Biological Computation 20.181

Homework 9

In this homework, you will use the code libraries you first saw in HW8 to implement a program to 'repack' a protein structure. Start by taking a look at the code outlined in the file hw9.py. You will find a number of convenient functions are available there for your use. In particular, there are functions to generate lookup tables of the energy of each rotamer with itself (computeRotamerSelfEnergies) and with other rotamer sidechains (computeRotamerPairs). These functions are briefly documented in the codebase itself.

To complete this assignment, you will need to code two additional functions: The first, computeRotamerEnergy, should compute the total cost for a single rotamer including rotamer-backbone and rotamer-rotamer sidechain interactions. You should try to use the lookup tables already computed for you in this function. The second, repack, will implement the bulk of the control logic. It should call your computeRotamerEnergy function to determine the cost of substituting one rotamer for another, and it should use the 'Metropolis' criterion to decide whether a randomly selected rotamer should be used.

Part 1. Implement the two missing functions, and generate a new 'repacked' version of the SH3 domain structure included in the HW directory.

Part2. Take a closer look at the behavior of your Metropolis search. Print out the total energy of the structure at each iteration, when does it converge (when do the energies stop decreasing)? Now update the range of residues being repacked to include twice as many residues. How many iterations until convergence now? What happens if you raise the 'temperature' (RT) of the simulation?

Part3. Compare your structures before(.pdb) and after(.pdb) minimizing the energy. You can do this by outputting detailed info from your program, or by looking at the two PDB files generated by eye (using PyMOL for example). Can you find specific unfavorable interactions that have been removed in the optimized structure?

Part4. Repacking an entire protein can be very computationally expensive. Use the Python profiler to determine which parts of your code end up taking the most time ("python -m profile my_program.py").

Part5. (Optional) The code given uses only Lennard-Jones terms to optimize structure, but the Energy module includes several methods for calculating electrostatic potential energies. Try including one of these in your total energy calculations.

Part6. (Optional) Use the 'buildAnonymousRotamers' function to try new amino acids in this protein. You will need either a fast computer, or a lot of time to run these calculations. When the program chooses a different amino acid for a particular site, how do these substitutions compare with the substitution matrices used for amino acid replacements in phylogenetic comparisons (*e.g.*, BLOSUM, PAM; see http://www.ebi.ac.uk/2can/tutorials/matrices.html).