Übung: RNA Sequence Design

Wintersemester 2014/2015, AST
Assignment due date: 18.1.2015
Exercise on 5.1.2015 and 12.1.2015

RNA is a ubiquitous and versatile biomolecule, transferring information from the genome to the ribosome in the form of messenger-RNA (mRNA) as well as performing various regulatory and catalytic functions in the form of noncoding RNA (ncRNA). Examples of ncRNA are the well-known ribosomal RNA (rRNA), transfer RNA (tRNA), as well as microRNA (microRNA), small nuclear RNA (snRNA), and many many many more.

Apart from its many roles in biology, RNA can be designed to self-assemble into complex geometrical shapes that might be useful for therapeutics and nanotechnological applications.

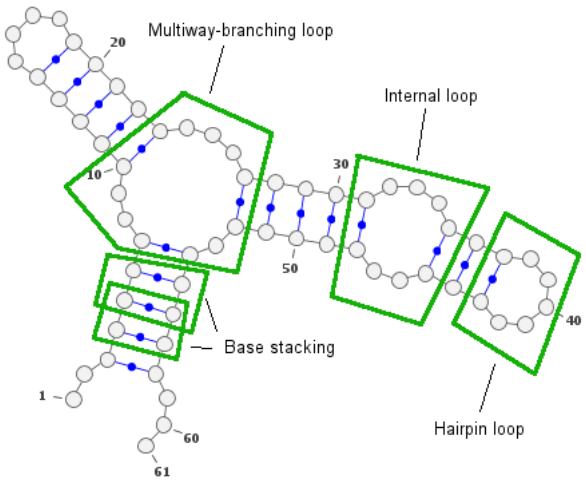
In this exercise, we will try to find ("design") an RNA sequence that will fold into a given RNA secondary structure.

One advantage of designing RNA compared to proteins is the availability of structure prediction methods for RNA secondary structures based on the so-called nearest-neighbour energy model. This model attempts to describe the free energy change when going from the unfolded state to a given secondary structure. The secondary structure is decomposed into loops (see figure 1), and each loop has a contribution ΔG_{loop} depending on the loop size and the bases contained within. This model allows us to predict the structures of designed sequences on the computer and compare them to the structure we want our RNA to fold into.

RNA secondary structure prediction

We will predict an RNA secondary structure and see which information we can gain from this prediction. Go to `http://www.pdb.org/` and search for the structure `6tna`, a typical tRNA. Download the pdb file and take a look at it with chimera, which can be found at `/home/matthies/bin/chimera` or `/usr/local/zbhtools/chimera/1.9/bin/chimera`.

Visit the NDB server at `http://ndbserver.rutgers.edu/` and find `6tna`. In the bottom right corner there is a box labelled "RNA View". If you click on the small picture inside the box, a larger image will appear. It shows a two-dimensional picture of the RNA, a so-called base pair graph depicting all base pairs, as well as additional contacts found in the three-dimensional structure. Compare the graph to the three-dimensional structure in chimera.



Can you recognize the helices in the three-dimensional structure which are shown in the two-dimensional picture?

Now we will try and predict the secondary structure starting from the RNA sequence. Download the RNA sequence from the PDB (download as FASTA format) and head over to the Vienna-RNA web servers found at `http://rna.tbi.univie.ac.at/`. Choose the RNAfold web server, put in the sequence of `6tna` and send off the web form.

The result page contains information on the most-likely structure, the minimum free energy structure or MFE structure as well as information about the ensemble (all possible secondary structures). You can see the free energy of folding and picture of the MFE secondary structure.

Also shown is the probability of the MFE structure occurring. When we design an RNA sequence, we would like the MFE structure to be the target structure and its probability as high as possible (i.e. near 100%).

Base pairs

In the nearest-neighbour model, the possible base pairs are the Watson-Crick base pairs G-C, C-G, A-U, U-A and the "wobble" base pairs G-U, U-G.

MFE structures and partition functions

As we have already seen, the simplicity of the RNA nearest-neighbour model allows various interesting things to be computed, amongst them the minimum free energy structure and the partition function, in polynomial time. This is remarkable because there are quite a few secondary structures: on the order of 2^n secondary structures for sequences of length n. This speedup is achieved by a technique called dynamic programming, which can be used on certain classes of optimisation and counting problems.

The meaning of the MFE structure should be clear: it is the structure with the lowest free

energy of folding.

The partition function Z is computed from the sum of Boltzmann factors over all structures:

$$Z = \sum_{\text{structure } s} e^{-G_s/kT}$$

Here, G_s is the free energy of folding into secondary structure s, k the Boltzmann constant and T the temperature.

With the partition function, we can compute the probability of any secondary structure occurring in the ensembles of structures

$$p_s = \frac{e^{-G_s/kT}}{Z}$$

You can see from this formula that every possible structure will have a certain probability.

Having the partition function available is very useful, because we can compute the probability of the target structure (the one we want to design sequences for) for a given candidate sequence.

