Computational Biology
Lecture 20: RNA secondary structures
Saad Mneimneh

As you might recall, unlike the DNA, an RNA molecule is a single stranded chain of the nucleotides A, C, G, and U (nucleotide U, for Uracil, replaces nucleotide T of the DNA, A and U are still complementary). For this reason, a nucleotide in one part of the RNA molecule can still base-pair with a complementary nucleotide in another part of the RNA molecule. Therefore, the RNA molecule folds on itself (in 3D) forming a secondary structure. We would like to predict the secondary structure of the RNA. This can be more or less determined by answering the following question: Given the RNA sequence, which bases pair with which?

Representation

The RNA secondary structure is typically represented by a two-dimensional picture (although folding is 3D). As an example, consider the RNA $r = AACGGAACCAACAUGGAUUCACGCUUCGGCCCUGGUCGCG$ and its secondary structure shown below:

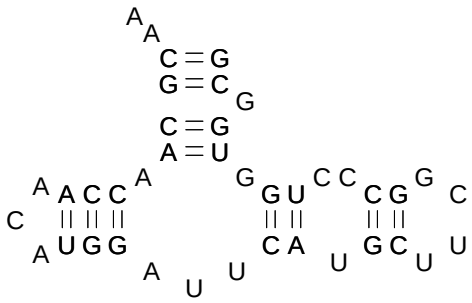

Figure 1: RNA secondary structure

The double lines in the figure above represent bondings between base pairs. Based on the above representation, we can identify several components of an RNA secondary structure that occur frequently:

- unstructured single strand
- stem (helical region)
- hairpin loop
- interlnal loop
- bulge loop
- branching loop

These components are illustrated in Figure 2 below. An unstructured single strand is simply a part of the RNA sequence that does not fold, i.e. does not form any base pairs. A stem consists of two parts of the RNA sequence that form a DNA-like helical structure (double stranded). A single stranded subsequence bounded by base pairs is called a loop. A simple substructure consisting of a simple stem and a loop is called stem loop or hairpin (because the structure resembles a haipin when drawn). Single stranded bases occuring within a stem are called a buldge loop if the single stranded bases are on only one side of the stem, or internal loop if there are singles tranded bases interrrupting both sides of a stem. Finally, there are branching loops from which three or more stems radiate.

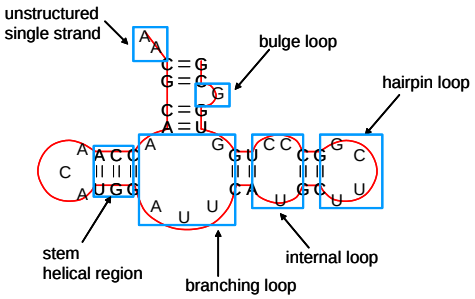

Figure 2: Components of an RNA secondary structure

The secondary structure of an RNA $r = r_1...r_n$ can be described as a set S of *disjoint* pairs (r_i, r_j), where $1 \leq i < j \leq n$. If we view the bases as vertices in a garph, and all possible pairs as edges, then the secondary structure is a matching in the graph $G = (V, E)$. Let $G = (V, E)$ such that:

- V: contains a vertex for every base r_i, $i = 1..n$
- E: contains an edge (u, v) iff $u, v \in V$ are complementary bases

Then a matching in G is a valid secondary structure for the RNA r. However, modeling the problem as a matching problem is not very useful for us because we would like to exlude a special configuration known as a knot. A knot exists when r_i is paired with r_j and r_k is paired with r_l such that $i < k < j < l$, i.e. (r_i, r_j) and (r_k, r_l) are overlapping pairs. We would like to consider only nested pairs because knots are infrequent. The figure below illustrates an example knot.

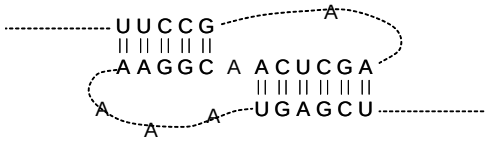

Figure 3: A knot

Energy

Among all possible secondary structures, which structure should we pick? The RNA molecule folds into the minimum free energy structure: each base pair (r_i, r_j) (r_i and r_j are complementray) contributes a negative energy $\alpha(r_i, r_j) < 0$, and $\alpha(r_i, r_j) = 0$ otherwise. Therefore, we would like to obtain the minimum free energy structure (if $\alpha(r_i, r_j)$ is the same for all base pairs, this is the structure with maximum number of base pairs). If we allow knots, the problem reduces to computing a minimum weighted matching in the graph G defined above, where an edge (r_i, r_j) will have a weight $\alpha(r_i, r_j)$ (if $\alpha(r_i, r_j)$ is the same for all base pairs, the problem reduces to computing a maximum cardinality matching in the graph G). But as mentioned earlier, we would like to avoid the formation of knots. We will develop a dynamic programming approach to solve the problem.

