UNIVERSITY OF CALIFORNIA

Los Angeles

Designing Protein Energy Landscapes

With Coarse-Grained Sequence-Energy Mappings

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Chemical Engineering

by

Trent Brennen Hinkle

2017

ABSTRACT OF THE THESIS

Designing Protein Energy Landscapes

With Coarse-Grained Sequence-Energy Mappings

by

Trent Brennen Hinkle

Master of Science in Chemical Engineering

University of California, Los Angeles, 2017

Professor Tatiana Segura, Chair

We develop and implement a brand new method for protein design. Our design method utilizes advances in protein energy scoring, Markov state modeling, machine learning, distributed computing, and optimization. First we generate a Markov State Model for human ubiquitin. This creates 100 separate fixed backbone conformations of the protein. Next, we generate quick and accurate energy functions for each of the 100 separate conformations utilizing the Rosetta energy function, mean field energy modeling, and stochastic gradient descent. Finally, we use these coarse-grained sequence-energy functions to design mutant sequences with specified conformational dynamics through use of optimization algorithms. We selectively stabilize target states through maximizing Boltzmann distribution probability for either one state or a pair of states. We showcase that we can accurately design sequences with altered energy landscapes for ubiquitin mutants up to eight mutations away from wild type ubiquitin using our design approach.

The thesis of Trent Brennen Hinkle is approved.

Philip Romero

Panagiotis D. Christofides

Yi Tang

Tatiana Segura, Committee Chair

University of California, Los Angeles, 2017

Table of Contents

List of Figures

List of Tables

List of Symbols

$E(\vec{\sigma})$	Minimized Sequence Energy
$\vec{\sigma}$	Sequence
N	Sequence Position
M	Allowable Amino Acid
J_0	Intercept Parameter
J_a^i	One-Body Interaction Parameters
$J_{a,b}^{i,j}$	Two-Body Interaction Parameters
$\phi(a,i)$	Basis Function
$\xi(a,b)$	Geometric Parameter Constraint
$\vec{E}$	Energy Matrix Training Set
$\vec{J}$	Parameter Vector
$\vec{X}$	Mean Field Encoded Sequence Matrix Training Set
i	Training set number
j	SGD Parameter Vector Length
$w_j^{(i)}$	SGD Parameter Vector at i’th training set
$w_j^{(i+1)}$	SGD Parameter Vector at i+1 training set
$\gamma^{(i)}$	SGD Learning Rate
$Q\left(w_j^{(i)}\right)$	Cost Function
$y^{(i)}$	i’th Sequence Energy
$x_j^{(i)}$	i’th Encoded Sequence
p_i	Boltzmann Probability of State ‘i’
ε_i	Energy of State ‘i’
R	Boltzmann Constant
T	Temperature
$P(\varepsilon_i)$	Probability list Optimizing State ‘i’
η	Learning Rate Multiplier
$\omega(y)$	Weight Function

List of Acronyms

CE:	Cluster Expansion
MSM:	Markov State Model
MD:	Molecular Dynamics
MSA:	Multiple Sequence Alignment
BACE:	Bayesian Agglomerative Clustering Engine
CV:	Cross Validation
REU:	Rosetta Energy Unit
SGD:	Stochastic Gradient Descent
SSE:	Sum of Squared Errors
CC:	Correlation Coefficient
MAD:	Mean Absolute Deviation
PDB:	Protein Data Bank

Introduction

Proteins are molecules that are capable of performing a wide array of biological functions. The ability to logically design a protein with specified functionality would have huge impacts on science and society as a whole. One could conceivably design an enzyme to increase its ability to catalyze a reaction or potentially design a new protein with an entirely new function in a biological system. Other applications include redesigning existing proteins to change their function or specificity. Furthermore, there are many industries that could be positively influenced by protein engineering including medical, agriculture, food, energy, environment, nanotechnology, and industrial chemistry [1].

In recent years, the reliability of protein design has increased. Specific designs include the de novo designs of Top7 and Retro-Aldol Enzymes [2, 3]. However, it is important to note that these designed proteins/enzymes are usually substantially less efficient at their functions as compared with proteins and enzymes in nature [4, 5]. Therefore, establishing and implementing new computational tools to aid in the design of proteins could help to create better and more efficient proteins in the future. Aside from the De Novo design process mentioned, which is the process of designing a protein from scratch, other important and useful protein design methods include protein redesign. Protein redesign is the processes of applying mutations to an existing protein sequence in order to improve or change the protein's function [6].

Protein design is commonly posed as an inverse folding problem, where a sequence is optimized to fold to a singular pre-defined three dimensional structure [7]. The problem with this is that proteins are dynamic molecules that can go through many conformational changes over time. Many proteins also rely on these conformational changes to perform their functions [8, 9]. There are current methods that attempt to take conformational variation into consideration in the protein design process. These methods include flexible backbone protein design, multi-state design and molecular dynamics-based filtering of fixed backbone design [10-12]. Although these methods have made protein design more reliable they still fail to capture relevant sequence/conformational degrees of freedom. If one could find a way to implement relevant conformational variation into the protein design problem coupled with the ability to sample large sections of sequence space then this could lead to more accurate solutions as the model would more closely follow the dynamic nature of proteins themselves.

In this paper, we develop a new method of designing proteins that takes into consideration the vast conformational space that has been neglected in other methods. First, we must discretize a protein's conformational space by selecting states or conformations that the protein is highly likely to be in. For this step we use Markov State Models (MSM's) to accurately represent the conformational landscape of a protein. Secondly, we need a method to quickly and efficiently score the energies of a vast amount of sequences in each of the conformational states separately. Here, we use a method similar to Cluster Expansion (CE) in which we develop fast energy scoring functions using mean field theory [13]. Then we will have functions that map sequence to energy for all of our pre-defined conformational states. Thirdly, once we have a quick and reliable method to map sequence to energy we can selectively control the energy from state to state through sequence optimization. Through this process we show that we can completely and accurately redesign the conformational energy landscape of a protein. We apply this method to human ubiquitin because of its importance in protein degradation and interesting dynamic properties [14].

Background

Protein Design

Recently, protein design has become much more reliable due to advances in algorithms that attempt to find optimal low energy sequences that fold to a specified 3-D structure [15]. This approach, denoted as the inverse protein folding problem, is concerned with finding specified amino acid sequences that will be energetically stable and fold into a pre-known 3-D structure [16]. Protein design is a complex problem given the vastness of potential sequence space coupled with the dynamics of a proteins topology. In regards to sequence space, information about relevant amino acids can be learned through implementation of multiple sequence alignments (MSA) [17]. MSA's can provide sequence similarity information for protein homologs, thereby aiding in the design of stable as well as active proteins and enzymes [15]. The prevalent mutations in a MSA exist or have existed in nature and potentially will be less deleterious or even advantageous in regards to activity or stability as compared with simply a random mutation that doesn't take into consideration any knowledge from sequence space [18]. In addition to sequence space, protein dynamics also provide problems in regards to design. Such protein dynamics include side chain orientation, complete protein structural rearrangement for functionality, as well as catalytic active site consideration [5, 8, 9]. Given the high number of degrees of freedom in such a problem, protein design becomes a difficult task to complete computationally while maintaining accuracy.

Methods to design and predict protein folding of new sequences include utilization of computational packages such as RosettaDesign [15, 19]. According to Liu and Kuhlman, RosettaDesign has two main features which include a fixed backbone energy scoring function for evaluating free energy based on structure and sequence, as well as an optimization feature for finding target low energy sequences through applying site based mutations [19]. The energy function consists of several features for energy optimization and minimization. First, dead-end elimination and Monte Carlo techniques provide energy minimization of side chains in order to find optimal low energy rotamer conformations [15, 20]. Other features includes utilizing a Lennard-Jones potential which favors amino acids that are closely packed. The Lazaridis-Karplus implicit solvation model which favors polar surface residues and hydrophobic interior residues is also implemented into the energy scoring procedure [21]. Hydrogen bonding potentials, reference amino acid energy values for each residue type, and electrostatic interactions between charged residues also play a role in the energy function [19, 22]. In addition, RosettaDesign explicitly models every atom in the protein sequence being scored, including hydrogens [15]. Through all these features, RosettaDesign is able to accurately provide structural energy minimization and free energy scoring for a sequence threaded onto a fixed 3-D backbone structure. In addition to the energy scoring function, RosettaDesign also has a built in optimization procedure to find low energy sequences. Monte Carlo optimization coupled with simulated annealing is utilized for finding optimal sequences [19]. The optimization procedure searches for low energy sequences starting from a random sequence where rotamer switches or single residue mutations are applied and either accepted or rejected based on a specified criteria [19]. Previously, RosettaDesign has been used to successfully design low free energy sequences for nine different globular proteins [23]. We focus on utilizing Rosetta's fixed backbone energy scoring function as well as an MSA on ubiquitin variants for our new protein design method.

Markov State Models

Proteins are flexible and dynamic molecules that can undergo large conformational domain changes over time [24]. In order to model this conformational space one needs to discretize this infinite domain into a discrete number of states. A Markov State Model (MSM) is essentially a simplified representation of a proteins conformational landscape through geometric and kinetic clustering of states found in molecular dynamics simulations [25]. Through MSM's one can learn about and inspect the dynamics of a proteins conformational space. In Figure 1, we present a simplified picture of an MSM taken from a paper by Bowman, Beauchamp, Boxer, and Pande [25]. We utilize MSM methodology to coarse-grain the conformational space of human ubiquitin as a first step in our design method. An MSM takes a proteins infinite conformational space and discretizes this space down to a discrete number of conformational states. The states represented in an MSM consist of the most probable states a protein tends to be in and is constructed through molecular dynamics simulations [25]. According to Bowman, Beauchamp, Boxer, and Pande, the first step to building an MSM is to cluster all of the conformations that are discovered by the molecular dynamics simulations based on geometric structure utilizing a k-means or k-centers algorithm. This initially geometric clustering can create many microstates from anywhere between 10,000 and 100,000 separate states [26]. Conformations that fall within the same microstate tend to have root-mean-square-deviations of less than 2 or 3 angstroms [26, 27]. The next step of constructing the MSM is to kinetically cluster these microstates through

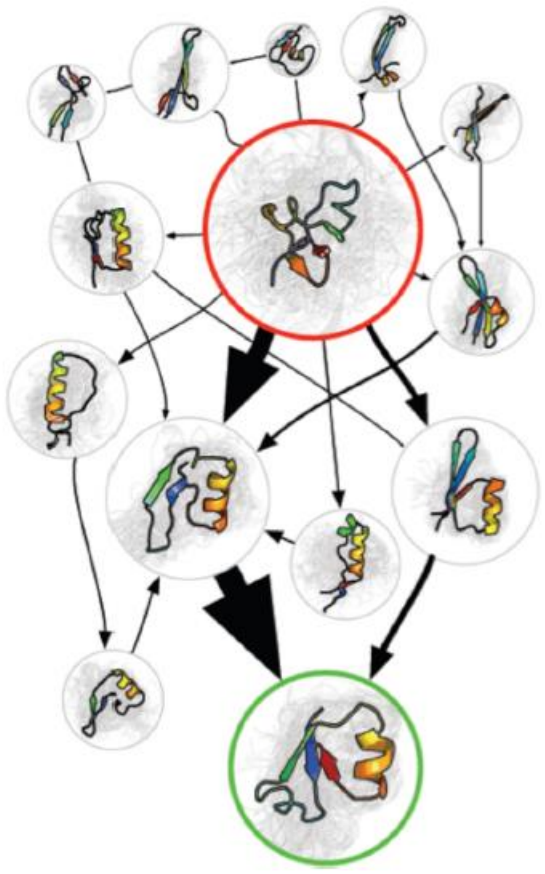


Figure 1: Markov State Model of the folding of NTL9.
The Figure shows highlights of some of the most likely conformations of NTL9

construction of a microstate transition matrix [25]. According to Bowman, Beauchamp, Boxer, and Pande; this matrix is first constructed by assigning each structure from the MD trajectory to one of the geometrically clustered microstates. Essentially, each structure from the MD trajectories will have one microstate that it is most closely related to geometrically. When this microstate is found the structure from the MD simulation is assigned to that specified microstate [25]. After this, the microstate trajectories are used to count transitions between pairs of microstates $i \ and \ j$ for a specified lag time $t \sim 10ns$. A count matrix $C_{ij}(\ )$ can then constructed by counting the number of transitions a trajectory transitions from state i at time t_0 to state j at time t_1 [25]. Next, probabilities of transitioning between microstates $i \ and \ j$ can be calculated through comparing counts of transitions from $i \ to \ j$ with counts of transitions from i and all other possible states [25]. A probability matrix $P_{ij}(\ )$ can then be constructed in this fashion. With this microstate transition matrix, microstates can then be further clustered by their kinetic similarity through setting a threshold on probability. Essentially microstate pairs that have high probabilities of transitioning between each other can be clustered kinetically. These MSM's can then be coarse-grained and simplified further by restricting the microstate transition matrix into a smaller number of states by looking at larger timescales for lag time $t \sim 100ns$. These larger timescales coarse-grain the MSM because each MD trajectory will be able to kinetically reach more microstates over the increased time thus lowering the number of kinetically independent states [25]. More complex methods of coarse-graining MSM's to fewer relevant states include the Bayesian agglomerative clustering engine (BACE) which can coarse-grain MSM's down to tens or hundreds of states [28].

Cluster Expansion

Cluster Expansion (CE), developed by Amy Keating's Lab, provides a method of quickly and accurately mapping sequence to energy [13]. We utilize a simplified version of CE to develop fast energy scoring functions. The (CE) model development involves taking a mean field approach to approximating the sequence-energy mapping [29]. The sequence-energy mapping of a protein can be represented by various energetic interactions between the different amino acids present in the protein. Different energetic interactions between different residues can be selected from an n-body system and we can approximate the entire n-body system energy with lower order interactions between residues [29].

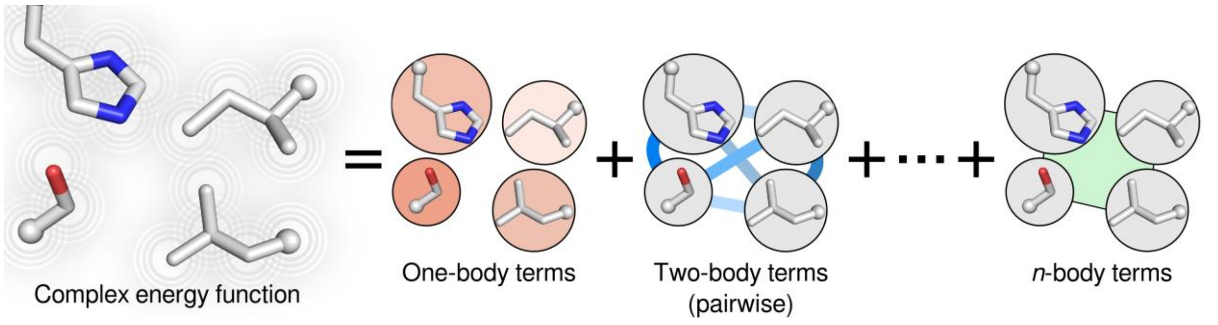


Figure 2: Mean Field Energy Decomposition
Picture representation of energy function decomposition. The energy of an n-body system can be decomposed into lower order interaction parameters. Pictured are one-body, two-body, and n-body terms. Many of the higher order terms have negligible magnitudes and systems can often be accurately approximated with lower order terms.

These interactions can stem from one body interactions to pairwise interactions between residues even up to n-body interactions between n residues. The Mean Field Model set up is briefly presented below taken from papers by Amy Keating's lab [13, 30, 31]. First, they let $E\left(\vec{\sigma}\right)$ be the minimized energy of a sequence $\vec{\sigma}$ in a specified conformation determined using a standard free energy scoring function (i.e. Rosetta). They then define a sequence $\vec{\sigma}=\left\{\sigma_1,\ldots,\sigma_N\right\}$ where σ_i is a representation of the i'th position of the sequence being modeled. Next, they let M be the number of possible amino acids at each of the N positions. Then each position σ_i of the protein can be any of the M amino acids, $\sigma_i = 0 \ldots M-1$ or the entire sequence can be represented as follows $\vec{\sigma} = \{\sigma_1 = 0 \ldots M-1, \ldots, \sigma_N = 0 \ldots M-1\}$. Each index of M then corresponds with a different allowable amino acid per position and an index of zero refers to a reference amino acid present in the wild type sequence per position. The energy of any sequence can then be represented as a series of deviations from the reference sequence. The cluster expansion then takes the following form, showing up to two-body interactions:

$$E\left(\vec{\sigma}\right) = J_0 + \sum_{a=1}^{N} \sum_{i=1}^{M-1} J_a^i \cdot \phi(a,i) + \sum_{a=1}^{N-1} \sum_{b>a}^{N} \sum_{i=1}^{M-1} \sum_{j=1}^{M-1} J_{a,b}^{i,j} \times \phi(a,i) \cdot \phi(b,j) \cdot \xi(a,b) + \ldots, \qquad (1)$$

Where $\phi(a,i)$ is the basis function of the mean field energy expansion. $\phi(a,i)$ is defined to take on a value of 1 if amino acid i is present at position a (excluding reference sequence amino acids) and takes on a value of zero otherwise. Essentially, sequences are encoded using binary indicator variables in vector space. A 1 present at a specified position in the vector indicates the presence of the specified interaction and a 0 represents the absence of an interaction. $\xi(a,b)$ is a function to decide which clusters to include in the model. Finally, the J values (mean field energy parameters) can be learned through statistical fitting algorithms such as multivariate linear regression with enough energy and sequence data. In the CE method, deciding where to truncate the expansion is arbitrary. However, deciding which J's to keep is done systematically through cross validation (CV) [13]. Essentially, if J values increase the CV score through numerical noise then they are excluded. When a J value improves fitting, the clusters are included in the final model. Models developed by Amy Keating's lab show good prediction accuracy for a coiled-coil backbone as well as a globular zinc-finger backbone [13]. We implement a simplified version of this CE method where instead of optimizing for certain J values or clusters, we simply select parameter values to include based on a geometric constraint $\xi(a,b)$.

