Implementation and Analysis of DEE Extension

In protein redesign one of the main challenges is the ability to evaluate many different conformations of the redesigned protein in order to determine the best structure. As we discussed in class, this problem is usually simplified by using the dead-end elimination (DEE) algorithm to provably get rid of bad protein conformations. DEE was first developed for application to protein structure evaluation by Desmet et al. [1]. The idea of the algorithm is that it identifies “dead-ends” of protein sub-structures that can no longer be part of the global minimum energy conformation (GMEC) and then prunes them from further consideration. This is run iteratively so that newly pruned structures can give new information about other structures to prune, until all the structures that can be pruned are.

This algorithm takes as input a set of rotamers, a protein structure, a pairwise energy function, and a set of mutations to make to the protein. A rotamer is a discrete set of side-chain conformations that is used to discretize the side-chain conformational space. The set of mutations are all the possible residue mutations to the protein that will be considered by the algorithm. The algorithm first computes the pairwise energy between all possible pairs of residue structures (rotamers) at each position. The dead end elimination criteria is then used to determine whether a given rotamer at a certain position can not possibly contribute to the GMEC and thus, be provably pruned. After several iterations to prune as many structures as possible, DEE outputs the possible structures that are consistent with the input structure and energy function.

Over the past 16 years there have been many improvements to the DEE pruning criterion. The original paper by Desmet et al. used what is called the singles and pairs elimination criterion. The singles elimination criterion is the most basic pruning criterion which states that if a particular rotamer at a particular position cannot possibly give a better energy than another rotamer at the same
position, then the first rotamer can be pruned from consideration. The pairs criterion is slightly more complex in that a given pair of rotamers A and B at two positions cannot both be part of the GMEC if there is another pair C and D in the same positions that always give a better energy. One main extension to the algorithm was made by Goldstein [2]. This basically says that a rotamer can be pruned if there is always another rotamer at the same position that can give a better energy. The main difference between this and the earlier criteria is that the “better” rotamer doesn’t have to be the same rotamer, there just must always exist a better one.

There have been many other improvements to the pruning criteria, and the most recent extensions by the Donald lab have dealt with incorporating protein flexibility into DEE. The recent advances have dealt with addressing both side-chain flexibility [3] and backbone flexibility [4]. These extensions have been used in combination with a protein redesign workflow by the Donald lab to redesign protein active sites and protein cores. These extensions have shown promise, but their implementation requires a protein ligand of at most size one. This is disadvantageous since many proteins bind a sequence of amino acids instead of just a single one. An extension to the implementation that is able to evaluate ligands of multiple lengths would most likely improve the robustness of the current workflow implementation.

The past semester I spent my time understanding the Donald lab protein redesign workflow and implementation, focusing on DEE, with the goal of changing the implementation to accept protein ligands of multiple lengths. For my project, I would like to continue this work by finalizing the implementation, use the implementation to redesign a protein’s ligand, and also analyze various parts of the algorithm. Finalizing the implementation should be a fairly straightforward continuation of the work I was doing last semester. After the implementation is completed, I would like to test the algorithm out on a real biological example. I will apply the algorithm to the PDZ binding domain system in order to predict strong binding ligands.

The PDZ binding domain is a widespread protein module that serves many functions in
proteins. In particular, PDZ domains play a prominent role in synapse formation and have also been linked to cystic fibrosis [5,7]. The canonical PDZ domain structure consists of five to six beta-strands and two alpha-helices. Usually, the C-terminus of the ligand protein binds to a groove in the PDZ domain between $\beta_2$ and $\alpha_2$ to form a beta sheet together with $\beta_2$ and $\beta_3$. There are two main classes of PDZ domains which have two different binding motifs. One class tends to bind the sequence $x(S/T)x(V/I/L)$ and the other $x\Phi x\Phi$ where $\Phi$ represents hydrophobic residues. Generally, these four C-terminus residues determine the specificity of the PDZ domain to the ligand, but further ligand positions up to about the 7th residue may also contribute specific interactions. These more peripheral interactions are more specific to individual PDZ domains instead of the PDZ domain classes [6]. A given PDZ domain can often bind multiple ligands so being able to evaluate the relative binding specificity and ligand affinity is very important for these domains.

One significant reason for understanding PDZ domain binding is the potential role they have in cystic fibrosis. The cystic fibrosis transmembrane conductance regulator (CFTR) is mutated in patients that have cystic fibrosis and the most common mutation of this protein leads to it being inefficiently synthesized and rapidly degraded. It has been shown that the PDZ domain of CFTR Associated Ligand (CAL) binds to the C-terminus of CFTR and that the knock-down of CAL can rescue the CFTR mutant function [7]. Thus, there is much evidence that if a peptide inhibitor of the CAL PDZ domain could be designed this could be used to potentially reduce the symptoms of cystic fibrosis.

A study was conducted to determine the PDZ domain ligand specificity for three different PDZ domains [6]. The study analyzed the PDZ domains of the All-1 fusion partner 6 (AF6), the ERB2 interacting protein (ERBIN) and the alpha-1-syntrophin (SNA1) for which ERBIN and SNA1 have solved crystal structures. The PDZ domains were each screened for binding against a library comprising 6223 human C-terminal peptide sequences in order to obtain an overview of the targeted sequence space diversity. Affinities in the $\mu$M range were obtained and significant analysis was done
on the binding for each PDZ domain which should allow for comparison with computational prediction.

I will analyze the ERBIN and SNA1 PDZ domains using DEE elimination to evaluate the predictive ability of the algorithm as well as more fundamental aspects of the algorithm. Both these domains bind ligands of length 4-8 amino acids so it will be important that I implement my ligand extension to the existing DEE implementation before I can do any analysis. The first part of my analysis will involve evaluating the predictive ability of the algorithm. The authors of the previous study reported the top 100 ligands that bound to each PDZ domain. By comparing this list to the predicted ligand structures, I will be able to evaluate how well the algorithm is predicting biologically relevant information, and if it is erroneously pruning good structures. Although not all possible ligands were screened, I will know which ones were and be able to evaluate their positioning in the outputted predicted structures.

The recent improvements of including backbone and side-chain flexibility should be very useful for trying to redesign a ligand. A ligand is much smaller than a protein and thus, is much more flexible and less constrained than the actual protein. I will try to evaluate the effect of incorporating either backbone or side-chain flexibility into DEE and determine which one is more beneficial to this ligand design, if any. Since backbone flexibility seems like it will be very important to correctly modeling a ligand, I will also try to enumerate different starting backbone conformations and then redo the analysis to see if this helps the predictive process. Although for this particular system, this might not be that important because it has been shown that the PDZ domain ligand’s backbones have very small RMSDs [6].

In addition to analyzing the predictive accuracy of the algorithm, I would also like to analyze the effect the different DEE criteria have on the actual runtime, and pruning of the algorithm. By using different criteria this should allow either more or less things to get pruned which will hopefully increase the speed of the algorithm or the pruning ability of the algorithm. This has been done on
other systems, but it will be interesting to note if redesigning a ligand instead of a protein will alter any of these factors significantly.

The above proposal should be sufficient for my class project. If for some reason I have time or feel that it is needed, either a different energy function or a different rotamer library could be used and the effects of these changes evaluated. Changing either one of these is theoretically trivial from the algorithmic point of view, but could contain many caveats in the actual implementation. Neither of these changes would directly affect the algorithm itself, but they could be used to understand the sensitivity of the algorithm.

References