Formulation

Let $E(S)$ be the total free energy for a set of pairs S, i.e $E(S) = \sum_{(r_i, r_j) \in S} \alpha(r_i, r_j)$. Let us make a simplifying assumption: Assume $\alpha(r_i, r_j)$ is independent of all other pairs and the positions of r_i and r_j in the structure (this is not necessarily true in reality). Then the minimum energy structure for a substring of RNA r, say $r_i...r_j$, is independent of the surrounding and can be computed by disregarding the $r_1...r_{i-1}$ and $r_{j+1}...r_n$ portions of r. We can, therefore, use solutions for smaller strings to determine the solutions for larger strings (i.e. a dynamic programming approach).

Algorithm

Let $S_{i,j}$ be the minimum free energy structure for $r_i...r_j$. Let's look at all the possibilities for r_j. r_j either forms a pair with some base in $r_i...r_{j-1}$ or it does not. If it does not, then $E(S_{i,j}) = E(S_{i,j-1})$. If r_j is paired with r_i, then we can say that $E(S_{i,j}) = \alpha(r_i, r_j) + E(S_{i+1,j-1})$. Finally, it could be that r_j is paired with some r_k, for $i < k < j$. In this case we can split the string in two: $r_i...r_{k-1}$ and $r_k...r_j$, and say that $E(S_{i,j}) = E(S_{i,k-1}) + E(S_{k,j})$, for some k (we can do this because we have assumed no knots). Below we list the three possiblities for computinh $E(S_{i,j})$ with visual illustration:

- r_j is unpaired: $E(S_{i,j}) = E(S_{i,j-1})$
- r_j is paired with r_i: $E(S_{i,j}) = \alpha(r_i, r_j) + E(S_{i+1,j-1})$
- r_j is paired with r_k $(k \neq i)$: $E(S_{i,j}) = E(S_{i,k-1}) + E(S_{k,j})$ for some $i < k < j$

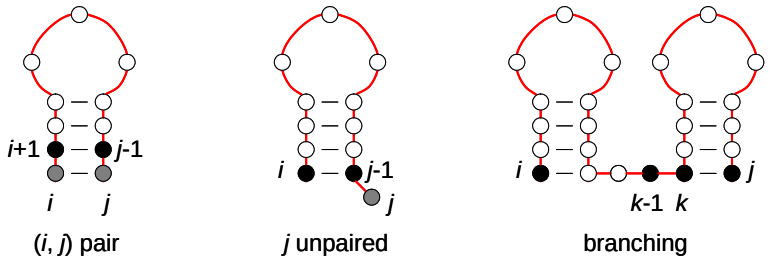

Figure 4: Three possibilities for $E(S_{i,j})$

Therefore, $E(S_{i,j}) = \min(E(S_{i,j-1}), \alpha(r_i, r_j) + E(S_{i+1,j-1}), E(S_{i,k-1}) + E(S_{k,j}))$ for all $i < k < j$. This is the basis for the dynamic programming presented below, known as Nussinov folding algorithm:

$$E(S_{i,j}) = \min \begin{cases} E(S_{i+1,j-1}) + \alpha(r_i, r_j) \\ E(S_{i,k-1}) + E(S_{k,j}) \end{cases} \quad i < k \leq j$$

$$E(S_{i,i}) = 0 \qquad i = 1..n$$
$$E(S_{i,i-1}) = 0 \quad i = 2..n$$

Note that when $k = j$ in the above program, we cover the first case of r_j being unpaired since $E(S_{j,j}) = 0$. At the end of the algorithm, the value for $E(S_{1,n})$ is the energy of the lowest free energy (i.e. optimal) secondary structure for the RNA $r = r_1...r_n$. The optimal structure itself can be obtained by the now familiar back tracing strategy.

The following figure illustrates the initialization step. Although $E(S_{i,i-1})$ does not correspond to any real structure (the string $r_i...r_{i-1}$ does not exist), it is needed for the computation of the entry $E(S_{i-1,i})$, and is therefore set to zero. Note that entries in grey are never used.

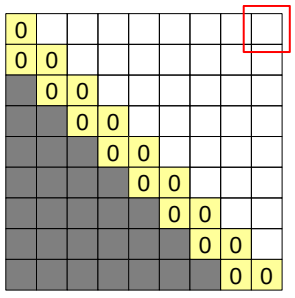

Figure 5: Initialization step

The following figure illustrates the computation process. Entry $E(S_{i,j})$ depends on $E(S_{i+1,j-1})$, $E(S_{i,i}) + E(S_{i+1,j})$, $E(S_{i,i+1}) + E(S_{i+2,j})$, ..., and $E(S_{i,j-1}) + E(S_{j,j})$.

