Lab exercise 5: 2D-grid folding - Biophysical Chemistry

April 4th 2017

1 Introduction

Previous exercises have used a state-hopping model in which we have not questioned what each micro-state *is* or represents. Nor have we cared about what determines the enthalpy of a micro-state. In this exercise we will explore this by thinking in terms of proteins, as always by making a model. This will allow us to understand the aspects of a simple yet representative case, then infer the consequences of implementing an even more complex model. In practical terms, you will understand just how complicated molecular dynamics simulations are, and why compromises are made to see interesting phenomena.

1.1 What is the enthalpy of a protein?

The enthalpy of a microstate corresponds to its potential energy, or ”internal” energy. To follow the logic of our simple models, it is definitely something which is completely associated with (hopefully can be derived from) the definition of the microstate. So how does one determine the enthalpy of a protein microstate in practice? If we use atomic positions to define our microstate, we need to have an idea of how ”good” or favourable each atomic position is with respect to all others, which in turn requires some model of how they interact. Thus **any** definition of how each atom interacts with all other atoms, provides an enthalpy. The definition we will be using will be fairly crude, but still similar (not identical) to experimentally measurable quantities.[1] In many simple models were non-physical (but still biologically relevant) enthalpies are calculated for microstates, one uses the term *scoring function* instead of ”enthalpy”. We will follow this convention, and calculate a score of how ”good” states are, but still treat this score like an enthalpy which is subject to a Boltzmann-like accept/rejection criteria in Monte-Carlo simulations.

2 Grid folding (2D)

2.1 System rules

For today's simple model, we will be running Monte-Carlo simulations of a ”protein”. We will model each amino-acid as a single point, located on an integer coordinate of a 2D-grid. The bond-length between residues equals 1. In today's universe there are only two possible amino-acids; H and W. H is hydrophobic and W is hydrophilic (w stands for water). We will attempt to fold proteins into stable structures using this model.

[1]Molecular dynamics tries to model different interactions very accurately to build the most realistic enthalpy possible, but it is also fairly pragmatic. If e.g. the partial charge of an atom needs to be lower than experimental measurements in order to reproduce quantities like diffusion rates or densities, then this overrides the experimental value.

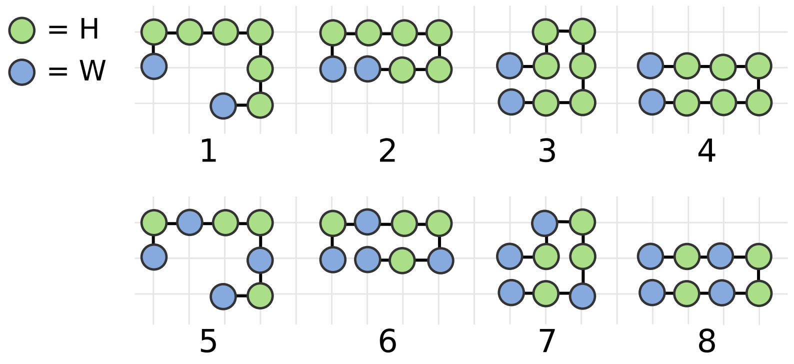
2.2 Scoring function

Let's choose a really simple scoring function (i.e. mock enthalpy) to begin with, and call it E^A. The simulation takes place in water, and we assume residue type H does not like water. Hence we define our simple scoring rule as follows;

For each position around an H-residue (maximum of 4, not diagonal) that is occupied by an H-residue, E^A is increased by 1 unit. Ignore adjacent H-residues if they are consecutive neighbors along the chain.

Tasks

1. Write down the score E^A, according to the defined rule, for the conformations in the bellow figure.
2. What qualitative difference do you see between high and low scoring conformations?



3 Simulating

You have been provided with a python-script which takes a file as input, much like the previous exercises. You are now asked to provide a file in which you only need a single line of characters, H or W. If you want to simulate a protein that contains 8 residues, all of which are hydrophobic, you should make a file called

```
protein_1.dat
```

with only one line of text:

```
HHHHHHHH
```

If you wanted to replicate the protein 8 in figure (1), you would make the line

```
WHWHHWHW
```

As before, you can get some additional information on input parameters to the program by running it with the -h flag. The basic steps in the program are as follows;

1. Try to make a random walk in 2D to get an initial protein conformation. If it does not cross itself, use it, otherwise use a staircase-like initial case.
2. Try to make N changes to randomly selected residues. This number can be controlled with the -n flag.

