How RNA folding is predicted

Many people might hear of fields such as RNA and protein folding and wonder how that stuff must work. RNA is a physical system, therefore it must obey the laws of physics. The most commonly used physical law in the field of RNA prediction is the partition function law [TODO: better name]. If RNA is modeled as a system in thermal equilibium then the probability of any state s with energy $E(s)$ is equal to

$$P(s) = \frac{e^{-E(s)/kT}}{\sum_{s'} e^{-E(s')/kT}}$$

Therefore to predict the folding of RNA we need 3 things:

- a specification of a state of RNA s,
- an energy function, which takes a state and outputs a physical energy,
- the partition function, the sum of the Boltmann factors over every possible state $Z = \sum_{s'} e^{-E(s)/kT}$.

States of RNA

RNA is polymer of 4 nucleic acids, adenine, guanine, cytosine, and uracil. This "primary structure" of RNA can be specified with a simple string of 4 letters, for example "GAAACCCCUUUUGGGG". For the secondary structure of RNA, we are interested in which of these bases are going to pair with each other. Any state of RNA can therefore be represented as a list of base pairs (i, j). For **tractability** reasons, in practice the secondary structure of RNA is assumed to be composed of several recursive types.

```
data Structure = (Paired 1 Int, Structure ) | Structure
me :: Structure
me = Me
```

This is actually a key assumption, because the computability of this problem hinges on it. There is the question of whether it is a physically valid assumption. The answer is unquivically NO. There are many RNAs found in nature that have "crossing" pairs, which break the recursive definition here, such as Group I and Group II introns [TODO: cite someone]. However, without the recursive definition, computing the partition function is intractable, so most RNA folding software packages make this assumption.

Energy Model

For RNA, the most relevant energy to consider is electric potential. Nucleic acids are covered in exposed ions, and each free charge interacts with every other charge. The total energy can be therefore expressed as a function of the charges and the distances between them $U = f(q_i q_j \overline{r_{ij}})$.

This is far too complicated. For n charges there would be n^2 charges to consider. As we shall soon see, we want to be sure that the energy model runs in constant time. How can we do this?

Doing a full physical energy model of RNA would be very difficult to compute. In practice, there are several levels of complexity that are implemented. A very simple energy model is one where each hydrogen bond contributes exactly -1 unit of energy to the total.

Reducing the complexity of the energy model to this allows us to compute the energy recursively.

```
typedef Joul = Real

energy :: Structure -> Joule
data Structure = (Paired 1 Int, Structure ) | Structure
me :: Structure
me = Me
```

Partition Function

The sum $Z = \sum_{s'} e^{-E(s)/kT}$ initially seems daunting. There are $O(1.8^n)$ structures possible for an RNA strand of length n. However, from the recursive definition of our energy model, we can easily see that the partition function can be defined recursively as well:

[TODO: partition function definition]

This becomes a standard dynamic programming homework problem! We can turn Z into a table of values where Z[i,j] is the partition function from base i to base n. Since each row of the table only depends on the entries below and to the left, we can compute the full partition function by starting from the bottom left and working out to the top right. This is even parallelizable along the diagonals, as shown in the diagram:

[TODO: make elm diagram.]

Now that we have the partition function, we know the probability of any structure from the Boltzmann distribution.

Sampling RNA Structures

To actually predict the structure of RNA molecules in nature, we can work backwards through the partition function table to sample structures from the Boltzmann distribution with probability equal to their probability to be found in nature.

[Todo: Haskell code]

Let's try it out:

[Todo: try example]

Cool!

Macrostates

One last thing to do is cluster these structures into Macrostates. Given two structures of the same strand, a distance measure can be computed between them and with this distance metric, we can classify and compare macrostates. Let's sample a bunch of structures

Applications

Many genetic diseases are caused by mutations causing RNA to fold incorrectly. The treatment is then to find some way to get it to fold correctly again. One way to do this is to insert some binding agent to block binding to a particular base or set of bases for that strand, which disrupts the partition function, energy landscape, and causes the strand to fold correctly. For example let's assume that the healthy strand is [TODO: insert] but a pointwise mutation has caused this to mutate

to [TODO: insert]. We can compute a statistical distance to the healthy boltzmann distribution and then pick a base to block that gets us as close to the healthy distribution as possible.

[TODO: do example]

Cool right?