Dot-bracket / Vienna notation for RNA secondary structures

RNA secondary structures without pseudoknots (non-nested base pairs) can be written as a string made of dots "." (unpaired bases) and brackets "()", which stand for paired bases. A simple hairpin loop would be written as `(((...)))`.

You can look at secondary structures with the program Varna, which can be found at

`/home/matthies/bin/varna`

Algorithm for parsing a secondary structure in dot-bracket notation

```
/* input: vienna (string in dot-bracket / Vienna notation) */
stack = []
bp = []    /* array of length n */
for (i = 0; i < length(vienna); i++)
    if vienna[i] == '.'
        bp[i] = UNPAIRED
    else if vienna[i] == '('
        push(stack, i)    /* push i onto stack */
    else if vienna[i] == ')'
        if stack is empty
            error(``too many closing brackets'')
        j = pop(stack)
        bp[i] = j
        bp[j] = i
```

```
    else
        error(‘‘unknown symbol’’)
if stack isn’t empty
    error(‘‘not enough closing brackets’’)
return bp
```

RNA sequence design by simulated annealing

Given that we can quickly predict the secondary structure of an RNA sequence, we can now iteratively design an RNA sequence on the computer until the sequence is predicted to fold into the target structure.

We will optimise sequences by simulated annealing, an optimisation algorithm mentioned in the lectures.

Simulated annealing

```
s = random
s_best = s
for (step = 0; step < maxstep; step++)
    s_trial = random_move(s)
    delta_c = cost(s_trial) - cost(s)
    if delta_c < 0
        s = s_trial
    else
        r = rand(0..1)
        if exp(- delta_c / T) >= r
            s = s_trial
    if cost(s) < cost(s_best)
        s_best = s
    T = epsilon * T    /* decrease temperature */
return s_best
```

Our cost function will be

```
cost(s) = - prob_of_target_struct(s)
```

i.e. the negative of the probability of the target structure for the sequence `s`.

The `random_move` function should change the RNA sequence at a random position. If the position is base-paired in the target structure, it should change both positions so that a base pair is still possible.

Calling the cost function

To compute the probability of a target secondary structure, use the script `/home/matthies/uebung-rna-design/targetp`, which can be called in the following way:

```
/home/matthies/uebung-rna-design/targetp GGGAAACCC '(((...)))'
```

Inside your program, you will have to call this script. In Ruby, you can do it like so (note the backticks ‘):

```
p = ‘/home/matthies/uebung-rna-design/targetp GGGAAACCC '(((...)))'‘.to_f
```

or

```
seq = 'GGGAAACCC'
vienna = '(((...)))'
p = ‘/home/matthies/uebung-rna-design/targetp #{seq} '#{vienna}'‘.to_f
```

Other programming languages will have similar functions, often named `system` or something similar. Ask if you have problems getting this to work in your programming language.

Note: the `targetp` script will output a warning if the sequence is incompatible with the given secondary structure, i.e. if there are illegal (not GC, AU, or GU) base pairs.

Assignment

1. Perform the secondary structure prediction (with Vienna RNA) for the sequence of `6tna` (from the PDB) and compare it to the secondary structure as found in the NDB. What are the differences? You may use the 3d structure from the PDB to help explain the differences, if there are any.

2. Size of the search space

   How many sequences are compatible with the secondary structure `.....` (dot-bracket notation)?

   How many sequences are compatible with the secondary structure `((...))` (dot-bracket notation)?

   The allowed base-pairs are G-C, C-G, A-U, U-A, G-U, U-G.

3. Explain in words or by using the formulas for the partition function why the MFE structure is the most probable structure.

4. Someone claims that it is sufficient to find sequences that have the target structure as the MFE structure. Why could this approach lead to designed sequences that are not useful in practice? (Assume that the RNA nearest-neighbour model is correct, i.e. this question isn't about if the model is realistic enough.)

5. Implement the simulated annealing algorithm. Your program should take the target secondary structure in dot-bracket notation from the command-line, as well as a seed value for the random number generator and the maximum number of steps to run the simulated annealing algorithm. The output should be the best found sequence and the probability of the target structure for this sequence.

   To perform the random moves in sequence space during simulated annealing, you will have to implement the algorithm for parsing a secondary structure.

   What is the best sequence you can find for the target structure `(((((((...)))))))` ? The best sequence found will win a prize.

   If you cannot program, run the algorithm by hand for at least five steps. Write down the individual steps, i.e. step number, current temperature, current sequence, trial sequence and which sequence was chosen for the next step.

Bonus assignment

1. Design sequences for the secondary structure of 6TNA (after cleanup of pseudoknots, etc.). The secondary structure is `(((((((..((((........))))((((((.......))))))).`
   `...(((((.......)))))))))))....`

2. Designing sequences for this larger secondary structure with simulated annealing can take a long time. Can you think of a way to modify the simulated annealing optimisation to make it more efficient? Or perhaps an entirely different algorithm? Either simply sketch your idea with pseudocode or implement and test it on larger target structures.