Methods

Markov State Model Generation

The first step in this new design method is the coarse graining of conformational space to create a select number of conformations to base our models on. This was done largely through collaboration with Dr. Greg R. Bowman's lab at Washington University in St. Louis. Running of the molecular dynamics simulations as well as construction of the MSM was done through methods similar to those in the following paper from the Bowman lab [32]. Briefly, twenty 500 ns simulations were run from the starting conformation using Gromacs as well as the Amber03 force field [33]. TIP3P water was applied as the solvent for each of the simulations [34]. Next, using the steepest descent algorithm the energy of the system was minimized until maximum force values fell below $1000\, {}^{Kj}\!/_{mol\cdot \min}$ with a step size of 0.01nm. Other parameters for the simulations were identical to those ran in the Hart paper [32]. Upon initiation of the simulations, snapshots were stored every 10 ps. After simulations were completed the MSM was constructed using MSMBuilder [35]. Similarly to the simulations, the construction of the MSM was based upon methods from the same paper by Kathryn Hart and the Bowman Lab. Briefly, clustering of the data for the MSM was done through using a k-centers algorithm based on the root mean square deviation (RMSD) between all backbone heavy atoms and C beta atoms. A threshold of 1 Angstrom for RMSD was used between clusters. Next the clusters were centered to the densest parts of conformational space using a k-medoids update. Finally, the ending structures were kinetically clustered into macrostates utilizing a microstate transition matrix to produce 100 separate conformations of ubiquitin [32]. Each conformation was output as a PDB file for ease of use in applying energy functions to the structures.

Energy Scoring

To generate the data to learn the mean field energy parameters (J values) we apply random mutations to the wild type ubiquitin and we use Rosetta to score the protein energy, thus generating lists of mutated sequences and minimized energies [36, 37]. Utilizing Rosetta we start by generating the random mutant via Python followed by threading it onto one of the one-hundred selected backbone conformations. Next we minimize the system energy over side-chain rotameric states (Coarse-Grain rotamers). The structure is then relaxed using fixed-backbone descent and then the energy of the minimized configuration is recorded in Rosetta Energy Units (REU) [36]. This process is then repeated thousands of times for all 100 conformational states separately until enough data is generated to achieve accurate models.

Parameter Estimation

Statistical Modeling of the Mean Field Approximation

Upon successful generation of sequences and energies we can then use this data to train statistical models to generate our sequence-energy mappings. First, we need to select where to truncate our mean field energy model. We select to truncate our models at either single or pair level interactions. For single interactions, we include all possible amino acids at all possible positions. For pair level interactions we implement a geometric constraint $\xi(a,b)$ that takes on a value of 1 if the constraint is met or a value of zero if the constraint is not met. This geometric constraint is a way to include certain pairs of amino acids that are geometrically close together and to cut out others that are spatially farther apart. Our constraint for pair level interactions is defined as follows. We take the minimum distance between every pair of amino acids and select pairs that have a minimum distance of d or closer between each other (taking into consideration the closest atoms in each residue). Therefore, when we utilize pair interactions the geometric constraint is a way of singling out seemingly important pair interactions and neglecting less important pair interactions due to a relative distance. Next, to learn the mean field approximations or effective energy parameters we start with a simple multivariate linear regression approach [38]. We can reduce the cluster expansion model (mean field model) as follows:

$$E(\vec{\sigma}) = J_0 + \sum_{a=1}^{N}\sum_{i=1}^{M-1} J_a^i \cdot \phi(a,i) + \sum_{a=1}^{N-1}\sum_{b>a}^{N}\sum_{i=1}^{M-1}\sum_{j=1}^{M-1} J_{a,b}^{i,j} \times \phi(a,i)\cdot\phi(b,j)\cdot\xi(a,b)$$

$$E(\vec{\sigma}) = J_0 + \overrightarrow{J_a^i} \cdot \vec{\phi}(a,i) + \overrightarrow{J_{a,b}^{i,j}} \times \vec{\phi}(a,i)\cdot\vec{\phi}(b,j) \qquad (1)$$

$$\vec{E} = J_0 + \vec{J}\cdot\vec{X} \qquad (2)$$

$\vec{E}$ Is simply the training set of all energies and $\vec{X}$ is the corresponding training set of the binary encoded sequences using the CE/mean field method and $\vec{J}$ is our effective interaction parameter vector which we will try to learn through fitting. Solving the previous equation for $\vec{J}$ results in:

$$\vec{J} = \left(\overrightarrow{X^T}\cdot\vec{X}\right)^{-1}\overrightarrow{X^T}\cdot\vec{E} \qquad (3)$$

We could then use our sequence and energy data to solve for our parameter vector $\vec{J}$. However, there are various problems with this approach given the size and complexity of the problem. First, we will be implementing this using human ubiquitin which has 76 amino acids in its chain. If we allow all 20 amino acids per position and only take into consideration single body interactions then we will be looking at parameter vectors equal to a size of 1445 $\left((76*19)+1\right)$.

If we take into consideration all one body and two body terms then our parameter vector would grow to a size of 1030295. Essentially, these vectors and corresponding design matrices for our system will potentially be too large to fit into system memory as we could need upwards of 1000000 training sets to see all interactions. Furthermore, we wish to have this framework be scalable to much larger protein systems. Therefore, we need to use algorithms that can iteratively train our models so that the data is manageable for processing. With all this in mind, we select to take an online machine learning approach to learning the mean field energy approximation utilizing stochastic gradient descent (SGD). Stochastic gradient descent is a logical choice for this large-scale learning problem as we are able to iteratively train our models which avoids the memory errors associated with the large design matrices of regular multivariate linear regression [39].

Below we apply the stochastic gradient descent algorithm to our learning problem taken from several papers by Leon Bottou [39, 40]. First, they start with the simplified SGD algorithm below.

$$w_j^{(i+1)} = w_j^{(i)} - \gamma^{(i)} \nabla Q\left(w_j^{(i)}\right) \qquad (4)$$

They say that j is the size of the parameter vector and encoded binary sequence depending on which terms we include from the mean field approximation and i is the training set iteration. Then $Q\left(w_j^{(i)}\right)$ is the cost function; in this case the sum of squared errors (SSE), $\gamma^{(i)}$ is the learning rate (hyper parameter), and $w_j^{(i+1)}, w_j^{(i)}$ are our parameter vectors (with length j) at training set $i+1, i$. Now we define the cost function as the sum of squared errors (SSE) as follows [41].

$$Q\left(w_j^{(i)}\right) = \frac{1}{2}\sum_{i=0}^{m}\left(y^{(i)} - h_w\left(x\right)^{(i)}\right)^2 \qquad (5)$$

$$h_w(x) = \sum_{i=0}^{m}\left(w_j^{(i)} x_j^{(i)}\right) = w^T x$$

Next we take the gradient of $Q\left(w_j^{(i)}\right)$:

$$\frac{\partial Q}{\partial w_j} = \frac{1}{2}\frac{\partial}{\partial w_j}\sum_{i=0}^{m}\left(y^{(i)} - h_w(x)^{(i)}\right)^2$$
$$\frac{\partial Q}{\partial w_j} = 2*\frac{1}{2}\sum_{i=0}^{m}\left(y^{(i)} - h_w(x)^{(i)}\right)\frac{\partial}{\partial w_j}\left(y^{(i)} - h_w(x)^{(i)}\right)$$
$$\frac{\partial Q}{\partial w_j} = \sum_{i=0}^{m}\left(y^{(i)} - h_w(x)^{(i)}\right)\frac{\partial}{\partial w_j}\left(y^{(i)} - \sum_{i=0}^{m}\left(h_w(x)^{(i)}\right)\right)$$
$$\frac{\partial Q}{\partial w_j} = \sum_{i=0}^{m}\left(y^{(i)} - h_w(x)^{(i)}\right)\frac{\partial}{\partial w_j}\left(y^{(i)} - \sum_{i=0}^{m}\left(w_j^{(i)^T} x_j^{(i)}\right)\right)$$
$$\frac{\partial Q}{\partial w_j} = \sum_{i=0}^{m}\left(y^{(i)} - h_w(x)^{(i)}\right)\left(-x_j^{(i)}\right)$$
$$\frac{\partial Q}{\partial w_j} = \sum_{i=0}^{m}\left(h_w(x)^{(i)} - y^{(i)}\right)\left(x_j^{(i)}\right)$$

Now substitute the gradient into equation (4).

$$w_j^{(i+1)} = w_j^{(i)} - \gamma^{(i)}\sum_{i=0}^{m}\left(\sum_{i=0}^{m}\left(w_j^{(i)} x_j^{(i)}\right) - y^{(i)}\right)\left(x_j^{(i)}\right) \qquad (6)$$

Equation (6) can be further simplified by taking the gradient with one training set at a time, essentially updating $w_j^{(i+1)}$ one iteration of i at a time. Equation (6) simplifies to the following:

$$w_j^{(i+1)} = w_j^{(i)} - \gamma^{(i)}\left(\left(w_j^{(i)^T} x_j^{(i)}\right) - y^{(i)}\right)\left(x_j^{(i)}\right) \qquad (7)$$

Equation (7) is the final form of our learning algorithm where $x_j^{(i)}$ is a random encoded sequence, $y^{(i)}$ is the corresponding minimized energy of sequence (i) and $w_j^{(i)}$ is our effective energy parameter vector at learning step (i). The stochastic gradient descent algorithm can then be implemented as shown below in Figure 3:

```
$Let\ \ w_j^{(0)} \in \mathbb{R}^j$
$define\ \ w_j^{(0)}\ \ as\ the\ zero\ vector\ in\ \mathbb{R}^j$
$w_j^{(0)} = 0_j = [0, 0, ...0] \in \mathbb{R}^j$
$Loop:$
$For\ \ i = 0\ to\ m$
    $update: \quad w_j^{(i+1)} = w_j^{(i)} - \gamma^{(i)}\left(\left(w_j^{(i)^T} x_j^{(i)}\right) - y^{(i)}\right)\left(x_j^{(i)}\right)$
```

Figure 3: Loop Procedure for Model Training
For model training we set the initial parameter vector to the zero vector in j space. We then run a loop from i=0 to m (where m is the maximum allowable number of training sets) updating the SGD algorithm with one encoded sequence and its corresponding minimized energy at a time.

For computational implementation of the SGD model we select to write our own SGD algorithm using Python [42, 43] with various modules including NumPy for our calculations [44]. We used NumPy to do matrix calculations for updating the model and also used the modules built in correlation coefficient (CC) and mean absolute deviation (MAD) calculations for model validation. We also used Matplotlib in generation of all relevant plots for our model results [45]. Given the design of our framework we can apply this algorithm using each of the 100 different conformations separately. We apply the different conformational state during the free energy scoring aspect before model training. This results in 100 different effective energetic parameter vectors corresponding with each of the 100 conformational states after model training is finished.

Online Learning Approach for Model Training

To train each of our 100 energy models for each conformational state we choose to take an online learning approach. We select this approach out of necessity for multiple reasons. First, we do not readily know how much data a model would need to generate an accurate and converged result. Secondly, during initial model training we ran into memory errors with the linear regression design matrix. Due to these factors an online learning approach makes sense as it makes the computation of our models possible and more efficient. To implement our SGD algorithms we take the following approach. Given that our approach treats each conformation independently for model training, we are able to distribute our algorithms across high-performance computer clusters. We utilize a computer cluster through the University of Wisconsin-Madison called HTCondor (High-Throughput Computing) [46]. Through HTCondor we are able to distribute all 100 of our algorithms to different nodes. Ensuring that we parallelize our algorithms we can then train all 100 of our energy models corresponding with all 100 individual conformational states simultaneously. We are able to update our models and also check model accuracy in real time. When a specific threshold for mean absolute deviation (MAD) is reached we can tell our algorithms to stop running and return our learned energetic parameter vectors in the form of line delimited text files. For Rosetta energy trained models we select a MAD value of 2 REU for the threshold error value (stopping criteria). This is a great aspect and benefit of our approach as we will not over-sample data so long as models train to an error of lower than 2 REU.

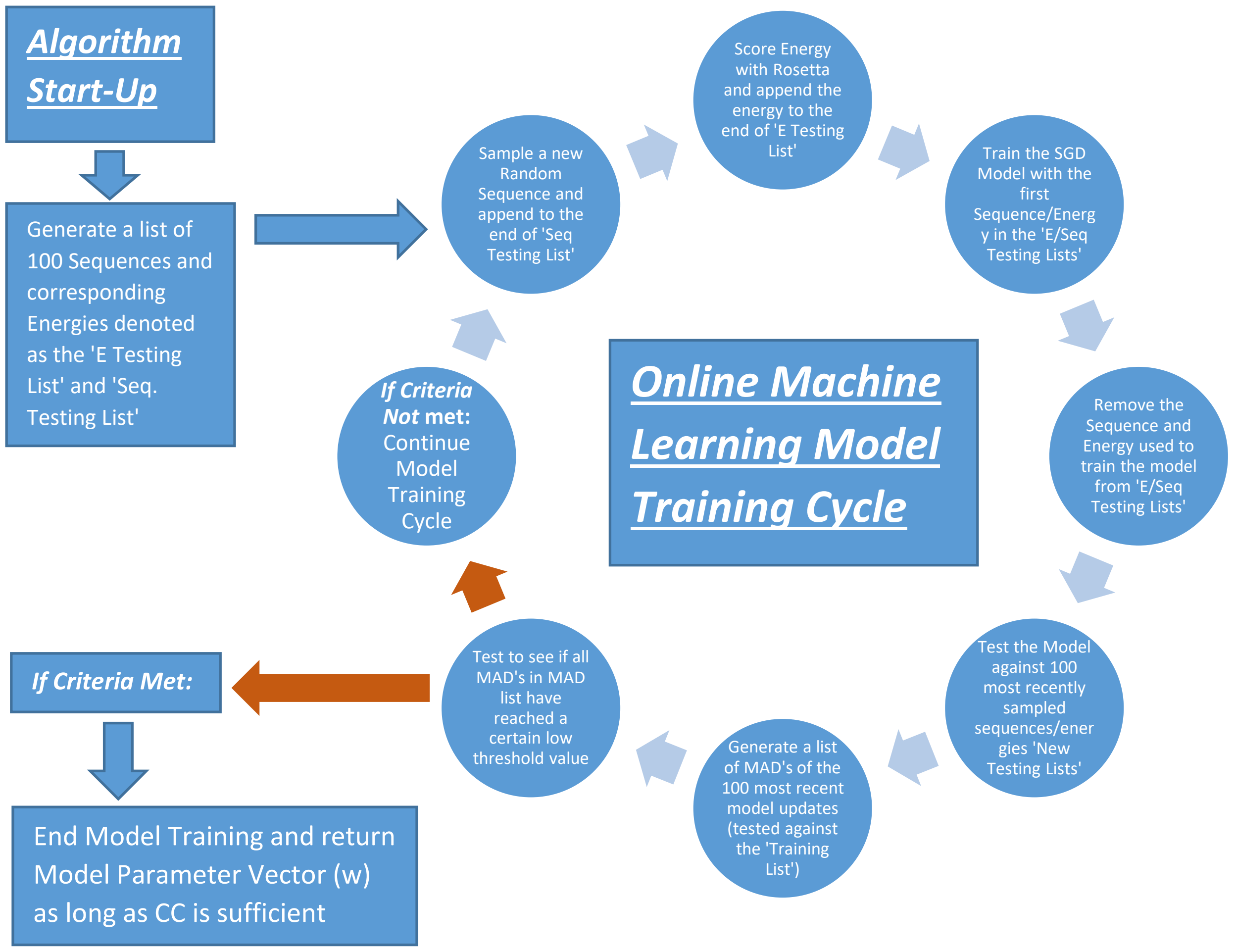

Algorithm Start-Up
Generate a list of 100 Sequences and corresponding Energies denoted as the 'E Testing List' and 'Seq. Testing List'
Sample a new Random Sequence and append to the end of 'Seq Testing List'
Score Energy with Rosetta and append the energy to the end of 'E Testing List'
Train the SGD Model with the first Sequence/Energy in the 'E/Seq Testing Lists'
Remove the Sequence and Energy used to train the model from 'E/Seq Testing Lists'
Test the Model against 100 most recently sampled sequences/energies 'New Testing Lists'
Generate a list of MAD's of the 100 most recent model updates (tested against the 'Training List')
Test to see if all MAD's in MAD list have reached a certain low threshold value
If Criteria Not met: Continue Model Training Cycle
Online Machine Learning Model Training Cycle
If Criteria Met:
End Model Training and return Model Parameter Vector (w) as long as CC is sufficient

Figure 4: Online Machine Learning Model Flow Chart
The training cycle starts by generating a list of 100 sequences and another list of 100 corresponding energies denoted as the 'Testing Lists' in Python. After the lists are 100 in length a new sequence and energy are sampled and appended to the end of the lists and the earliest sequence and energy sampled in the past is used to update the model. Both the sequence and energy are then removed from their respective lists once they are used for model training. The 100 most recently sampled sequences/energies are then used to test the model. With this method, we ensure that sequences/energies are always used for testing the model before training the model. Once a sequence/energy combination is used to train the model the pair is then discarded and is no longer under consideration for model accuracy testing. Correlation coefficient values (CC) and mean absolute deviation values (MAD) are calculated at each iteration with all 100 sets of sequences/energies in the testing lists. This model training process continues until a predefined low threshold for MAD is met for a certain number of consecutive iterations or until a specified time has elapsed. After this criteria is met for each conformational model individually the algorithms stop and HTCondor returns our energy parameter vectors in the form of text files.

