RNA structure and structural alignment

Introduction

RNA molecules are usually single-stranded (though double-stranded RNA also exists in nature) and often fold into a three-dimensional structure driven by the pairing of complementary nucleotides. The exact shape is dependent on many biophysical factors, however a reasonable approximation can be obtained using fairly simple assumptions.

RNA structure prediction (RNA folding)

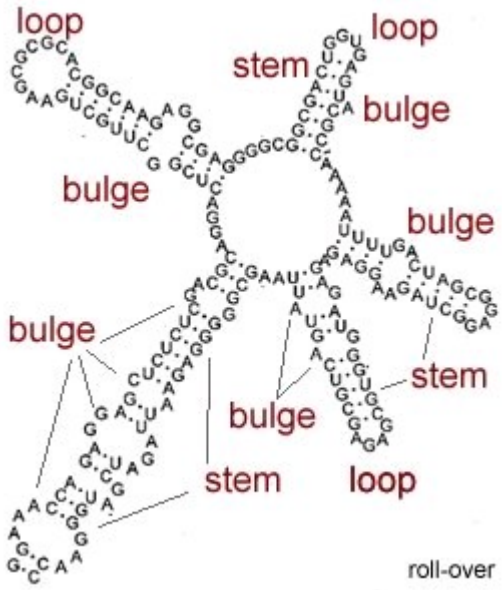


Illustration 1: RNA fold structure (from: http://virtuallaboratory.colorado.edu/Biofundamentals/lectureNotes/Topic3-2_NucleicAcids.htm)

The simplest definition of the RNA folding problem involves pairing up complementary nucleotides (A-U, C-G) in a way that maximizes the number of "contacts" - correctly paired nucleotides. This problem can be easily solved if we assume all pairings are properly nested, i.e. the structure can be expressed as a series of parantheses:

```
>Sample RNA
AAAAAAAAAAAAAGGGGGGGUUUUUUUUUUUUUUCCCCCCCCCCCCCCCCCC
.............((((((.............))))))..........
```

In many cases this assumption is not true in real life, resulting in what are called pseudo-knots - pairings of nucleotides that do not obey the "proper parenthesis" rules (see figure below). Figuring such situations out is a lot harder to do computationally and will be discussed under probabilistic folding methods below.

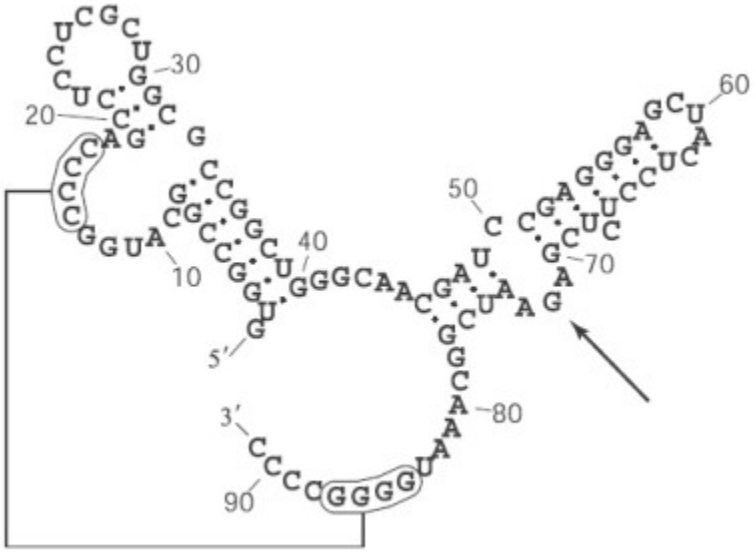


Illustration 2: Pseudo-knot - the link indicates nucleotides that are physically "connected" in the 3D conformation.

Nussinov's algorithm - Dynamic Programming folding approach

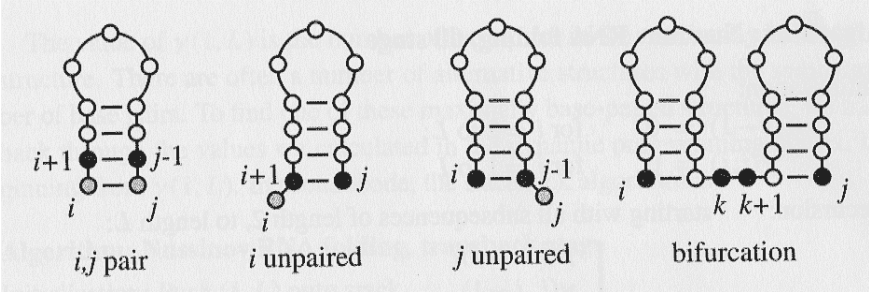
This algorithm is a variant of the CYK algorithm used to parse Chomsky Normal Form grammars.

The basic idea is to focus on the simpler problem of folding a short stretch of RNA - i.e. the dynamic programming recurrence will build upon value V[i,j] - the number of contacts in the best fold of substring S[i..j].

