Introduction to Grid computing

Protein folding

Protein folding is an extremely hot topic in medical research these days, unfortunately protein folding is extremely computationally demanding and requires a huge supercomputer to fold even the simplest proteins. Luckily the task of calculating protein foldings is quite well suited for Grid computing.

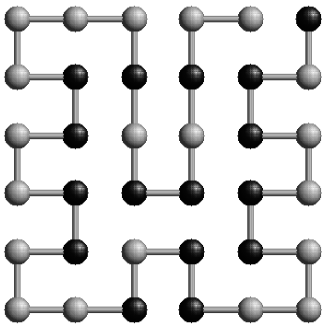
Proteins are made up of amino-acids, of which there are 20 types. Thus a protein can be viewed as a sequence of amino-acids and folding such a sequence means that the sequence 'curls up' until there is a minimum of unbound energy present in the protein.

Example of a protein fold.

As computer scientists we need not concern ourselves with the chemistry behind protein-foldings. Instead we can play with a simplified version of proteins called prototeins – proto-type proteins.

Prototeins

Our simplified prototeins are folded in only two dimensions and only in 90 degree angles. This is much simpler than real three dimensional foldings with angles depending on the amino-acids that are present at the fold, but as a model it's quite sufficient. Our amino-acids are also reduced to two types; Hydrophobic (H) and Hydrophilic (P). When our prototein is folded it will seek the minimal unbound energy, modeled by the highest number of H-H neighborships.



A prototein with 15 H-H bindings.

Even though prototeins seems very simplified we can still leatn quite a lot about real protein-foldings from the way the prototeins are folded.

You can read more on prototeins in:
http://www.americanscientist.org/template/AssetDetail/assetid/15717.

Exercises

a) How can you design an method where all possible foldings are examined? We want to make sure that we do not accidentally skip a folding that is in fact the best one!

b) How can you formalize the method from a) into an algorithm that a computer can execute?

c) Calculate the complexity of the algorithm in b) – i.e. how does the execution time grow as the length of the prototein grows? This question is central in all algorithms in computer science. If the time it takes is not influenced by the length of the prototein we note it as constant time O(1). We can also have algorithms that increase linearly in time by the length of the input which is denoted by O(n). Other kinds of problems exists where each data item is processed once for each other item which is called $O(n^2)$ or polynomial time. However things can get much worse than this and some algorithms can take exponential time, i.e. the time taken to go through all numbers that may be represented by n-bits is $O(2^n)$.

If you can't come to an analytic solution, try experimenting with different length prototeins and see if you can find a correlation.

d) As stated previously protein folding is a super-computer problem – and prototeins are just as hard to fold. How can we parallellize the process to use more machines?

e) Take the prototein code available with this assignment and transform it into a parallel version using OfficeGRID.

f) In our very simple grid-folding we actually test a lot of identical foldings, i.e. whether we start by going left, right, up or down – has no influence on the final energy level, mathematically these are simply rotations of the same folding. How many redundant foldings are we checking? How do we alter our algorithm so that it eliminates the redundant foldings?

f) How much do we gain by eliminating the foldings? Experiment with the two versions of the code!