Figure 4 is a picture representation of the approach we take to implementing our algorithms. This approach ensures that we do not over-sample data. HTCondor also limits the time our algorithms are allowed to run to 72 hours. Therefore, if our models do not hit the MAD threshold because they are no longer improving or have not been trained with enough data our parameter vectors will be returned at this point. Models can also be resubmitted for training if needed with the starting point vector being the last learned parameter vector as opposed to the zero vector in J space.

We selected to take this online learning approach not only out of necessity but also because it makes our method scalable to much larger model systems. This method could definitely be expanded to proteins much larger in length so long as accurate models can still be generated. This is because larger protein systems would need more parameters and likely more training sets to generate accurate models. In our approach using stochastic gradient descent we iteratively train our models with one training set at a time. Therefore, we do not run into any memory errors associated with the linear regression design matrix. This inherently makes our method scalable to proteins larger in size. The main issue expanding to larger protein systems would be to make the SGD models accurate enough for optimization purposes. Next, in regards to conformational space we only took into consideration 100 separate conformations while this method is easily scalable to a thousand or even ten thousand separate conformations depending on the service used for algorithm distribution. Essentially, a much larger conformational space can be optimized over if desired. Furthermore, when using a larger number of conformations it is not as important to get every single conformational model to be accurate. Those models that are inaccurate can simply be thrown out if the conformation is not deemed important. Essentially, as long as one can generate models that are accurate enough for optimization purposes then this method can easily be scaled up in terms of sequence length/space as well as conformational space.

Sequence Design

Reduction of Sequence Space

Before designing sequences we wish to limit the number of possible random mutations per sequence we score for our trained models. There are several reasons to setting this limit. First, if we allow mutations at all possible positions for all possible amino acids then that would be 76^20 possible sequences. This is such a vast space and the majority of these sequences with high numbers of mutations from the wild type sequence will likely not be optimal for our design objectives. The reason for this is that as we apply more mutations to the wild type ubiquitin it is likely to raise the energy score of the sequence and we are interested in low energy sequences especially for design purposes. Preliminary testing has shown that allowing mutations at all positions generates sequences with high energies relative to sequences with low numbers of allowable mutations. For our design purposes we are interested in our models being extremely accurate at low energies. Much of the high mutation - high energy sequence space is not important to us and can essentially be ignored. Thus, training our models using sequences with a low number of mutations is ideal for our design objectives while still covering a large portion of optimal sequences. For our sequence generation we allow wild type ubiquitin to be mutated at four random positions. We then select to further reduce sequence space by running a multiple sequence alignment of over 1000 ubiquitin variants and then selecting the top 4 most observed amino acids per position. The multiple sequence alignment was run using the MUSCLE algorithm [47] and ubiquitin variants were found using the NCBI Protein Database [48].

1 MILF	11 KREN	21 DENY	31 QEAH	41 QHRL	51 EDAK	61 IVLT	71 LIVF
2 QHKR	12 TESI	22 TSNR	32 DEAG	42 RHCK	52 DNEG	62 QERS	72 RHAK
3 IVLT	13 IVLF	23 IVTL	33 KREQ	43 LIMF	53 GDSE	63 KNGR	73 LVFI
4 FYKS	14 TAEI	24 EDAK	34 EKVD	44 IVLF	54 RKCH	64 EGDK	74 RCKQ
5 VIAL	15 LIVF	25 NDST	35 GESR	45 FYLI	55 TSIA	65 SAND	75 GSCD

6 KREN	16 EDKN	26 VILM	36 IVTL	46 AGSV	56 LVIA	66 TVIS	76 GSAN
7 TSNM	17 VIAT	27 KREQ	37 PSLA	47 GSRD	57 SAKG	67 LIVM	
8 LFVP	18 EDKN	28 AQSE	38 PSVL	48 KREQ	58 DEYN	68 HYQR	
9 TSAM	19 SPAT	29 KRQM	39 DENV	49 QELR	59 YCHF	69 LMVI	
10 GSER	20 STNG	30 IVLT	40 QHEL	50 LMVF	60 NSDG	70 VLIA	

Table 1: Multiple Sequence Alignment for Ubiquitin Variants
Presented in the table is each allowable amino acid (represented by its singular letter) at each possible position. This is the sequence space we move forward with for all relevant model training.

Table 1 is a representation of each allowable amino acid at each position in the ubiquitin mutants we wish to take into consideration for model training. This is the final sequence space we set for training our models, coupled with four random mutations of the above sequence space at four random positions per sequence.

Design Objective: Boltzmann Distribution Optimization

Upon accurate generation of all 100 sequence-energy models for all 100 conformations we can then design sequences with specified conformational dynamics. We start with the simplest design objective of stabilizing one of the one-hundred conformational states. We do this through maximizing the Boltzmann Distribution for a selected conformational state. The Boltzmann Distribution (presented below) gives the probability that our protein system will be in the specified state based on energy and temperature [49].

$$p_i = \frac{e^{-\varepsilon_i/RT}}{\sum_{j=1}^{M} e^{-\varepsilon_j/RT}} \qquad (8)$$

p_i is the probability of state i existing, ε_i is the energy of conformational state i, and RT is the Boltzmann Constant multiplied by system temperature. For our designs we select to use $R = 1.99 * 10^{-3}\, kcal/K \cdot mol$ and $T = 298.15 * 2$. Furthermore, the denominator of the function is a sum of the energies of all states while the numerator only takes into consideration state i. Based on the distribution a higher probability corresponds with a lower ε_i. This maximization of the Boltzmann Distribution then leads to a theoretically more stable conformational state i.

```
$Set\ T\ and\ R$
$Set\ initial\ Sequence\ as\ Wild\ Type\ Sequence$
$Calculate\ p_i\ of\ Wild\ Type$
$Loop:$
$for\ n = 0\ to\ N \sim 1000000$
    $Apply\ 1\ mutation\ to\ current\ Sequence$
    $Score\ Energies\ of\ all\ States\ with\ all\ 100\ Mean\ field\ Energy\ Approximations$
    $Update\ p_i$
    $if:$
        $p_i\ increases,\ accept\ mutation\ and\ continue\ loop\ with\ current\ Sequence,$
        $save\ Sequence / p_i\ data\ in\ list\ P(\varepsilon_i)\ restart\ with\ Wild\ Type$
        $when\ set\ number\ of\ mutations\ is\ reached$
    $if:$
        $p_i\ decreases,\ decline\ mutation\ and\ continue\ loop\ with\ previous\ Sequence$
$Take\ highest\ overall\ p_i\ as\ optimized\ sequence\ as\ follows:$
$\arg\max_{p_i \in P(\varepsilon_i)} P(\varepsilon_i)$
```

Figure 5: Sequence Optimization Routine
Algorithm set-up and loop procedure for searching through sequence space and maximizing probability p_i for state 'i'.

In this procedure we essentially walk through sequence space with one mutation at a time starting at the wild type sequence. If a mutation increases p_i we accept this mutation and continue applying mutations/calculating probabilities until a set number of mutations from the wild type sequence is met. If a mutation decreases p_i we decline this mutation and revert back to the previous mutation and continue the loop. We only care for mutations that increase probability of the chosen state so those are the only mutations we accept. Once the mutation threshold is met we save the probability and sequence data and we restart at the wild type sequence again continuing the process. At the end of the loop we take the maximum p_i value over all iterations and select the corresponding sequence as the optimized sequence for the specified state i for the overall system. This process was repeated for each state i once and the accuracy of probability and energy (using our mean field approximations) for all designed sequences was tested against the probability and energy values calculated with Rosetta energy values. Before optimization through the use of the Boltzmann Distribution we first had to apply a conversion factor to the REU values to transform them into real energy units. We use the following conversion factor taken from a paper by Kellogg et al, $y = 0.57x$. Where y is the new converted energy in ${}^{kcal}\!/_{mol}$ and x is an REU energy value [50]. They use an experimental approach to generating a linear fitting converting REU to ${}^{kcal}\!/_{mol}$. Given that we do not have experimental energies for ubiquitin mutants we select to use their conversion factor for simplicity. After this conversion our energy values are in the correct units and we can begin to start sequence optimization. This design optimization was written and implemented using Python. Sequence-Energy models (text files) were imported into Python and turned into NumPy arrays and put into Python Dictionaries for ease of calling and distinguishing between different models. After sequences were designed based on stabilizing a specific state the probability values generated from our energy approximation models were then tested against probabilities from true Rosetta energy values. Percent error between the approximated and true probabilities were calculated in order to determine relative accuracy of our designs.

Design Objective: Two State Stabilization

Another design objective we pursued was to simultaneously stabilize two states. This was accomplished through maximizing the probability for two states at the same time. We do this through minimization of the following objective function:

$$\underset{p_i,\, p_k \in (0,1)}{\arg\min}\left[(p_i - .5)^2 + (p_k - .5)^2 \right] \quad (9)$$

Where p_i is the probability of state i, p_k is the probability of state k and $i \neq k$. Here we try to force each probability towards a value of .5 through minimizing the sum of both probabilities. As each probability approaches .5 the objective function value will approach zero. Therefore, through minimizing the two-state objective function we can force each probability toward a value of .5. We use a similar algorithmic approach as was presented in the single state stabilization (Figure 5). We start with the wild type sequence and apply 1 mutation. If p_i and p_k both increase then we accept the mutation and continue with another mutation. If either p_i or p_k decrease or if both probabilities decrease then we do not accept the mutation and we apply a new mutation to the previous sequence. We again do this until we reach a certain threshold for the number of allowable mutations. When this is reached we save the overall probability values and objective function values and restart the process at the wild type sequence. When the loop has finished we take the sequence with the smallest objective function value as the optimized sequence for the two state stabilization. Accuracy of two state stabilization is also assessed through comparisons with Rosetta energy values and corresponding probabilities for the optimized states/sequences similarly to accuracy testing of probabilities in single state optimization.

Energy Landscape

Upon successful designs of sequences with specified conformational stability we wish to test the validity and accuracy of our designs. We do this through comparisons of conformational energy landscapes. After we design a sequence we will have completely altered the energy landscape of the mutated protein away from the wild type sequence energy landscape. We create plots of the energy landscapes of our designed sequences using our energy models and overlay them with energy landscape plots using energies of the ubiquitin mutants generated with the Rosetta energy function. Through this comparison we can check accuracy of our energy models to see if they remain accurate across all states for designed sequences.

Results

Protein Design using Coarse-Grained Energy Landscapes

We developed and implemented a brand new method for protein design. Our approach incorporates conformational variation and dynamics directly into the design process and is also able to handle large sequences spaces, both of which tend to be neglected in other design methods. Our method utilizes advances in protein sequence-structure energy scoring, protein conformational space modeling, machine learning, distributed computing, and optimization. The first step in our design process involves discretization of the conformational landscape of human ubiquitin. Conformational space is discretized and coarse-grained through implementing a Markov State Model (MSM). The MSM is essentially a simplified representation of the conformational landscape of the chosen protein and is constructed through all-atom molecular dynamics simulations. From the MSM we end up with a specified number of highly probable conformational structures for the specified protein. Secondly, we develop fast sequence-energy mappings for each of the fixed backbone structures from the MSM. To do this we implement methods similar to Cluster Expansion (CE) which involves training linear regression models based on mean field theory using sequence and energy data [13]. In this step we use Rosetta for fixed backbone energy scoring of ubiquitin mutants for each conformational variant from the MSM. We take an online learning approach to our model training using stochastic gradient descent. These SGD models are trained with the sequence and energy data generated from

Rosetta. Each model for each conformational variation is trained simultaneously through distribution over a high performance computer cluster (HTCondor) [46]. We end up with a fast and accurate energy function in vector form for each conformational variation from the MSM. Finally, we applied optimization algorithms in an attempt to stabilize a specific conformational state or multiple conformational states across the coarse grained conformational energy landscape of ubiquitin. We do this through maximizing the Boltzmann Distribution for a singular state or pair of states while applying single site mutations to the wild type ubiquitin sequence until a certain threshold number of mutations is hit. Essentially, through this optimization process we increase the probability of our selected state(s) existing relative to all other states by applying single site mutations and calculating energy and probability at each mutation. We found that our method can accurately stabilize a state or pair of states even with minimal searching through sequence space. Furthermore, given the stability and accuracy of our mean field energy functions our method can be expanded to much more complex design objectives. Much larger regions of sequence space could potentially be searched using branch and bound optimization techniques. Ultimately, with our new approach we can completely and accurately redesign the energy landscape of human ubiquitin.

Mean Field Approximation of the Sequence-Energy Landscape

Learning the Mean Field Approximation

We are able to learn the sequence-energy landscape through applying statistical learning algorithms to our mean field models. Before applying statistical algorithms to our mean field model we must truncate the mean field model at a certain ‘n’ number of n-body interactions. We truncate our models at either one-body interactions or at a specified number of two-body interactions given a geometric constraint. The statistical algorithm we implement is stochastic gradient descent with the sum of squared errors cost function. For model training we write python scripts that first take the wild type sequence and applies four random mutations at four random positions in the sequence. The mutations allowed per position were based on the sequence alignment that we ran on the 1000 ubiquitin variants. Upon generating a random mutation the energy is scored with Rosetta. This sequence/energy pair was then used to update the SGD model (also implemented and written in Python). Once models have converged (MAD levels off and no longer increases) then we output our mean field approximation parameter vectors as a text file. This text file of parameters is our sequence-energy approximation and is uploaded as a vector into Python for fast approximation of energies of random ubiquitin mutants during sequence optimization.

Developing Accurate Sequence-Energy Models

Generating Accurate Sequence-Energy models was mostly done through trying to minimize the mean absolute deviation (MAD) of our models through applying different variations to the SGD algorithm and through implementing restrictions on sequence space. We chose to optimize our models based solely on the State 0 conformation and then applied the final model parameters to all 100 conformational states. To start we ended up restricting sequence space through the implementation of a sequence alignment on ubiquitin variants using the MUSCLE algorithm [47]. This restricted sequence space to only four possible amino acids per position as opposed to all twenty amino acids per position. This change in our model approach lead to lower overall energy scores for mutations because most of these amino acids at each of the 76 positions are likely to exist in nature. Below we showcase two separately trained models: one of the models uses sequences without restricted sequence space from the sequence alignment (all 20 AA's allowed per position) while the other model restricts sequence space to the top 4 amino acids per position based on the sequence alignment.