3. Score the new conformation according to the scoring function above. A multiplicative factor can be added through the −−score flag.
4. Accept or reject the new conformation.
5. Repeat the last three instructions for as many simulation steps as specified.

The program will also show you the highest scoring conformation found so far. If it finds something equally good to the previous best-yet state, it will update the plot anyway so that you know it changed. **Try making a very small 5-residue protein with any mix of residues and run it for a while.** Do not spend an hour running it, this small test-case is just to show you how the simulations work. But feel free to try different values for the -n and -s flags, as well as re-run simulations to see if your protein finds the same conformation each time.

3.1 Folding a protein

We have added an option to specify an output-file, which will show you the score as a function of simulation steps:

```
-e filename.dat
```

This will be useful when we start assess if we are looking at a realistic model of folding, so from now on it's a good idea to output files with new names to save your work. If you name your proteins, it's easy to keep a clean history. If you abort a simulation which uses the output-flag, there will still be output with all the data up until the simulation was aborted. Using the plot.py-file, you can plot this output data. Run the plot-file with the -h flag to see what options to specify.

1. Create a protein with 20 residues and simulate it and look at the plot of the different conformations. At each step the chain is perturbed by a limited set of movements that maintains the bond length. In a few sentences, describe how the chain is changed at each step.
2. Run the same simulation for 30 000 steps. **To make the simulation go fast use the flag `-p` *X*, which will make the program only plot every *X* step. The less frequent you plot the faster is the simulation.** Behind the scenes, however, the simulation runs just as it did without the flag. You will find that using this flag makes the simulation appear to make larger changes to the protein much faster, but it is simply performing better so that we have to wait less time to see the same behavior as before. Use the `plot.py` script to plot the energies throughout the simulation and include the plot in your report.
3. How would you define a folding sequence? What makes a folded state stable apart form a low energy (or high score)?
4. Try to find a sequence that folds.
5. Try values between 2-10 for the flag `--score`. What effects does it have on the dynamics and why?

3.2 Enhanced Sampling

If all we care about is scanning all the possible conformations to find high-scoring states, then we do not actually need to stay in high-scoring states. Using the flag

```
--noMetropolis
```

we can turn off the rejection of new conformations simply because they have bad scores. This will make the protein much more mobile, but it won't stay in high-scoring states.

1. Run a simulation with this flag. Do you see a difference?
2. Our simple model lacks a tunable temperature, but what does using this flag correspond to if put in the context of a thermodynamical system with a temperature? Is it corresponding to an increase or decrease of temperature?
3. What is the drawback of this method when simulating proteins, regarding sampling of "relevant" states?

3.3 Excursion

If you want to, you can design your own scoring function. If you think you have a good or interesting idea but don't know how to implement it, let us know and we'll help you try it out.

4 Relation to molecular dynamics

In the simple model above, there is a single interaction used to calculate a score of the microstate. But this still means that we need to calculate the distances between all pairs of H-residues. If there are M such residues, there are $M(M+1)/2$ distances to calculate. Importantly, this is proportional to M^2, so the effort grows quickly with the number of residues in the system.

If we had a score that included W-residues, then we *also* have to specify what happens when a W-residue and H-residue interact, as well as pairs of W-residues. If there are 20 amino-acids, then the possible interactions in this model become much larger and the scoring function very complicated. Molecular Dynamics (MD) uses tried and tested scoring functions which have parameters based on extremely accurate experiments and calculations. These scoring functions are called **force-fields**, because they contain all the parameters used to calculate the forces that act on all the atoms of the simulated system. These forces are then used to update the positions of all the atoms based on their current position and direction of motion, according to classical mechanics (i.e. Newton's Law of Motion). This makes MD fundamentally different from the type of simulations we have performed so far, because in general it is not based on randomizing new states or transition probabilities. In short, MD is not based on MC. Instead the law of physics are directly applied to calculate the evolution of the system given its state at each step.