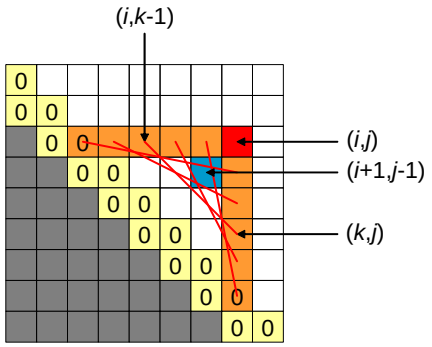

Figure 6: Computation of $E(S_{i,j})$

Therefore, the running time of the algorithm is $O(n^3)$ since each entry requires $O(n)$ time of computation and we have $O(n^2)$ entries.

Example: $r = GGGAAAUCC$, $\alpha(C,G) = \alpha(G,C) = -1$, $\alpha(A,U) = \alpha(U,A) = -1$, $\alpha(x,y) = 0$ otherwise.

	G	G	G	A	A	A	U	C	C
G	0	0	0	0	0	0	-1	-2	-3
G	0	0	0	0	0	0	-1	-2	-3
G		0	0	0	0	0	-1	-2	-2
A			0	0	0	0	-1	-1	-1
A				0	0	0	-1	-1	-1
A					0	0	-1	-1	-1
U						0	0	0	0
C							0	0	0
C								0	0

Figure 7: Example

In the example above, the optimal structure has a minimum free energy of -3. The optimal structure itself can be obtained by tracing back through the dynamic programming table. A diagonal trace from (i,j) to $(i+1,j-1)$ indicates a pair (r_i, r_j).

	G	G	G	A	A	A	U	C	C
G	0	0	0	0	0	0	-1	-2	-3
G	0	0	0	0	0	0	-1	-2	-3
G		0	0	0	0	0	-1	-2	-2
A			0	0	0	0	-1	-1	-1
A				0	0	0	-1	-1	-1
A					0	0	-1	-1	-1
U						0	0	0	0
C							0	0	0
C								0	0

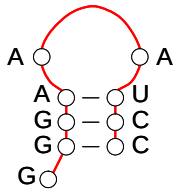

Figure 8: Tracing back (in red)

The trace back takes linear time $O(n)$ if we keep back pointers, and obviously $O(n^3)$ time if we verify the computation backward.

The trace above is unbranched. In general we need to keep track of multiple traces due to the rule $E(S_{i,j}) = E(S_{i,k-1}) + E(S_{k,j})$ which splits the trace at (i,j) into two branches. Another detail that we should consider is the following: if $\alpha(r_i, r_j) = 0$, then we may have many options that result in the same value for $E(S_{i,j})$; for instance, we may have $\alpha(r_i, r_j) + E(S_{i+1,j-1}) = E(S_{i+1,j-1}) = E(S_{i,j-1})$. In this case, we should not favor the pair (r_i, r_j) since r_i and r_j may not be complementary. Can we have $\alpha(r_i, r_j) = 0$ with $\alpha(r_i, r_j) + E(S_{i+1,j-1}) = E(S_{i+1,j-1})$ being the

unique minimum among all possibilities. The answer in no, since we can easily prove that $E(S_{i+1,j}) \leq E(S_{i+1,j-1})$ and $E(S_{i,j-1}) \leq E(S_{i+1,j-1})$. Therefore, $E(S_{i,k-1}) + E(S_{k,j})$ also achieves a minimum for $k = i + 1$ and $k = j$.

Below we present an algorithm for tracing back which (correctly) does not favor pairs. The same preference should be made if pointers for the trace are kept while the computation is performed.

Tracing back algorithm

trace $(1, n)$

trace (i, j)
if $i < j$
then for $k = i + 1$ to j
do if $E(S_{i,k-1}) + E(S_{k,j}) = E(S_{i,j})$
then trace $(i, k - 1)$
trace (k, j)
return

[*if here, it must be that* $E(S_{i+1,j-1}) + \alpha(r_i, r_j) = E(S_{i,j})$ *and* $\alpha(r_i, r_j) < 0$]
pair (r_i, r_j)
trace $(i + 1, j - 1)$
return

References

Setubal J., Meidanis J., Introduction to Computational Molecular Biology, Chapter 8.
Durbin R. et al, Biological Sequence Analysis, Chapter 10.