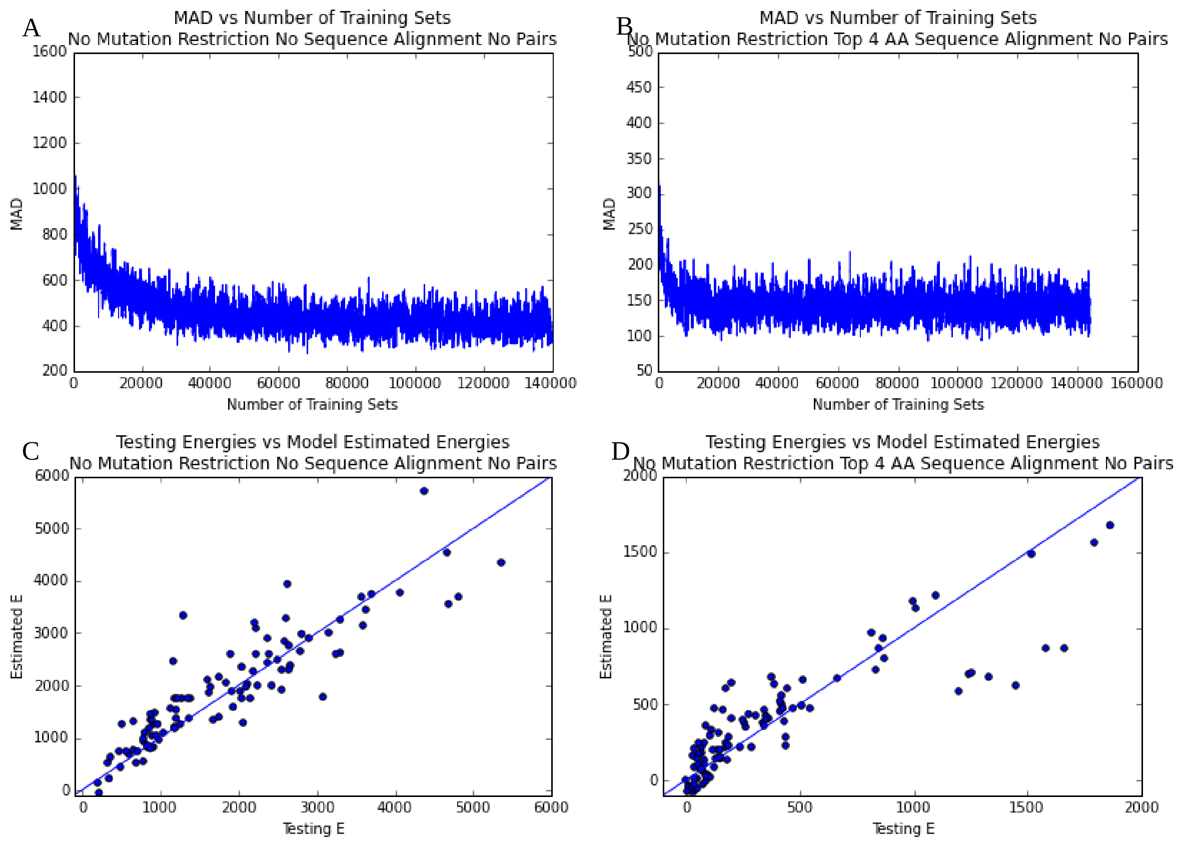


Figure 6: Sequence Alignment Model Comparison
(A) MAD vs Number of Training Sets for a model with 76 mutations per sequence generated and 20 amino acids per position with only single interaction parameters. (B) MAD vs Number of Training Sets for a model with 76 mutations per sequence generated and 4 amino acids per position (from sequence alignment) with only single interaction parameters. (C) Model Estimated Energies vs Rosetta Energies for a model with 76 mutations per sequence generated and 20 amino acids per position with only single interaction parameters. (D) Model Estimated Energies vs Rosetta Energies for a model with 76 mutations per sequence generated and 4 amino acids per position (from sequence alignment) with only single interaction parameters.

In Figure 6-A and Figure 6-C we see a model trained on sequences and energies with no sequence alignment while in Figure 6-B and Figure 6-D we see a model trained on sequences and energies with the top 4 amino acids per position from the sequence alignment. At first observation we can already see the reduction in energy when comparing the axis between Figure

6-C and Figure 6-D. The model with no multiple sequence alignment has energies that seem to range from around 0 to 6000 REU while the model with the multiple sequence alignment has energies that range from 0 to 2000 REU. This makes sense as the amino acids from the multiple sequence alignment are those amino acids that exist readily in nature in ubiquitin variants. Therefore, we see lower overall energies in sequences that utilize the multiple sequence alignment. This is exactly what we are interested in as we care most about being accurate and modeling lower energy sequences for optimization purposes. Aside from the lower energies generated through the multiple sequence alignment we also see that the model that utilizes the multiple sequence alignment mutations also performs better compared to the model that can have any amino acid at any position.

Multiple Sequence Alignment	Avg MAD last 1000 Values	Avg CC last 1000 Values	Number of Model Parameters
No, All 20 AA per position	384.39851	0.88	1445
Yes, Top 4 AA per position	137.79078	0.88	229

Table 2: Multiple Sequence Alignment Model Comparison
'No, All 20 AA per position' row corresponds with Figure 6-A and Figure 6-C while 'Yes, Top 4 AA per position' corresponds with Figure 6-B and 6-D. Applying the sequence alignment seems to improve model accuracy by over double.

As we can see from Table 2, restricting sequence space using the sequence alignment improves model accuracy from a MAD of 384 to a MAD of 137. This is a drastic improvement in model accuracy which can likely be attributed to the smaller energy range associated with sequences that are restricted by the multiple sequence alignment.

After the implementation of the multiple sequence alignment we saw a large improvement in model accuracy. However, we thought we needed another method to improve the accuracy of our models. We selected to further reduce sequence space by only allowing 4 possible mutations for each sequence we generated (quadruple mutants) as opposed to mutating every single amino acid in the protein. We saw that having less deviations from the wild type sequence lead to lower overall energy scores for the mutated sequences. Therefore, we should expect these mutation restricted data sets to train more accurate models. This restriction of sequence space was a very important part of our model training as without it MAD values for our models were far too large to run optimization on. Although this may seem like a drastic restriction in sequence space, we still feel that we cover a large and reasonable amount of the space. Most importantly we wanted our models to perform well on sequences with low energies as that is what we will ultimately optimize for and we believe training the models with these restrictions helps accomplish that goal.

Upon restricting sequence space, we next wanted to determine the amount of parameters we should be including in our models. First, we included all single interactions and gradually included more and more pair interactions given a threshold distance between amino acids. Surprisingly, after restricting sequence space, pair interactions did not drastically effect the accuracy of our models. After seeing this, we were interested in why pairs were not so important for our models. We selected to investigate this through generating different data sets while gradually increasing the number of possible mutations in the protein per data set. We selected to generate data sets with 10, 20, 35, 50, 65, and 76 mutations per sequence. We then trained models for each of these data sets first starting with only single interactions (229 model parameters) and then including pairs of amino acids that were within 8 angstroms of each other as well as single interactions (8212 model parameters for state 0). Here we wanted to see if pairs mattered at all in our models and if they did, then when they begin to matter and when they stop influencing the models as much. We generated the following table of data and corresponding plots to test to see if and where pairs matter when altering the number of mutations. When changing and altering sequence space as presented in the table below we also had to alter our learning rate $\gamma^{(i)}$ (stochastic gradient descent hyper parameter) to generate optimal models. To ensure convergence of our algorithm, literature on SGD suggests we use a decaying learning rate $\gamma^{(i)} \sim i^{-1}$ [51]. Below is the learning rate we selected for model training.

$$Learning\ \ Rate = \gamma^{(i)} = {\eta}/{(i-100)^{0.5}} \qquad (10)$$

η is the scalar we optimized through trial and error and i is the training set index. We had to keep altering this scalar until we reached a minimum MAD value for the specific data set. The type of data set influences what the optimal learning rate multiplier should be and for our quadruple mutant model we found the optimum scalar multiplier to be 2.

Multiple Sequence alignment	Number of Mutations per Sequence	Pair Distance	AVG MAD last 1000 Values	AVG CC last 1000 Values	Weight	η	Number of Model Parameters
Top 4 AA	76 Mutations	No Pairs	137.79078	0.88	None	.1	229
Top 4 AA	76 Mutations	8 Angstroms	86.931540	0.96	None	.1	8212
Top 4 AA	65 Mutations	No Pairs	122.80745	0.90	None	.1	229
Top 4 AA	65 Mutations	8 Angstroms	101.01928	0.95	None	.1	8212
Top 4 AA	50 Mutations	No Pairs	109.31297	0.89	None	.1	229
Top 4 AA	50 Mutations	8 Angstroms	80.502496	0.96	None	.1	8212
Top 4 AA	35 Mutations	No Pairs	75.636257	0.90	None	.1	229
Top 4 AA	35 Mutations	8 Angstroms	60.950644	0.96	None	.1	8212
Top 4 AA	20 Mutations	No Pairs	37.424889	0.94	None	.1	229
Top 4 AA	20 Mutations	8 Angstroms	38.378149	0.95	None	.1	8212
Top 4 AA	10 Mutations	No Pairs	14.842482	0.97	None	1	229
Top 4 AA	10 Mutations	8 Angstroms	14.728381	0.99	None	1	8212
Top 4 AA	4 Mutations	No Pairs	2.8544475	0.99	None	2	229
Top 4 AA	4 Mutations	8 Angstroms	2.9023789	0.99	None	2	8212

Table 3: Model Accuracy of Pairs vs No Pairs for different Numbers of Mutations per Sequence
This table represents model accuracy of models trained on specified datasets with limited numbers of mutations per position. We see that as we decrease the number of mutations allowed per sequence that our MAD values decrease and our CC values increase. We also see that after 20 Mutations per position pairs no longer influence our models.

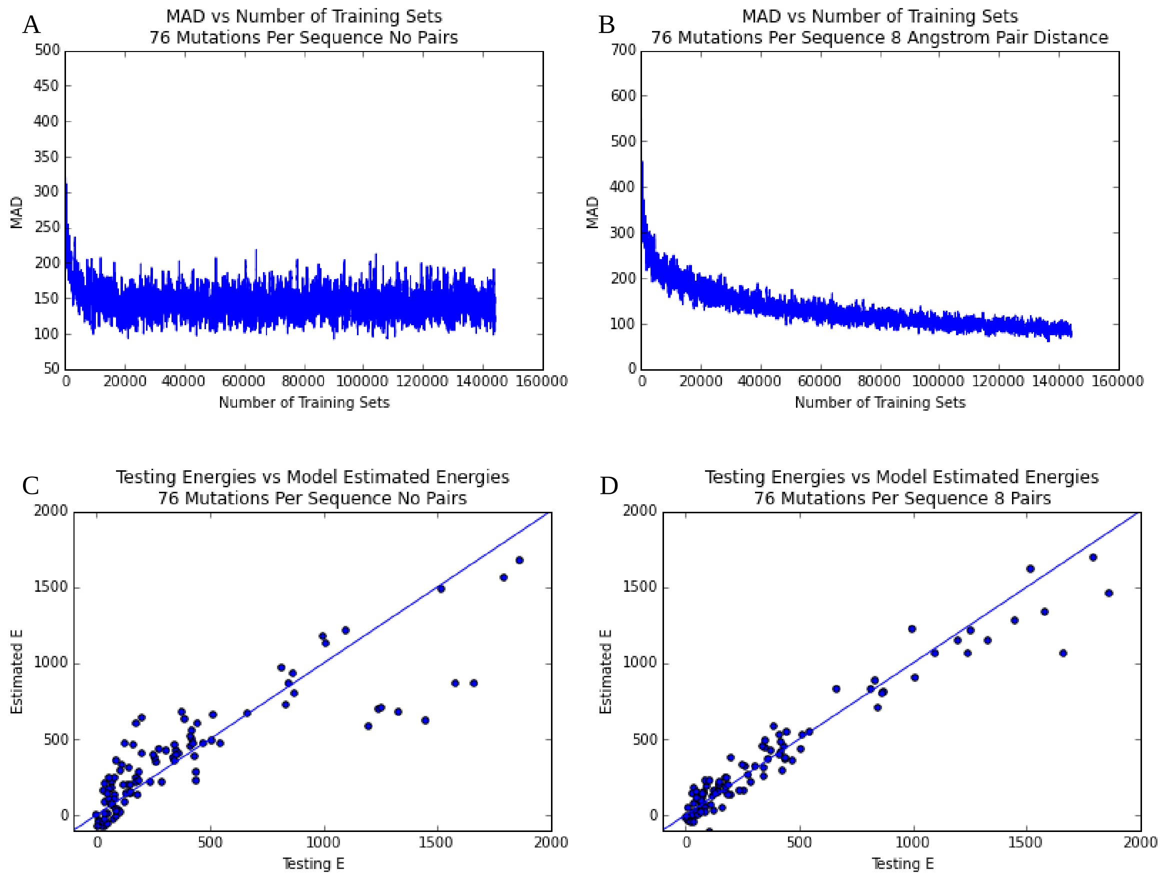


Figure 7: 76 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 76 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 76 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 76 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 76 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that pairs seem to make the model more accurate at this mutation threshold and (B) seems to not have even leveled out yet. With more training data it looks that it will reach an even lower MAD value.

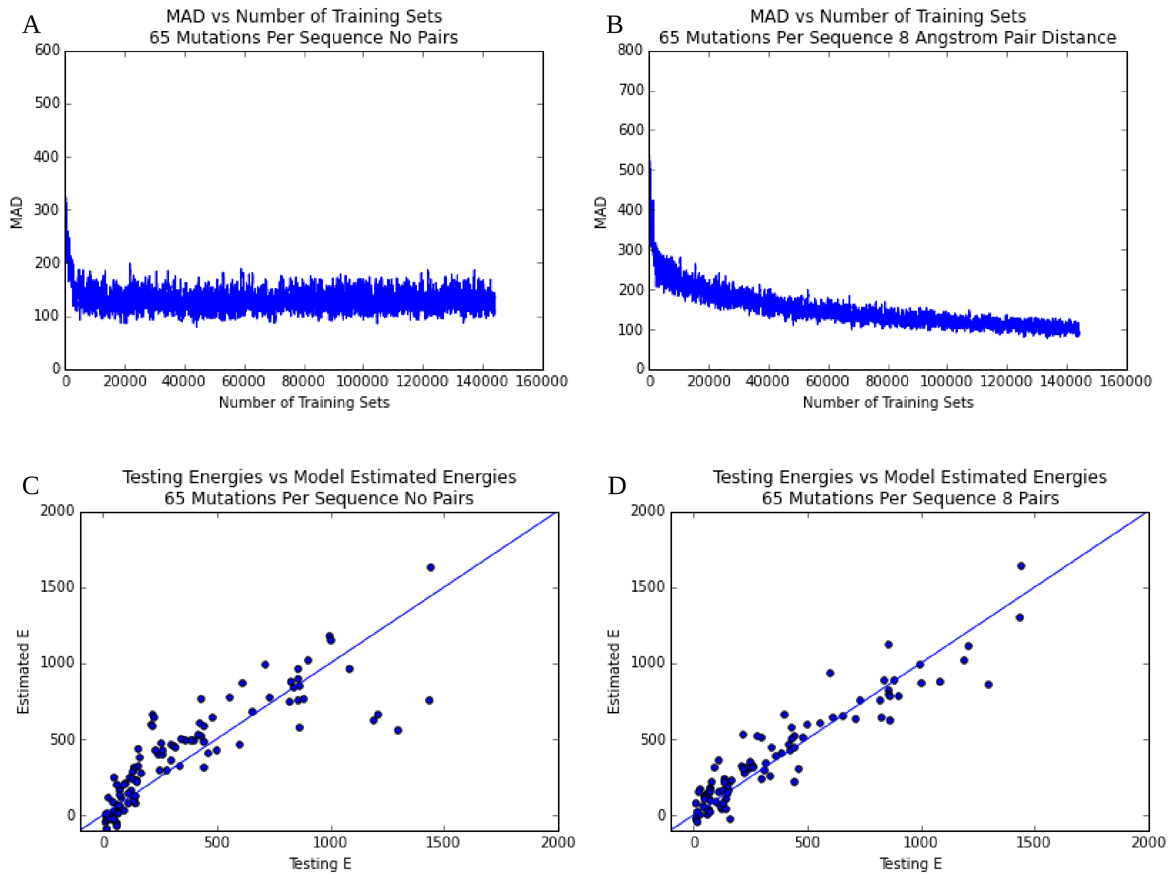


Figure 8: 65 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 65 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 65 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 65 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 65 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. In (B) we see that MAD has not leveled off and the model will likely improve further with more training data. (D) Also seems to have more linear agreement as compared with (C).

