNA molecules depends on how they fold, based on nucleotide base-pairing.
Types of structures

- Nested (hairpin)

```
ACAUGGAUGU
((((..))))
```

- Pseudo-knots

```
UUCCGAAGCUCAACGGGAAAUGAGCU
UUCCGAAGCUCAAUGGGCAACUCGA
-A-A--UGAGCU
UUCCGAAGCUCAAACGGGAAUUGAGCU
```
RNA folding... not just for RNA

• RNA – given the sequence figure out how it folds
• Parsing
  – Given a grammar
    \[ A \rightarrow BC \]
  – Parse a string of letters (text, program) according to grammar
    \[ A \rightarrow \alpha \]
    \[ S \rightarrow \lambda \]
• ... example
  – Given a C program
  – Find whether parentheses, braces, and brackets are matched up correctly
  – if not, where are the likely errors (find the smallest number of corrections that would make the program balanced)
• Both can be solved by the same algorithm
From multiple alignment to structure

- Find columns in the alignment where mutations are correlated
- Mutual information - how correlated are the columns?

\[ M_{i,j} = \sum_{x_i,x_j} f_{x_i,x_j} \log \left( \frac{f_{x_i,x_j}}{f_{x_i}f_{x_j}} \right) \]

\( M_{i,j} \) = mutual information between columns \( i \) and \( j \)
\( f_{x_i,x_j} \) = frequency of each of 16 pairs of nucleotides at columns \( i \) and \( j \)
\( f_{x_i} \) = frequency of each of 4 nucleotides at column \( i \)
\( f_{x_j} \) = frequency of each of 4 nucleotides at column \( j \)
Mutual information

- Ranges from 0 to 2 for a 4-letter alphabet
- Correlated columns - mutual information high
- Advantages:
  - Don't need to know how RNA folds - pseudo-knots should “pop” out of the alignment
- Disadvantages:
  - Need many sequences in an alignment (to compute frequencies)
  - The aligned sequences must be sufficiently divergent (conserved columns provide no information)
Nussinov's algorithm

• Assumes no pseudo-knots
• Dynamic programming approach – maximize # of pairings

• S – string of nucleotides representing the RNA molecule
• Sub-problem – F[i,j] – score of folding just S[i..j]
• Initial values: F[i-1,i] = F[i,i] = F[i, i+1] = 0
Nussinov's algorithm

$F[i,j]$ is the maximum of:

I. $F[i+1,j]$  
   $S[i]$ unpaired

II. $F[i,j-1]$  
    $S[j]$ unpaired

III. $F[i+1,j-1] + 1$  
     if $S[i+1]$ complementary to $S[j-1]$  
     $S[i]$ paired with $S[j]$

IV. $\max_k F[i,k] + F[k+1,j]$  
    Branch
Questions

• In what order do we fill the dynamic programming table?

• How can we ensure that "loops" consist of at least k nucleotides?

• Note: related to CYK parsing algorithm for Chomsky Normal Form grammars
\[
F[i+1, j] \\
F[i, j - 1] \\
F[i+1, j-1] + 1 \text{ (if paired)} \\
\max_{k} F[i,k] + F[k+1,j]
\]
<table>
<thead>
<tr>
<th>G</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>U</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

GGGAAAUCC
((.(..())))
.((..())))
A better objective function

• Find the RNA fold that minimizes the Gibbs free energy

• Zucker's algorithm keeps track of:
  – Stacking energy - $f$(# of base-pairs in a stem)
  – Loop energy - $f$(length of loop)
  – Bulge energy - $f$(length of bulge)
  – Dangle energy - $f$(length of dangle)

• Computation is done with an extension of the traditional (Nussinov) dynamic programming approach

• One extension: compute sub-optimal folds
  – during backtracking, try multiple paths
Question

How do you change Nussinov's algorithm to allow the computation of the stacking energy (score depends on # of next-to-next pairings)?

Hint: think affine gap penalties.