Assignment 1

Yannick Vinkesteijn

May 5, 2020

1 Introduction

Computational protein folding has been a very vibrant field in computational science for quite some years now . A process that happens in only a few micro- to milliseconds in biological settings has proven to be extremely complex and time consuming when trying to solve computationally. In 1985 Dill proposed a 2D protein model to study folding behavior mathematically and computationally. The model uses two amino acid classes, hydrophobic (H) and polar (P), which together form a protein string. Besides these hard bonds H amino acids can form non-covalent 'soft' bonds with other H compounds directly neighboring each other that gives the protein its secondary shape. According to Gibbs free energy theory proteins will fold towards maximum stability by reducing free energy. In case of the HP model forming soft bonds between H compounds through folding reduces free energy and therefore improves stability. Every H amino acid has four binding spots in total of which either one or two spots are already taken by hard bonds, based on being on the outside of the chain. Based on these assumptions we can describe the free energy of a protein by equation 1 where H represent the total number of hydrophobic compounds and H_{out} the number of H amino acids on the outside of protein.

$$E = 2H + H_{end} \tag{1}$$

Although this energy value E represents the absolute upper bound of potential stability gain based on the limitations of both the grid and bonding sites this score is rarely the true optimum. Because of the lack of descent verification algorithms and a state space that expands exponentially ($3^n - 1$) it is no surprise that this problem is considered to be NP-hard. Given that this problem can not be solved in reasonable time many studies have tried to solve the problem with a wide variety of optimization algorithms often with moderate success.
Considering the hardness this paper will tackle the problem in a more structural setting with the use of a beam search algorithm to get a better understanding of what makes a protein hard and find possible markers that can be used to optimize the search trajectory. Beam search is a best first search algorithm that uses a search tree structure. Based on the hyperparameter beam width decides how many of the partial solutions are continued to the next layer. By investigating the effect of different beam widths on the solution quality we expect to capture information on what makes a protein hard to solve and if early success can be a good marker for future success.

2 Methods

All code will be written in python 3.8.x.

2.1 Protein Generation

As benchmark we will randomly generate 100 unique proteins for the proteins with the length of 10, 20, 50 and 100 amino acids. Every amino-acid has a uniformly distributed equal change of being either hydrophobic or hydrophilic.

2.2 Folding

Proteins will be placed one amino acid at the time on a neighboring node of the previous amino acid in a 2D or 3D lattice grid. A node is considered to be a neighbor if it exists in the von Neumann neighborhood of the previously placed amino-acid node. A node can only be occupied by one amino acid and bonds can only formed between direct neighbors. Therefore hard bonds can not cross each other. A (partial) protein will only be considered a valid candidate if all amino acids can be placed, otherwise it will be discarded. The first two amino acid will be fixed with the first in the middle of the grid and the second placed directly left of the first one. This is done to remove rotation symmetry. Additional pruning will be investigated to limit other forms of symmetry while making sure the algoritmic complexity will not outweigh the benefits of exluding some stereoisomers.

2.3 Beam Search

For beam search algorithm a tree structure is created in a breath-first manner, where every node represents a partial protein. A child represents a copy of the parent with an additional amino acid at the tail end of the protein in a any valid direction. A parent therefore can have 3 children at most in 2D and 5 in 3D, based on the fact that one of the direct neighbors is always occupied by the previous amino acid in the chain. The children are sorted best first based on the stability score and depending on the beam size the first k children are selected for the next layer. Ties are solved with uniform random selection if needed.