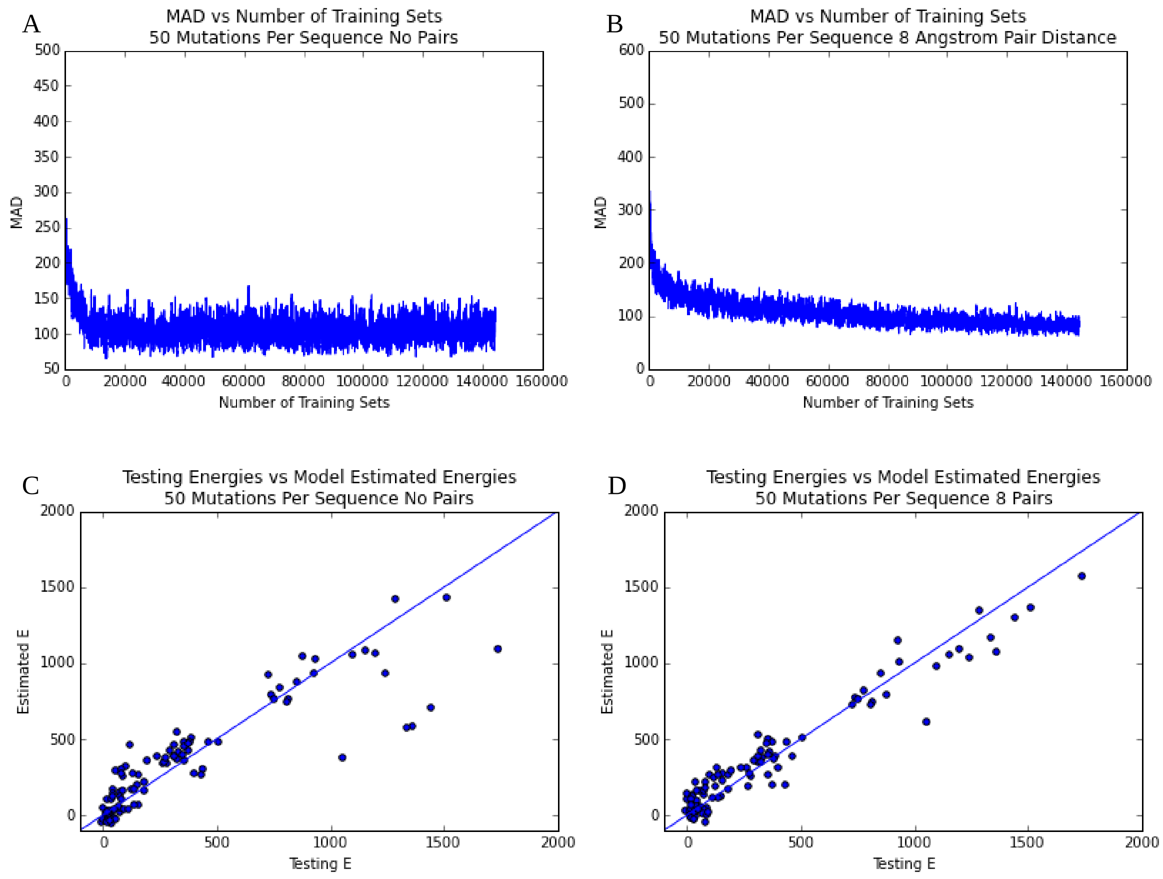


Figure 9: 50 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 50 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 50 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 50 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 50 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that (B) has not leveled off and will likely still improve with more training sets. Also, (D) seems to have better linear agreement than (C). At this mutation threshold pairs seem to still have some influence on model accuracy.

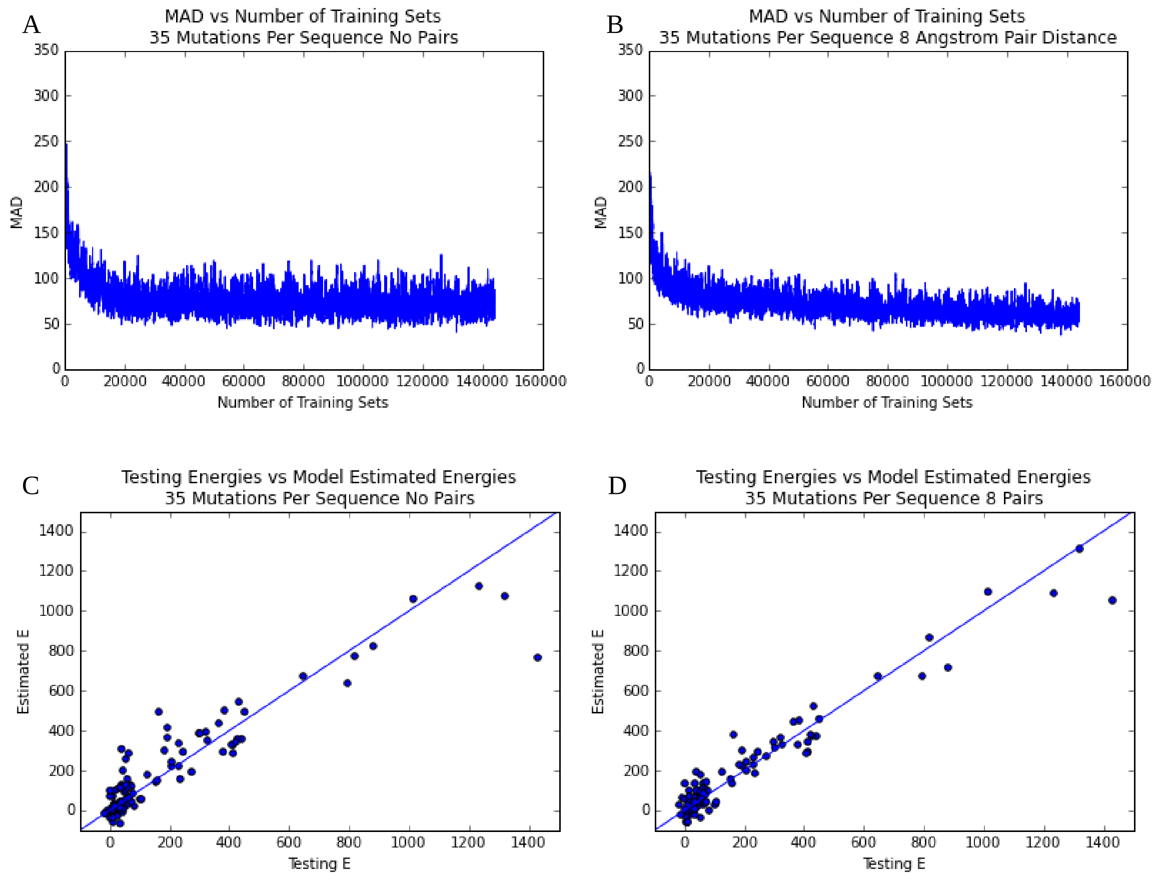


Figure 10: 35 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 35 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 35 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 35 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 35 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that (B) has not leveled off but almost has. It will likely still improve with more training sets. Also, (D) and (C) are actually quite similar so correlation seems to not matter as much at this threshold. At this mutation threshold pairs seem to still have some influence on model accuracy.

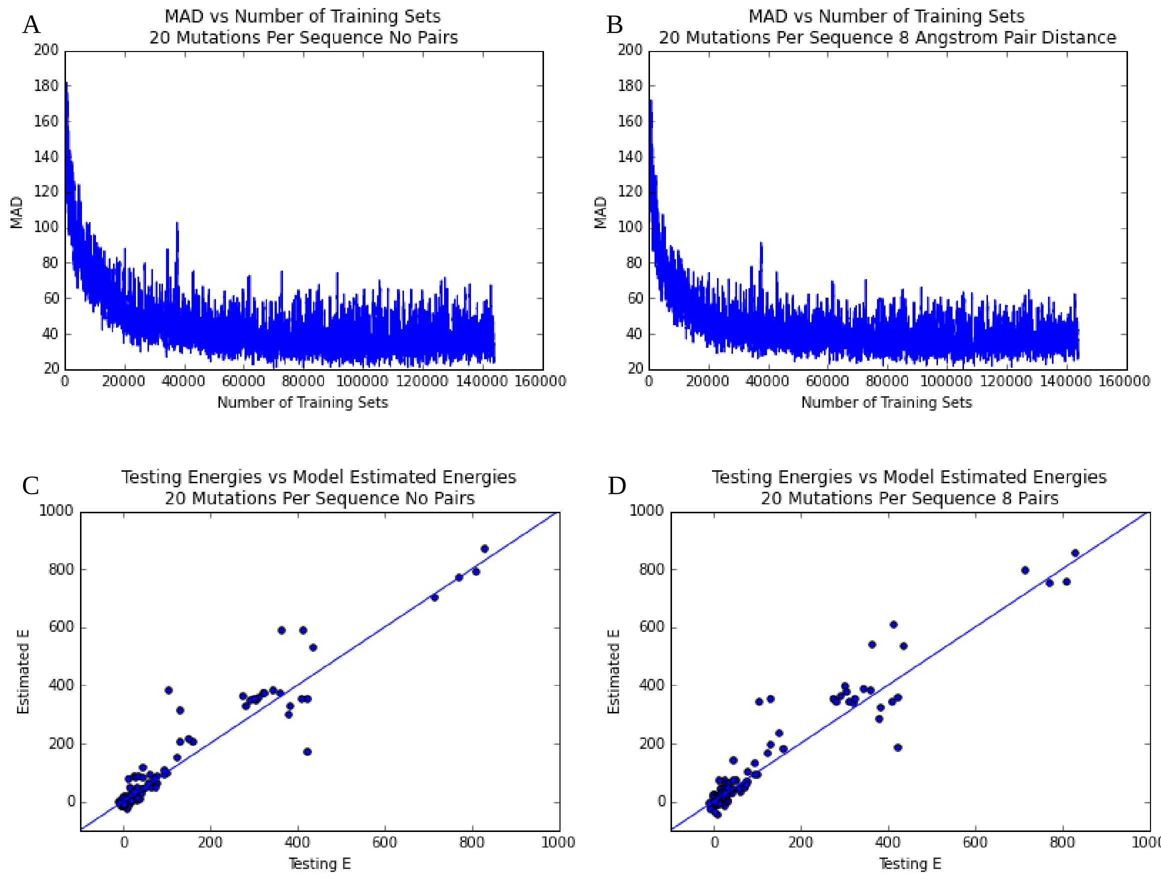


Figure 11: 20 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 20 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 20 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 20 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 20 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that (A) and (B) seem to have both leveled off and are nearly identical. Also, (C) and (D) are nearly identical. At this mutation threshold pairs seem to no longer influence model accuracy.

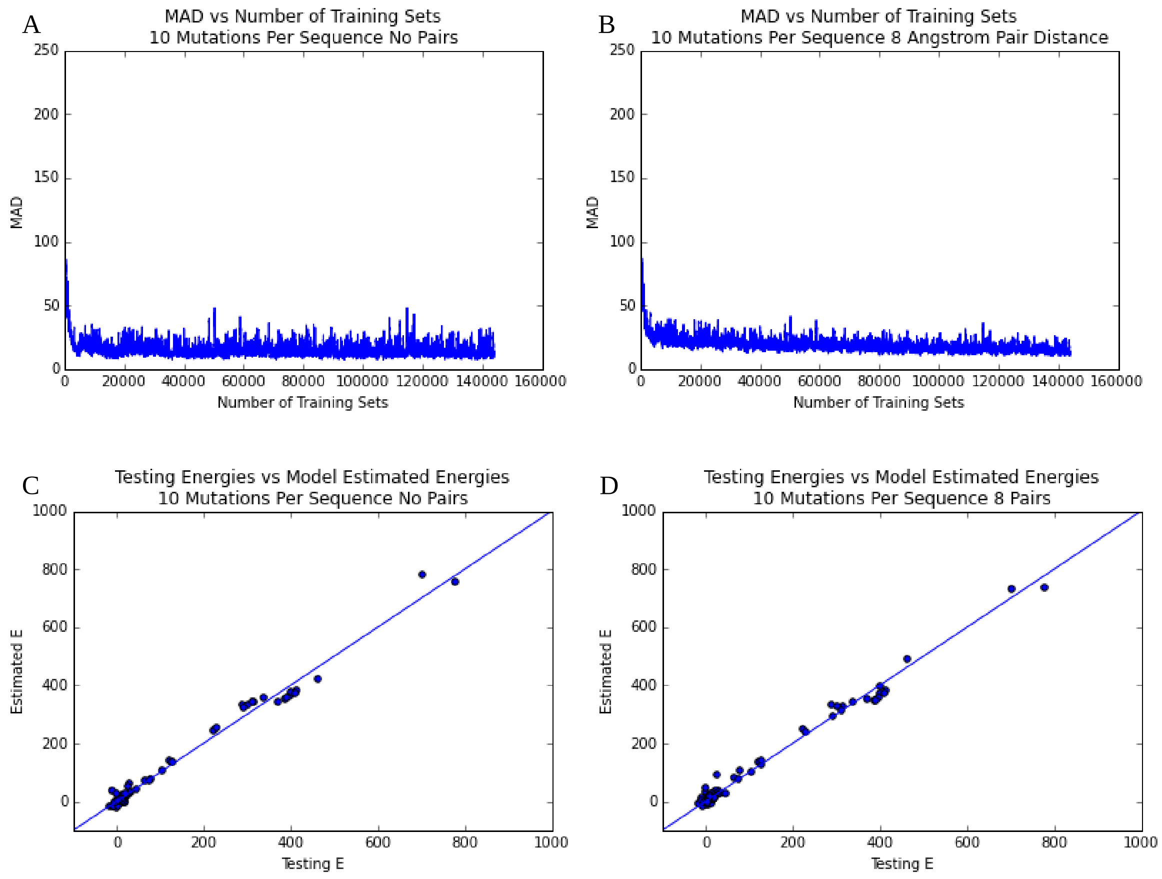


Figure 12: 10 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 10 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 10 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 10 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 10 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that (A) and (B) seem to have both leveled off and are nearly identical. Also, (C) and (D) seem to be identical. At this mutation threshold pairs seem to also have no influence on model accuracy.

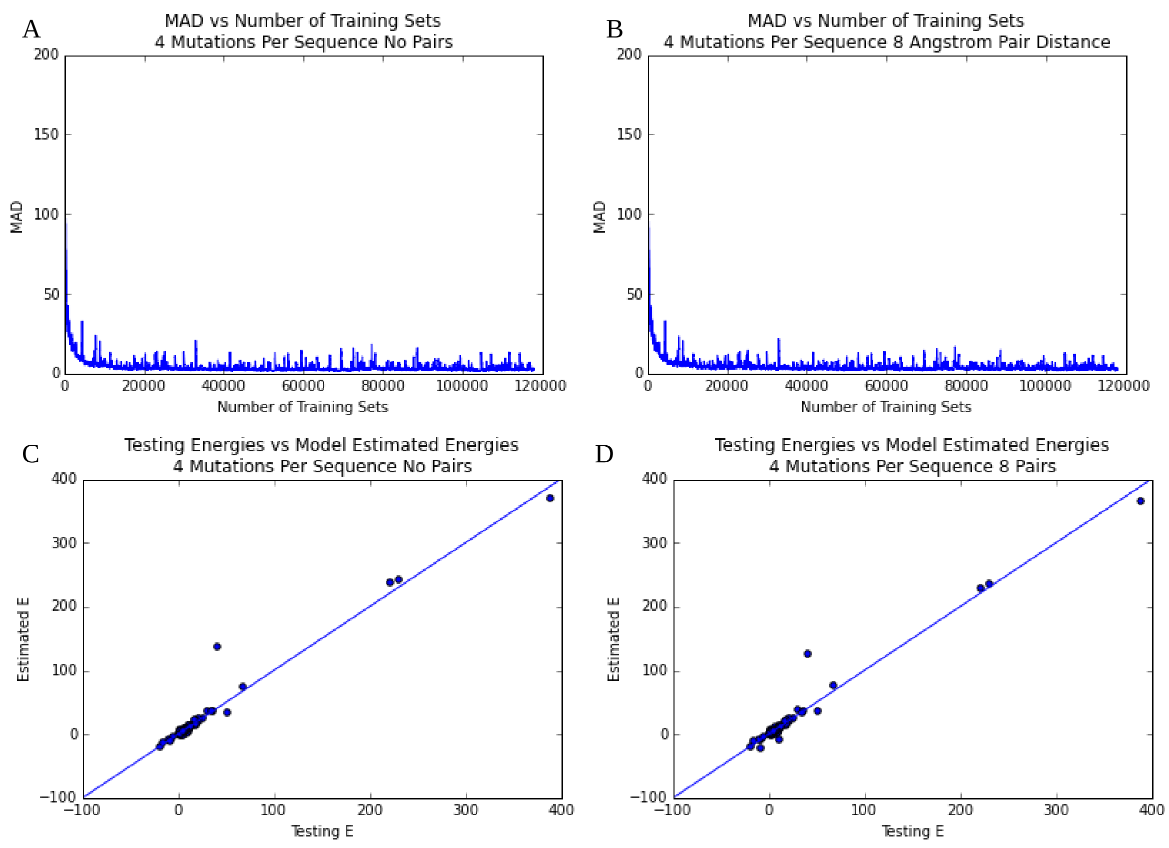


Figure 13: 4 Mutations per Sequence Pair Comparison
(A) MAD vs Number of Training Sets for a model trained on sequences with 4 mutations per sequence and only single interactions. (B) MAD vs Number of Training Sets for a model trained on sequences with 4 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. (C) Model Estimated Energy vs Rosetta Energy for a model trained on sequences with 4 mutations per sequence and only single interactions. (D) Model Estimated Energy vs Rosetta for a model trained on sequences with 4 mutations per sequence with single interactions and 8 angstrom pair distance two-body interactions. We see that (A) and (B) seem to have both leveled off and are also essentially identical again as in Figure 11 and Figure 12. Also, (C) and (D) look identical. At this mutation threshold pairs seem to also have no influence on model accuracy.

As we can see from the Figures 7-13 and Table 3, the 8 Angstrom pair distance seems to improve model accuracy for the 76, 65, 50, and 35 mutations per sequence models. Once we reach the 20 mutation per sequence model we see that the MAD vs Number of Training set plots (Figures 11-13 A and B) look nearly identical for no pairs vs 8 Angstrom distance pairs and ending MAD values are nearly identical as shown in Table 3. Upon first inspection at 4 and 20 mutations per position the inclusion of pairs seems to actually make the models less accurate according to Table 3. This may just be due to the specific data sets trained on as the difference in MAD values between pairs and no pairs in these numbers of mutations per sequence as well as 10 mutations per sequence seems to be marginal. Going forward it appears that pairs do not greatly influence models for our final restricted sequence space of quadruple mutants per sequence. The most important factor of our models seems to be restricting the number of mutations per sequence to a max of 4 as well as utilizing the sequence alignment for allowable amino acids per position. However, we still select to train two sets of all 100 conformational models. The first set of models will have single interactions and no pairs included while the second set of models will have single interactions as well as pairs with a set threshold distance between them.