We can distinguish four different cases:

1. V[i,j] = V[i+1, j-1] + 1 - if nucleotides S[i] and S[j] match
2. V[i,j] = V[i+1,j] - nucleotide S[i] is unpaired
3. V[i,j] = V[i, j-1] - nucleotide S[j] is unpaired
4. V[i,j] = $\max_{i<k<j}$ (V[i,k] + V[k+1,j]) - fold of S[i..j] is built from two subfolds of S[i..k], S[k+1,j]

At each step in the algorithm, V[i,j] will be set to the maximum of the 4 cases specified above (see figure below)



As initial values, the DP matrix can be filled in with 0s for V[i,i] and V[i, i+1] as adjacent nucleotides cannot be paired up. Also, if we expect that loops cannot have fewer than k nucleotides, we could set V[i, i+j] to 0 as well for all $j < k$.

Once the initialization is complete, the DP matrix can be filled in, diagonal by diagonal (for each diagonal the difference between i and j is constant) starting from the main diagonal (i-j = 0) as all the information needed will be available in previous diagonals. V[1,n] will eventually contain the maximum number of contacts achievable.

If we want to find out all the pairings, the algorithm will need to backtrack from V[1,n] using a similar approach as that described for sequence alignment. There is one catch, however, the backtrack pointers form a tree structure due to the bifurcation operation (case 4 above) - and the backtrack needs to recurse on either side of the bifurcation.

Zucker's algorithm - Extension to actual energy functions

Nussinov's algorithm simply assumes that each nucleotide pairing is equally good, and also that there are no penalties for unpaired bases - both assumptions that are incorrect in the real world. To account for these factors Michael Zucker extended Nussinov's algorithm to incorporate energy terms (see http://www.bioinfo.rpi.edu/zukerm/rna/energy/ for a list of the structures being evaluated). The goal here is to find the fold that minimizes the free energy of the resulting structure - a more biophysically reasonable objective function than # of base pairings.

A full description of Zucker's algorithm (implemented in the program mfold) is beyond the scope of this document. I just want to mention two features. First, the algorithm allows the computation of sub-optimal folds by performing several rounds of (imperfect) backtracking with the goal of identifying structures that have similar energies but differ significantly. The assumption is that due to a variety of reasons (e.g. energy terms cannot be correctly computed and differ from actual energies in the real system) the actual structure might not be the one that minimizes the energy, rather is one of the many structures with low energy.

A second feature of Zucker's algorithm is that it allows the computation of "stacking" energies - taking into account the fact that the energy of a stem is not simply the sum of the energies of the individual base-pairings. To account for length-specific stem energies the algorithm uses a

similar approach as that used in computing affine gap penalties in sequence alignment.

Probabilistic folding

The folding approaches described above assume a fair amount of prior knowledge - which bases can be paired, what are the energy terms for certain simple structures, etc. Assuming we had a large number of sequences that we know fold the same way (e.g. because they do the same things), could we learn the fold itself?

A simple approach starts by taking all the sequences and constructing a multiple sequence alignment. Within this alignment (assuming it's well built - which is generally not the case given that multiple alignment is NP-hard) we can assume that each column represents a same position within the fold common to all the sequences. We can observe that if two nucleotides are paired up in the common fold, then these nucleotides need to be complementary in all the sequences, i.e. the corresponding columns in the alignment are correlated. A way to measure this correlation is through the concept of mutual information - how much information can we learn about column j given that we know the content of column i.

Mathematically, the mutual information between columns i and j can be expressed as:

$$M(i,j)=\sum_{x_i,x_j} f_{x_i x_j} \log_2 \frac{f_{x_i x_j}}{f_{x_i} f_{x_j}}$$

where x_i and x_j are the corresponding nucleotides in the two columns within the aligned sequences, and the f terms are the frequencies of a specific pairing or nucleotide, respectively, in the whole set of sequences. M(i,j) varies between 0 (no correlation) and 2 (full correlation). By computing M(i,j) for all pairs of columns in the alignment we can identify those positions that are likely paired up in the final structure. If we assume the structure is properly nested we can use a variant of Nussinov's algorithm to maximize the mutual information between paired-up positions. In the non-nested case (pseudo-knots) the problem is likely NP-hard (I still need to look this up).

Structure alignment

When aligning sequences it often the case that at least one of the sequences being aligned has some known structure. An example are 16S ribosomal RNA databases - the structure of the ribosomal RNA is known, now we want to use this information to guide the alignment of a new sequence against the database.

Jiang et al. (paper on syllabus page) describe several variants of this problem depending on whether the structure is nested or not, and on whether the structure is known for one or both of the sequences. In brief, the (NOSTRUCTURE, NOSTRUCTURE) alignment is simply the traditional sequence alignment algorithm. (NOSTRUCTURE ,NESTED) can be solved with a variant of Nussinov's algorithm, and all other variants ((NESTED, NESTED), (NON-NESTED, NOSTRUCTURE), (NON-NESTED, NESTED), (NON-NESTED, NON-NESTED)) are NP-hard. Note that formalizing the structure-sequence alignment can be challenging - Jiang et al. create a

set of scores for the various ways in which pairings are broken up during alignment.

An alternative approach is a probabilistic alignment framework similar to the profile HMMs described earlier in the class. A good example is the Infernal package from the Eddy lab (see ftp://selab.janelia.org/pub/software/infernal/Userguide.pdf) which relies on stochastic context free grammars to take into account the pairing of columns in a multiple sequence alignment then extends the grammars in a similar way HMMs extend the standard profile alignment.