To select the threshold distance on pairs for final model training we select to compare No pairs, 4 angstrom pair distance, 8 angstrom pair distance, and 12 angstrom pair distance. We use the same sequence and energy data as in the previous table (quadruple mutant data set). We do not include plots of this data as each model produces largely identical plots.

Pair Distance	Avg MAD of last 1000 Values	Avg CC of last 1000 Values	Number of Model Parameters
No Pairs	2.8544475	0.99	1445
4 Angstroms	2.7437790	0.99	3298
8 Angstroms	2.9023789	0.99	8212
12 Angstroms	3.2178822	0.99	14332

Table 4: Pair Distance Comparison for Quadruple Mutant Models
Here we show comparisons of MAD and CC for only single interactions, 4 angstrom pair distance two-body interactions, 8 angstrom pair distance two-body interactions, and 12 angstrom pair distance two-body interactions.

Displayed in Table 4 we see that the lowest MAD values come from the 4 Angstrom distance pair model. Although most of the models appear to give very similar performance, we select to go forward with an only single interaction model as well as a model that includes pair interactions that are closer than 4 Angstroms away in the 3-D structures.

Next, we wanted a way to potentially improve accuracy of our models further. We did this through the use of weighted linear regression. The implementation and proof for the SGD algorithm including weights is identical to the unweighted equations so it will be skipped for efficiencies sake. Below are the general linear regression algorithm with weight matrix (equation 11) and final stochastic gradient descent algorithm also with weights included (equation 12):

$$\vec{J} = \left(\overrightarrow{X^T} \cdot W \cdot \vec{X}\right)^{-1} \overrightarrow{X^T} \cdot W \cdot \vec{E} \qquad (11)$$

$$w_j^{(i+1)} = w_j^{(i)} - \gamma^{(i)} \left(\left(w_j^{(i)^T} \cdot x_j^{(i)}\right) \cdot \omega - \left(y^{(i)} \cdot \omega\right)\right)\left(x_j^{(i)}\right) \qquad (12)$$

The variable ω is a scalar weight depending on the energy value of the current sequence to be trained by the model. Before selecting our weight distribution we first wanted to examine the energy distribution of the State 0 conformation. We did this through finding the median energy value of a large data set of 4 mutations per sequence with the top 4 amino acid sequence alignment. We also generated the following histogram plot with this data:

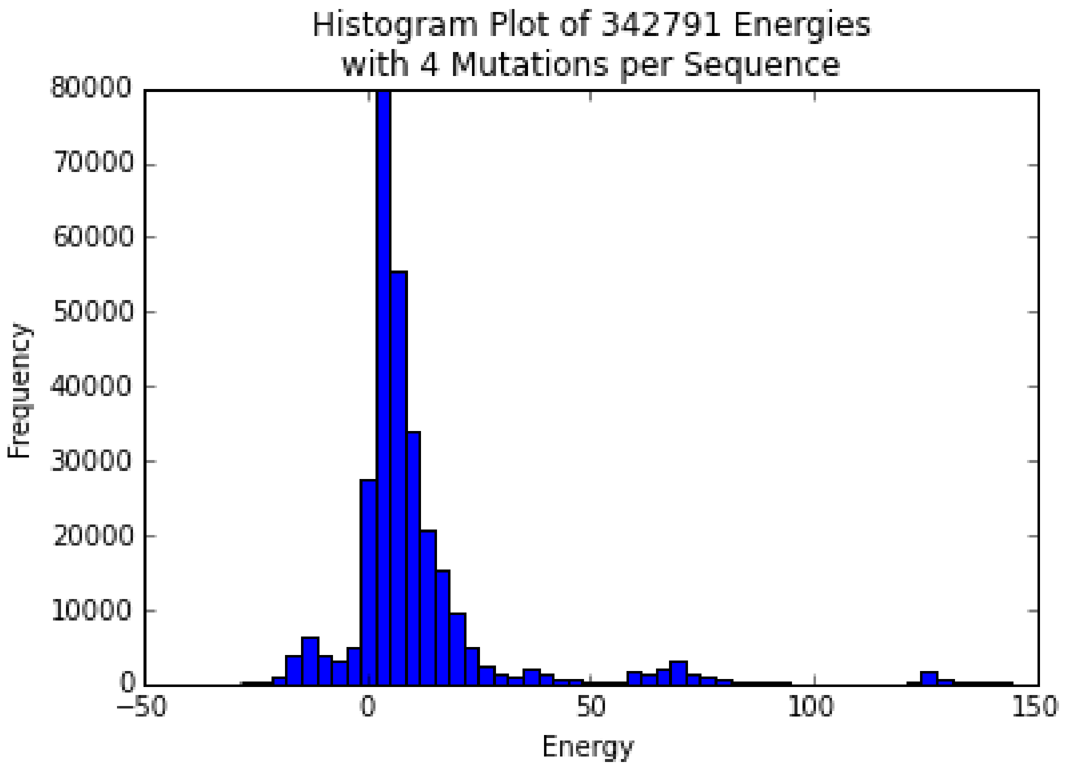


Figure 14: Histogram Plot of Energies with 4 Mutations per Sequence for State 0
Most of the energy values seem to be at values less than 50 REU. The vast majority of energies are just above a value of 0. We restrict the x-axis to (-50,150) for presentation purposes.

From Figure 14, we see that most of the energies fall below a value of 50. Further investigation of this data set reveals a median value of around 7. We then wanted to implement this median value into how we assign weights for our energies. We wanted to create a weight distribution that fully weights low energies (less than 7) and then weights higher energy values (greater than 7) in a decaying fashion. The final weight distribution we came up with is presented below as an exponentially decaying function:

$$\omega(y) = \min\left(1, e^{(7-y)} / 1200\right) \qquad (13)$$

Below is a picture of a plot of the weight distribution. As pictured, if an energy value is less than 7 the weight gets assigned as 1. Once an energy value is larger than 7 we apply the exponentially decaying function. We select to divide the exponentially decaying function by 1200 because we find this to be optimal for producing the most accurate models. This selection leads to an energy value of 2000 REU receiving a weight of .189 which seems reasonable given that an energy of 2000 REU was around the maximum value we are saw in the energy distribution.

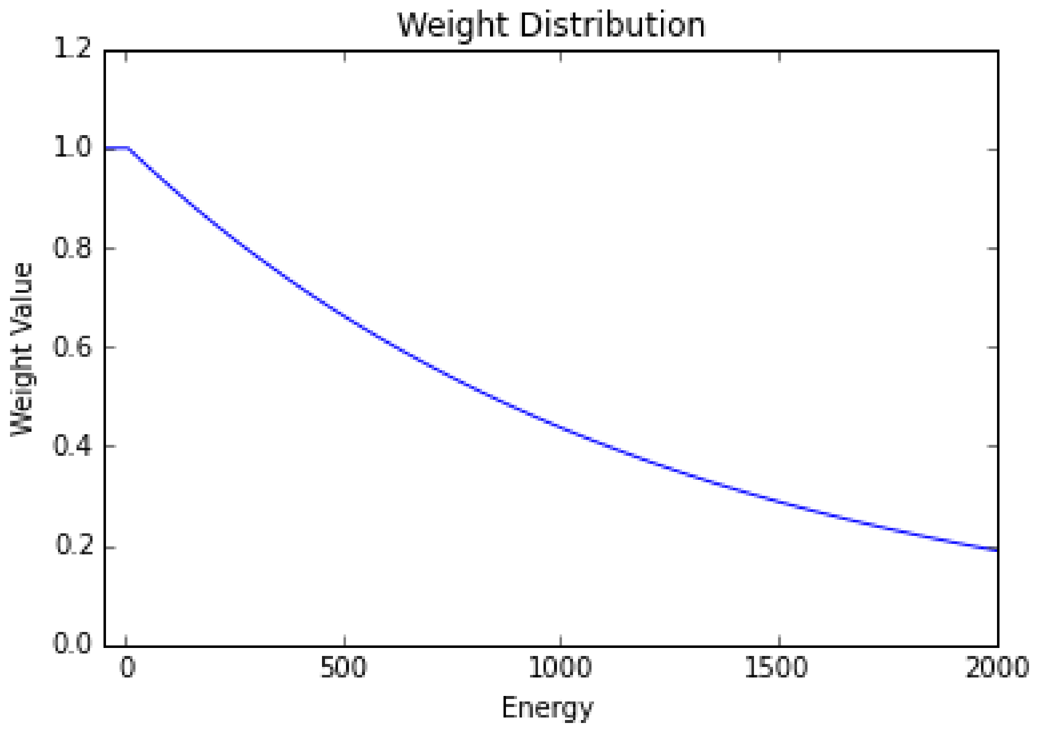


Figure 15: Weight Distribution Values Based on Energy Being Trained
A weight of 1 is assigned at energy values less than an REU of 7. An exponentially decaying function then assigns weights for energies above an REU of 7 as presented in the distribution.

When applying this weight distribution to model training we saw slight improvement in accuracy of our quadruple mutant models. In Table 5 we present the results of comparing an unweighted quadruple mutant model and a weighted quadruple mutant model utilizing the same quadruple mutant data set as before.

Weights	Pair Distance	Avg MAD of last 1000 Values	Avg CC of last 1000 Values	
Yes /1200	No Pairs	2.6054508	0.99	229
No	No Pairs	2.8544475	0.99	229
Yes /1200	4 Angstroms	2.5281883	0.99	3298
No	4 Angstroms	2.7437790	0.99	3298

Table 5: Model Accuracy between weighted and unweighted Stochastic Gradient Descent models for Quadruple Mutants
We observe the average MAD value for the last 1000 Values improved from 2.85 to 2.60 for the single body models and the MAD values improved from 2.74 to 2.52 for the 4 Angstrom pair models.

Upon further model accuracy testing with a completely different data set (shown below) we observed that the weighted model performs better than the unweighted model with low energy values as well (less than 7 REU). This is very important for our purposes as we are most interested in being accurate on low energy sequences.

Weights	Avg MAD of last 1000 values for Energies less than 7	Avg MAD of last 1000 values for Energies 7<x<100	Avg MAD of last 1000 values for Energies greater than 100
Yes /1200	1.15242503453	2.52471694016	4.50868347544
No	1.34844898124	3.3187129332	5.01383673149

Table 6: Weighted vs Unweighted Model Accuracy at Different Energy Values for an Only One-Body Model
Table showcasing the accuracy of a weighted and unweighted single interaction model. Models were tested against an independent data set in order to determine accuracy at respective energy ranges. Our models perform best on energy values less than 7.

In Table 6 at every energy range the weighted model outperforms the unweighted model. At first look it seems like an easy selection to include the weight distribution into model training. However, we do not readily know the energy distributions of the other 99 conformations. This could create problems when utilizing this weighting scheme on other conformations. However, based on the wild type energy landscape of all the conformations we see that most of the energies of the wild type sequence are quite close in value. Therefore, it would seem likely that some of the conformational states will have similar energy distributions. Thus, we selected to train all 100 conformational models with weights included as well as another set of models without weights.

These were the final model parameters we set to include in model training. We were able to optimize our models and achieve great accuracy through utilization of the sequence alignment of ubiquitin variants (selecting the top 4 amino acids per position), only allowing for sequences to be quadruple mutants, including certain pair interactions given a geometric constraint, and applying weights based on the energy of the sequence being trained. Going forward we selected to train three sets of all 100 conformational models. The model variations selected to be trained were as follows: single interactions only without weights, single interactions only with weights, and single interactions plus pair interactions closer than 4 angstroms without weights. In the end during optimization we select the model set that performs the best across all conformations.

Modeling the Energy Landscape

Markov State Model

The MSM of ubiquitin was generated through collaborations with Dr. Greg R. Bowman's lab at Washington University in St. Louis and consists of 100 separate conformations which serves as the coarse-grained conformational space for our system. Each conformation in the MSM was output as a PDB file or a file of the atomic coordinates of every atom in the structure. Below we present several pictures of the generated conformations of ubiquitin overlaid with each other from the MSM to showcase how ubiquitin can change drastically through conformational space. Each of the following pictures was created by loading each PDB file into PyMOL, running the align command to structurally align the PDB's, and selecting to view them based on secondary structure (alpha helices, beta sheets) [52].

A. States: 0-green, 2-cyan, 12-purple, 26-yellow

B. States: 33-pink, 47-white, 54-purple, 69-orange

C. States: 71-green, 85-blue, 97-magenta, 93-gold

D. States: 7-purple, 94-silver, 44-blue, 59-yellow

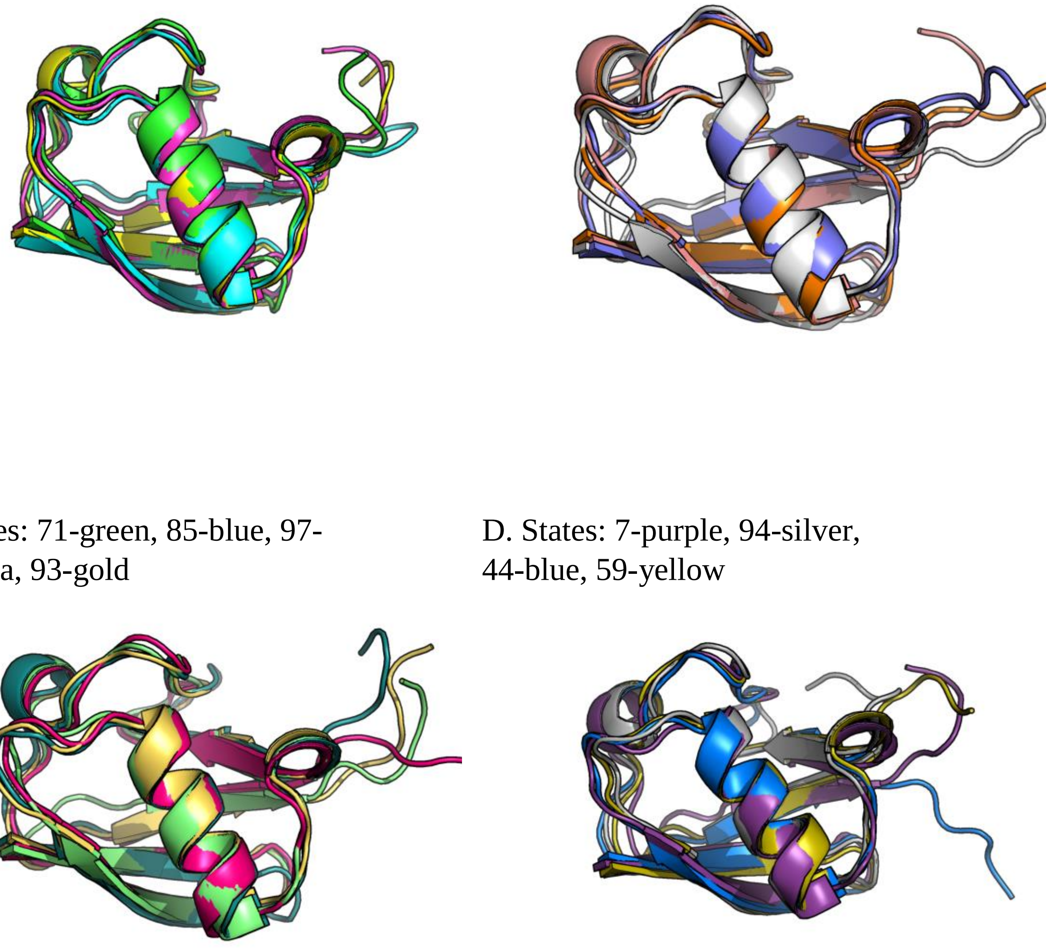


Figure 16: Picture Representation of Select Conformational States of the Ubiquitin MSM Overlaid and Aligned
Each of the pictures (A-D) consist of 4 chosen states which are aligned with one another structurally and overlaid with each other. We select to overlay 4 states per picture for a total of 16 showcased states. As we can see, there is definite conformational variability from state to state.

Although this image representation of the space is limited to just 16 of the 100 states it showcases the variability of ubiquitin over conformational space. Based on the images. there seems to definitely be conformational changes between states which should have an impact on the energy distributions of each state as well. Specifically, in Figure 16-D we see that State 44 (blue) has a large change on the right side of the picture at the end of the chain as compared with the other states. Also, structures overall seem to be slight variations of one another with small changes all throughout the entire structure. We utilize each of these separate conformational states during model training where each model is based on one of the one-hundred separate conformational states.

Mean Field Approximation of all Conformational States

Upon success in modeling a single states sequence-energy mapping followed by successfully coarse graining conformational space we then moved on to modeling the entire sequence-energy mapping for all conformational states. Our online machine learning approach was vastly important for this step as it allowed us to train all 100 conformational models simultaneously and efficiently. We wanted to train several different model variations when we applied the modeling to the entire coarse grained conformational space. We did this so we could compare which model variations were best for particular states and to avoid potential inaccuracies that could occur due to the differing energy distributions associated with each conformational state. For example, just because pairs seem to not matter for State 0 does not mean that pairs may not improve a model for a different conformational state. We selected to train three different model variations each over all 100 conformational states for a total of 300 models. The model variations we trained were as follows: single interaction parameters only with no weights, single interaction parameters only with weight distribution, and single interaction parameters plus 4 angstrom pair distance interaction parameters with no weights. We were able to achieve good accuracy for all three sets of the 100 different conformational models and below is a table of the relative accuracy of the models:

Model Type →	single interaction parameters only with no weights	single interaction parameters only with weights	4 angstrom pair distance interaction parameters with no weights
Range of Avg MAD of last 1000 values	Number of Conformational Models that fit the Criteria	Number of Conformational Models that fit the Criteria	Number of Conformational Models that fit the Criteria
Less than 1	0	0	0
Less than 2	17	14	19
Less than 3	61	45	56
Less than 4	83	69	82
Less than 5	89	76	90
Less than 6	94	86	97
Less than 7	95	88	98
Less than 8	97	88	98
Less than 9	97	88	99
Less than 10	99	90	99
Less than 11	99	91	100
Less than 12	100	91	100

Table 7: Relative Accuracy of all 100 Conformational Models for Quadruple Mutants with various Model Variations
This table presents the relative accuracy of all 100 models for each of the 3 model variations. We test the accuracy by presenting how many models in each variation have an average MAD over the last 1000 values of less than a threshold value (in our case values between 1-12). The vast majority of models in each of the three variations seem to have an MAD of at least less than 6 REU.

Based on the table, the most accurate set of 100 models seem to be those with single interaction parameters only with no weights as well as the 4 angstrom pair distance parameter model with no weights. Both models showcase very similar accuracy but for initial optimization we selected to use the single interaction/unweighted model. The 4 angstrom pair distance with single interaction parameters and no weight model is quite accurate and is essentially on par or even better than the single interaction/no weight model. Although either model could be used for optimization, we select to use the single interaction model due to the fewer parameters associated with the model.

The single interaction models have 229 parameters while the 4 angstrom pair models have parameters around 3000 parameters (varies with state). These increased parameters for the pair models will lead to slightly longer calculation times. Given that we get nearly identical results from an accuracy standpoint we select to go forward with the single interaction/unweighted model as we believe results will be similar for both models. We also see that the weighted single interaction model has poor performance. The weights we selected were good for optimizing the State 0 conformation. However, we did not readily know the energy distributions of the other conformations. Therefore, selecting the weight distribution as we did was not guaranteed to work for the other conformations. We thought that the weights might make some of the lower energy conformations train more accurate models. However, as we can see from the table the pair and single unweighted models outperform the weighted model at every threshold MAD value. Given the inaccuracy of the weighted model with only single interactions, we decided to not train the single interaction parameters plus 4 angstrom pair distance interaction parameters with weight distribution as we expect this model variation to perform as bad as the other weighted model.

As a whole, we see that for our most accurate model variation that 83 of the models had an average MAD value of less than 4. Based on the data we were successful in our ability to accurately model the energy landscape of all conformational states. There were a few conformations that gave us some difficulty however. These inaccurate conformations were generally those that had large energy distributions over sequence – energy space. Given the high energies of these specific conformations they are mostly seen as off target conformations anyway and could be discarded if needed. These conformations provided difficulty during optimization as well due to their inaccuracies.

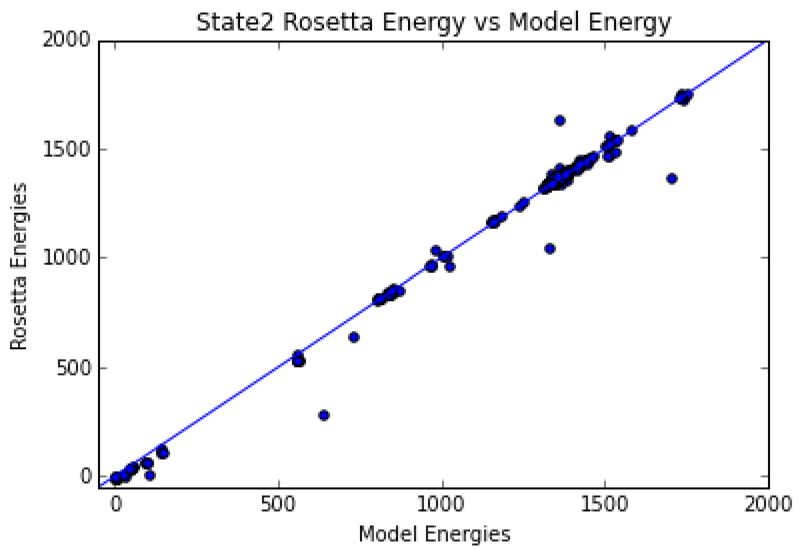


Figure 17: Rosetta Energy vs No Pair No Weight Model Energy for State 2
Plot of Rosetta Energy vs Model Energy for State 2. We see that there are groups of energies scattered all through the plot from around a value of 0 to a value of around 1500.

As seen in the plot above, we showcase the accuracy of our model for State 2. For our no pair no weight model State 2 had an average MAD of 9.73 for the last 1000 terms. Although this is not terrible it is one of the worst models we generated. We think this inaccuracy is attributed to the large energy spread for this conformation. As we can see in the plot above the energies span from 0 REU to nearly 2000 REU while State 0's energies span from around -30 REU to 100 REU. Although most of the energies in the plot seem to be quite accurate a few look to be very inaccurate. State 2 also had greater inaccuracy when trained on with the weighted model which is expected given then large amount of high energies present. Other inaccurate conformational models include States 91, 94, 96, and 98. All of these States as well as State 2 had MAD values greater than 7 for our most accurate modeling scheme (no pair no weights). Despite a few inaccuracies we believe we successfully modeled the sequence-energy mappings for all conformations based on the results from Table 7. Going forward we decided to run our optimizations with the most accurate models we generated which were the no pair no weight models.

Protein Design Using Coarse Grained Energy Landscapes

After successfully modeling the sequence-energy mapping of all conformations we then moved forward to the design of energy landscapes. Below we present the energy landscape of the wildtype sequence ubiquitin for our specified MSM. Most of the energy values of wild type ubiquitin for all conformational states fall below a value of zero. During optimization many of these states are destabilized and the energy landscape changes substantially.

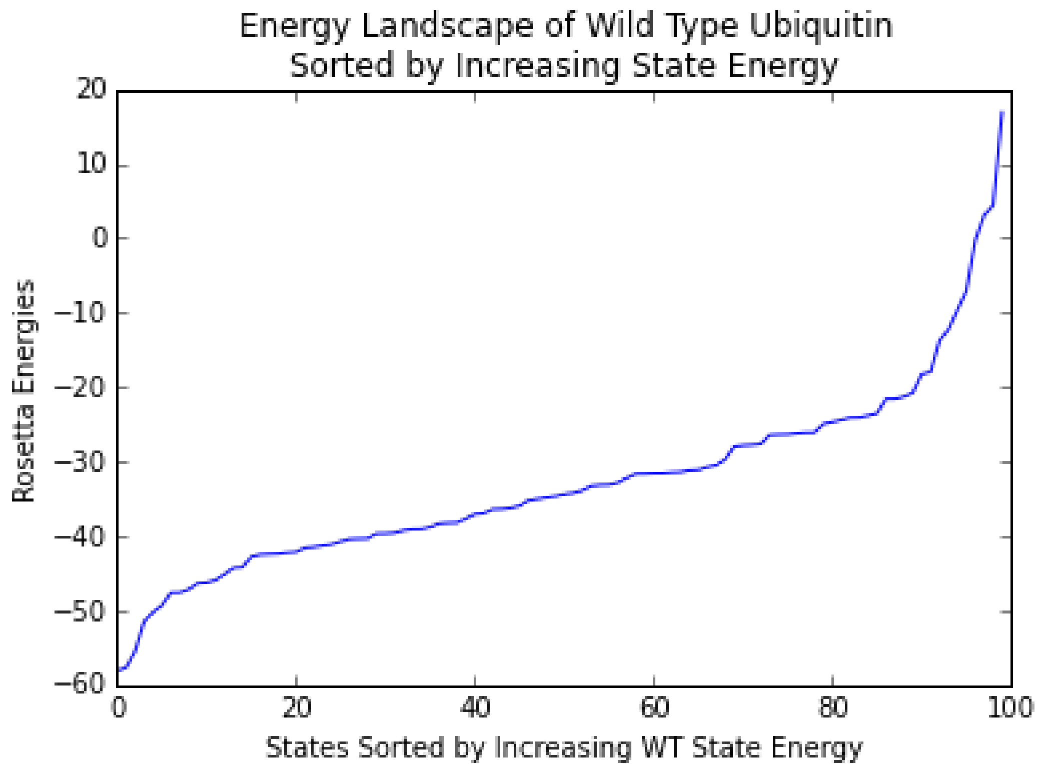


Figure 18: Energy Landscape of Wild Type Ubiquitin
Energy values of wild type ubiquitin generated through Rosetta presented in an energy landscape plot over all conformational states from the MSM. States are sorted by increasing energy value.

The goal of this project was to completely alter this energy landscape and design sequences that stabilize certain conformational states. We did this through maximizing the

Boltzmann Distribution for either one state or for a pair of states. The general protocol for this was to run the optimization maximizing the Boltzmann Distribution or minimizing the dual state objective function for a specified state or pair of states using our sequence-energy approximations. After this, to check the accuracy of our optimized sequence we would go back and calculate energy values using Rosetta (of the optimized sequence for all 100 conformations) and use these energy values to calculate Boltzmann Distribution values (probabilities). We then calculated the percent error between the probabilities our energy model approximations give and the probabilities generated through the true Rosetta energy values.

We ran optimizations that maximize the probability of each conformational state once which generates a single mutated sequence per optimization for a total of 100 optimized sequences. We also ran optimizations that maximize the probabilities of each pair of conformational states once for a total of 4950 total optimized sequences. We then tested the accuracy of each of these optimized sequences to see if certain optimized states give false positive results for maximizing probability (minimizing energy). For these optimizations we select to optimize sequences that can at most be triple mutants from the wild type sequence. We do this to ensure a higher degree of accuracy as our models were trained on quadruple mutants. Below are tables that showcase the accuracy of the sequences we optimized:

Model Type →	single interaction parameters only with no weights	single interaction parameters only with weights	4 angstrom pair distance interaction parameters with no weights
Percent Error Threshold Value Comparing the Probability Values of the Optimized State	Number of Optimized Sequences that fit the Percent Error Threshold	Number of Optimized Sequences that fit the Percent Error Threshold	Number of Optimized Sequences that fit the Percent Error Threshold

Less than 0.01	16	15	17
Less than 0.1	26	26	27
Less than 1	47	49	52
Less than 10	57	61	66
Less than 100	68	72	72
Less than 1000	75	79	78
Less than 10000	83	84	86

Table 8: Accuracy of Triple Mutant Single State Optimized Sequences for all Three Model Variations
For each model variation we showcase how many of the optimized sequences of 100 fit a specified percent error threshold value. Percent error is calculated through comparing our model energy generated probability values with Rosetta energy generated probability values. Each of the designed sequences is optimized for 1 of the 100 states where each state is optimized once. We see that around half of our sequence designs from our models are within 2-3 orders of magnitude of the Rosetta comparisons.

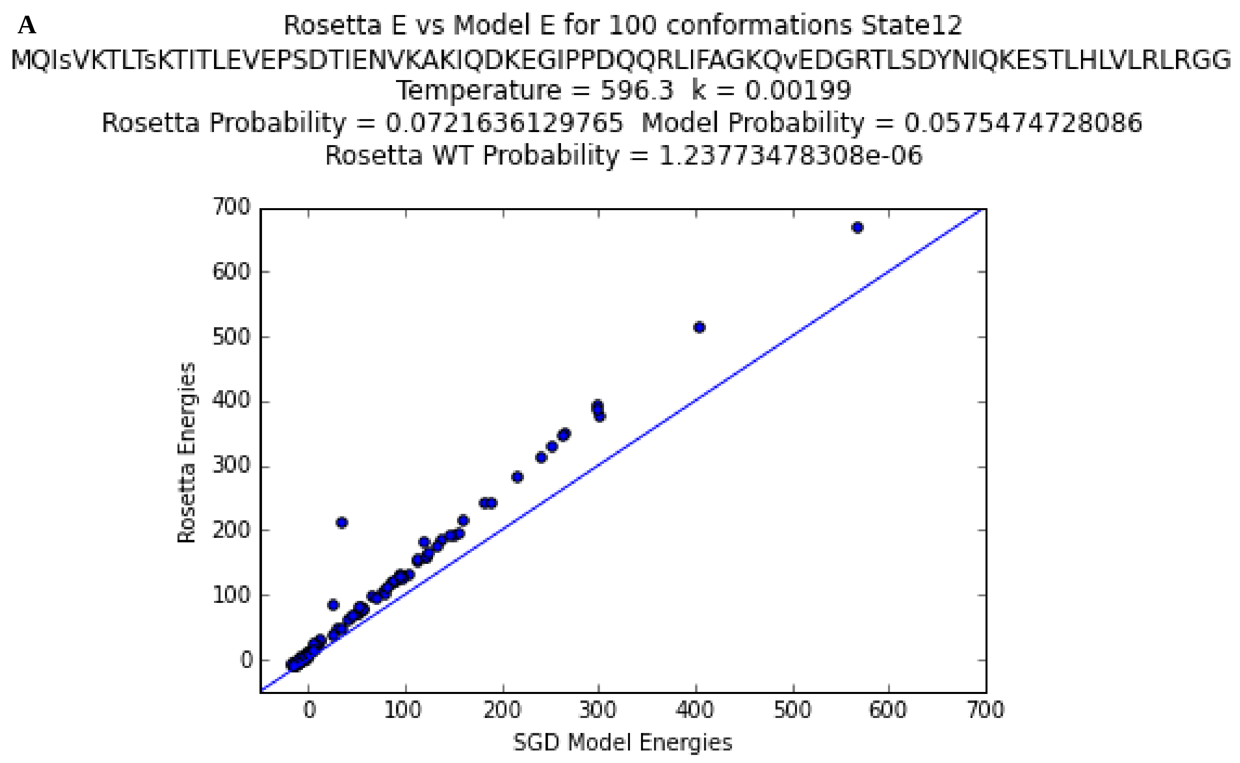
Percent Error Threshold Value Comparing the Probability Values of the Optimized State	Number of Optimized Sequences that fit the Percent Error Threshold for First State	Number of Optimized Sequences that fit the Percent Error Threshold for Second State
Less than 0.01	15	29
Less than 0.1	218	224
Less than 1	1655	1514
Less than 10	2737	2437
Less than 100	3492	3227
Less than 1000	3930	3666
Less than 10000	4224	3979

Table 9: Accuracy of 4950 Dual State Optimized Triple Mutant Sequences using the No Pair No Weight Models
For the no pair no weight model we showcase how many of the optimized sequences of 4950 fit a specified percent error threshold value. Percent error is calculated through comparing our model energy generated probability values with Rosetta energy generated probability values. Each of the designed sequences is optimized for 2 of the 100 states where each pair of states is optimized once for a total of 4950 optimized sequences. We showcase the percent error threshold value for the probability values of both states that were optimized. We see that around half of our sequence designs from our models are within 2-3 orders of magnitude of the Rosetta comparisons.

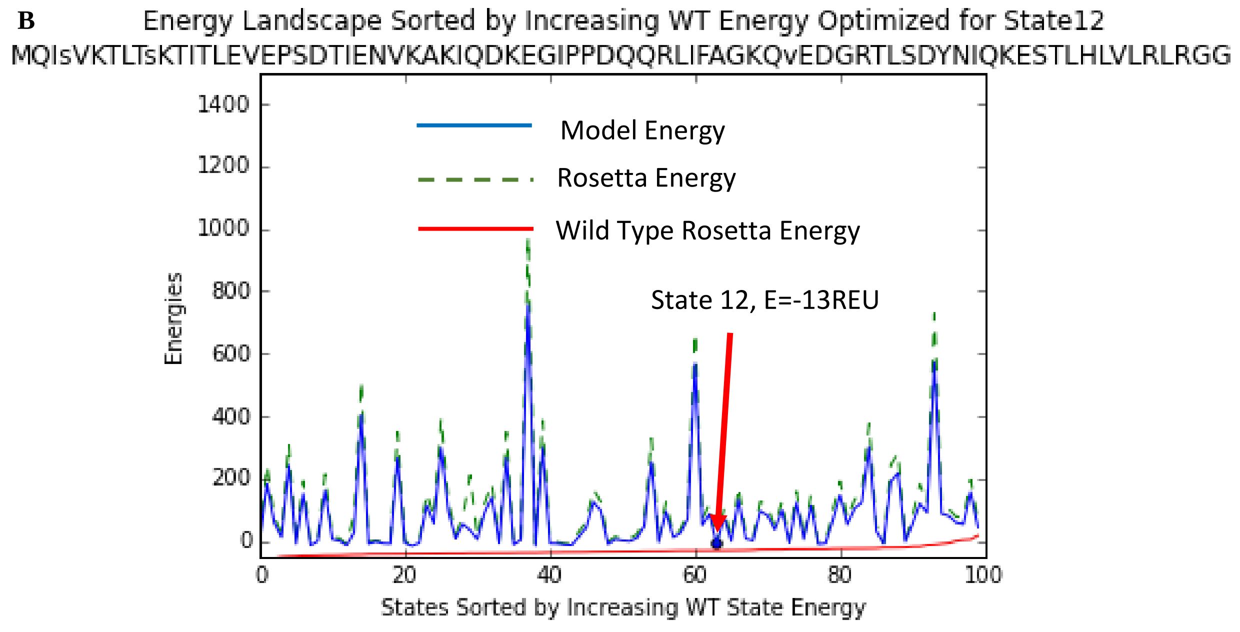
From Table 8 we see that all three model variations are essentially on par with one another. The 4 angstrom pair model actually seems to perform the best and the single interaction/weighted model actually performs well too. Given that all of the models perform similarly we still feel confident running optimizations with the single interaction/unweighted model. However, running optimizations with the 4 angstrom pair model may produce better results but would take longer to compute energy values during optimization. We believe the weighted single body model gives accurate results of probability given that these models emphasize low energies during training. Therefore, the weighted models have have higher MAD values for all states, but given that they are accurate at low energies then these models can provide accurate results for calculating probabilities of the optimized (low energy) sequences. From Table 8 and Table 9 we see that around half of our optimized sequences for single state and dual state optimization achieved a percent error on Boltzmann Probability of less than 10. Although this may not seem ideal we actually believe that this is a fairly accurate result given how we calculate percent error. We calculate our percent error values through comparing the Boltzmann Distribution values (for the selected state) of our models with those generated through Rosetta Energy values so it is essentially it is a test to see how accurate our energy models perform on the designed sequences over all conformational states. If our probabilities are off by 2 or 3 orders of magnitude or around 10 to 100 percent error then our predictions are still quite accurate. Based on the Table 8 and Table 9, the majority of our predictions of probability are within a few orders of magnitude compared with the Rosetta probabilities. Therefore, we believe our optimizations provide accurate results based on the scale of the optimization and compared with the wild type probabilities.

Next, we showcase results of a few of our designed sequences and landscapes for single state and dual state optimization. We present Rosetta energy vs model energy prediction plots for all states, energy landscape plots sorted by increasing wild type energy, and plots of shifted probability across all states sorted by increasing wild type state probability. For our shifted probability plots we select to plot our probabilities in increasing order and we select to plot the inverse of the logarithm of our probability values. Therefore, a lower value on the shifted probability plot corresponds with a higher probability. We do this transformation to more easily showcase highly probable states for the designed sequences. When looking at the shifted probability plots a more stabilized state will be represented by a downward spike in green (mutant shifted probability). The nature in which we showcase specific states is largely arbitrary as any state could be selected and optimized for. However, we do showcase several states that are easily destabilized or states that have high wild type energies. In each of the state optimized plots we list the mutation that generated the energy values in the title of the plot. First, we showcase results for a few randomly selected triple mutants that each stabilize one conformational state.

A


Rosetta E vs Model E for 100 conformations State12
MQIsVKTLTsKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAGKQvEDGRTLSDYNIQKESTLHLVLRLRGG
Temperature = 596.3 k = 0.00199
Rosetta Probability = 0.0721636129765 Model Probability = 0.0575474728086
Rosetta WT Probability = 1.23773478308e-06
Rosetta Energies
SGD Model Energies
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700

B


Energy Landscape Sorted by Increasing WT Energy Optimized for State12
MQIsVKTLTsKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAGKQvEDGRTLSDYNIQKESTLHLVLRLRGG
Model Energy
Rosetta Energy
Wild Type Rosetta Energy
State 12, E=-13REU
Energies
States Sorted by Increasing WT State Energy
1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100

C

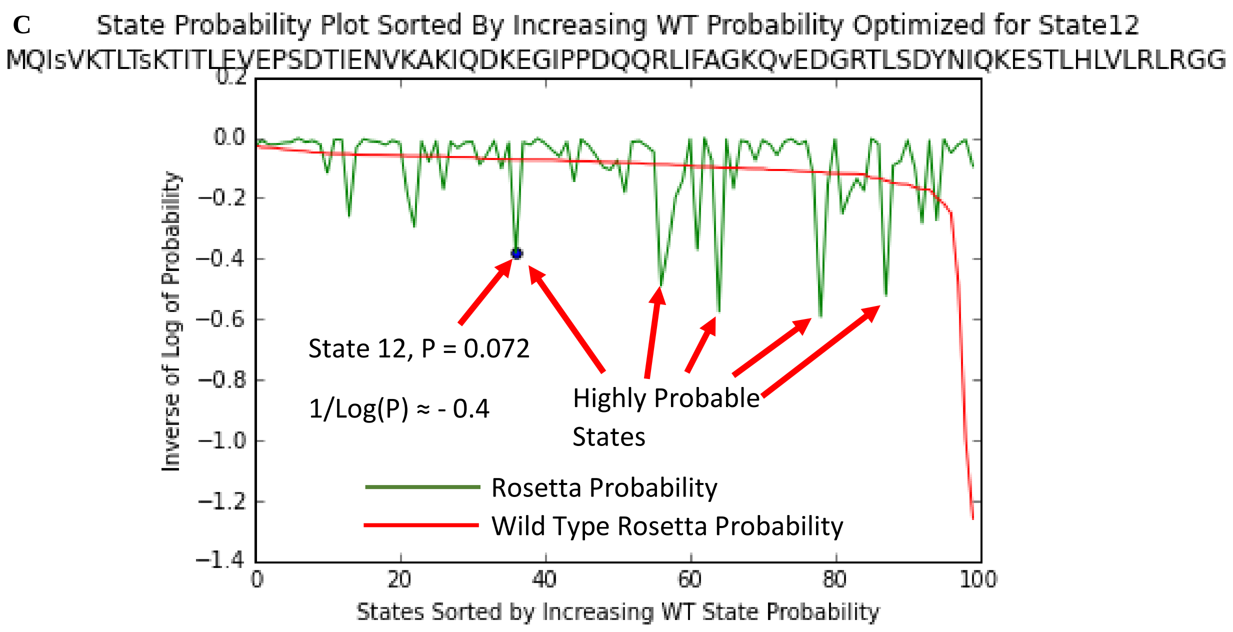
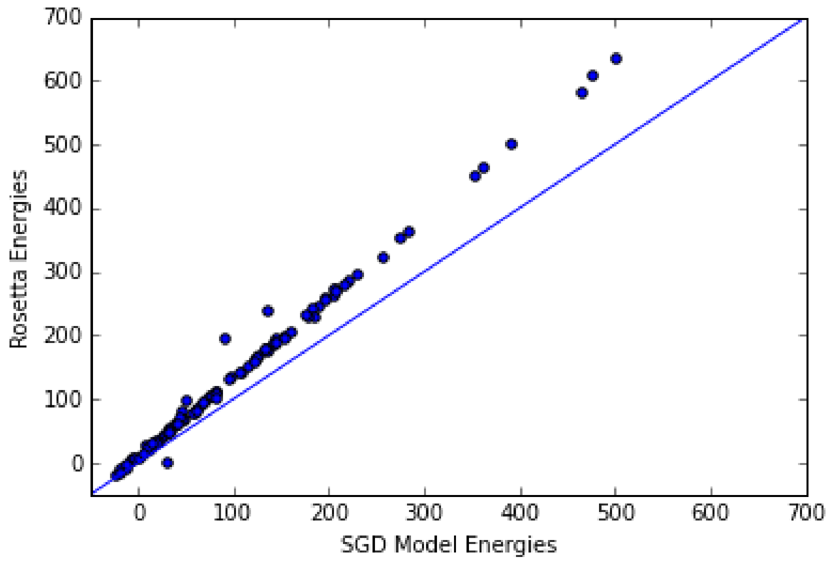


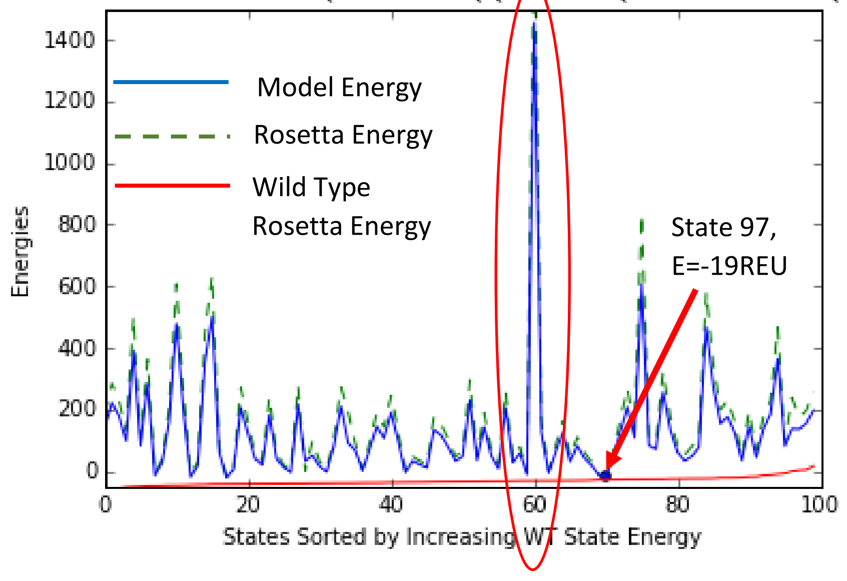
Figure 19: Triple Mutant Sequence Optimization for Stabilizing State 12
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see good linear agreement especially at low energies. As energy raises our models seem to underestimate energy values. (B) Energy landscape of specified sequence. Blue line represents model energies green dashed line represents Rosetta energies, and red line represents wild type energies. In (B) we see good agreement between Rosetta and Model energies. (C) Shifted Probability Plot of specified sequence. The green line represents the probabilities calculated with Rosetta energies and the red line represents the probabilities of the wild type sequence calculated with Rosetta energies. We select to plot the inverse of the log of the probability value for each state. We do this so we can clearly see stabilized states. States that are below the red wild type line are seen as stabilized states for the specified mutation. We see that our algorithm optimized for State 12 (blue dot) as it is below the wild type value on the plot. This means that our optimization worked for stabilizing State 12. Our optimized sequence also has various other states that are highly probable as well.

In Figure 19-A, we showcase a comparison between model energies and Rosetta energies (of each state) for the sequence which was optimized for stabilizing State 12. We also show the energy landscape of the optimized sequence over all states in Figure 19-B. In Figure 19-A and Figure 19-B, we see that as energy increases our models get less and less accurate (deviate from the line of slope 1). This makes sense based on preliminary model testing for State 0, generally the lower energy values used for model testing the lower MAD values we got. Essentially our models perform better on low energy sequences and this becomes more apparent during optimization. In Figure 19-B we also see that the energy landscape has been entirely and accurately redesigned. The blue line corresponds to our model approximation energies of each state while the green dotted line corresponds with the Rosetta energy values of each state. Based on the landscape we can see that our models tend to underestimate high energy values but seem to be fairly accurate at lower energies. It is also important to note that our Rosetta and model probability values are also of equal magnitude. The value of State 12 was measured to be -13 REU for the optimized sequences based on our models. The wild type sequence has an energy of -31 REU for this same state. Although the energy value increased for this state raised for this sequence, the probability of this state existing is still higher due to the destabilization of other states. For our optimization we did not declare that energy for the optimized state must decrease. Although such a restriction could be easily implemented into the design problem. In Figure 19-C we present the probability of each state existing sorted by increasing wild type probability. We plot these probability plots by shifting the probability values by taking the inverse of the logarithm of probability for each state. This makes our shifted wild type probability data decreasing with increasing wild type probability. A stabilized state (more probable state) is seen as being a spike lower than the red wild type line. Essentially, if the green mutant line is above the red wild type line, then we have destabilization occurring in this state and the state has a low probability of existing. If the green mutant line is below the red wild type line then we have stabilization and the state has a higher probability of existing in this mutation. Our optimized state in this case, State 12, has been stabilized with respect to all other states. We see that it is below the red wild type line on the plot which is exactly what we were looking for in this optimization procedure. Although it is not the most probable state as seen in the plot, it definitely is one of the few stabilized states.

A Rosetta E vs Model E for 100 conformations Optimized for State97
MQIFVKTpTGKTITvEVEPSDTIENVKAKIQDKvGIPPDQQRLIFAGKQLEDGRTLSDYNIQKESTLHLVLRLRGG
Temperature = 596.3 k = 0.00199
Rosetta Probability = 0.150415009999 Model Probability = 0.182823012054
Rosetta WT Probability = 2.2043971988e-07



B Energy Landscape Sorted by Increasing WT Energy Optimized for State97
MQIFVKTpTGKTITvEVEPSDTIENVKAKIQDKvGIPPDQQRLIFAGKQLEDGRTLSDYNIQKESTLHLVLRLRGG



C

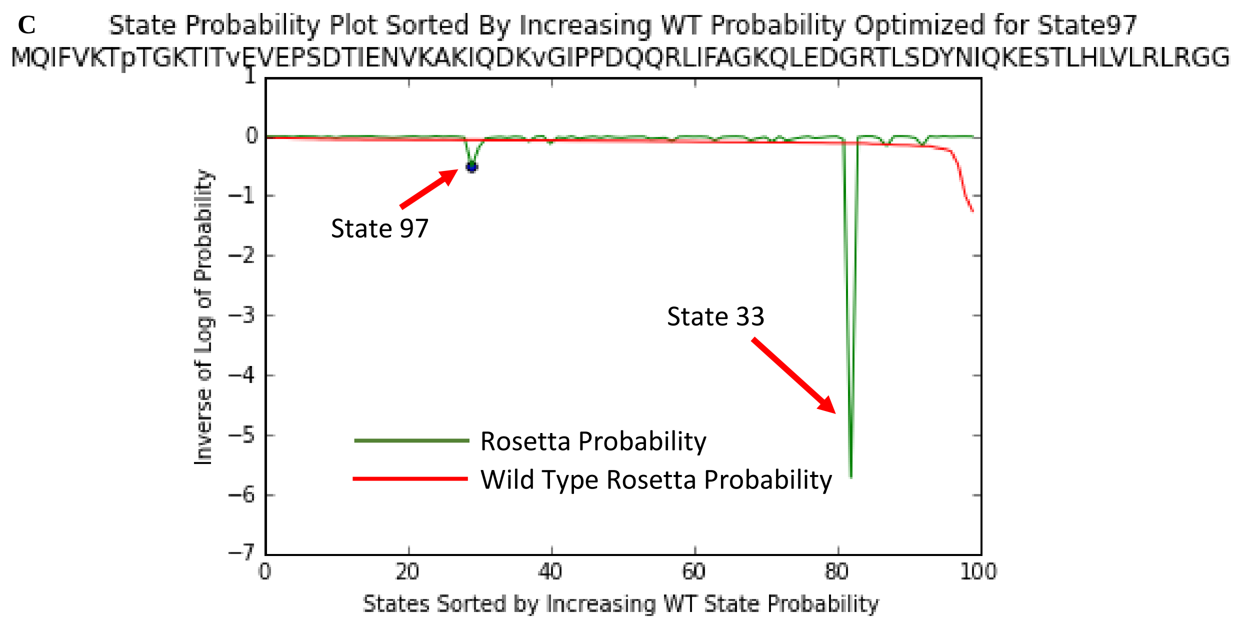


Figure 20: Triple Mutant Sequence Optimization for Stabilizing State 97
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see good linear agreement especially at low energies. As energy raises our models seem to underestimate energy values (B) Energy landscape of specified sequence. Again we see similar agreement as in Figure 19-A and Figure 19-B. However, in (B) we see that State 2 has been substantially destabilized to a value of around 1400 REU. (C) Shifted Probability Plot of specified sequence. We see that our algorithm optimized for State97 (blue dot) as it is lower than the transformed wild type probability and is one of the only probable states in the entire conformational landscape. At the right hand side, we do see a very probable state (State 33) that is substantially stabilized even more so than State 97.

In Figure 20-A and Figure 20-B we present another single state designed sequence to show that the energy landscape changes again but differently for a different design. In most designs we see, State 2 (circled in red in Figure 20-B) is substantially destabilized. As we can see, the energy value for State 2 is around 1400 for our model predictions as well as the true Rosetta value. Therefore, State 2 could potentially be discarded or treated as an off target state if deemed necessary for optimization purposes as the destabilization of this state seems to play a role in the optimization of other states. In Figure 20-C we see again that the probability of our optimized state, State 97, has increased from our optimization (presented as lower on the plot when we shift probability to inverse of the log of probability). So, we have successfully stabilized State 97.

However, we see that State 33 is even more probable, as its shifted value is around -6. Upon further inspection we see that State 33 has an energy of around -21 while State 97 only has an energy of -19. So, we were successful in stabilizing State 97 and as a consequence State 33 was stabilized as well.

Next, we wish to compare and contrast two landscapes to show just how different landscapes can be when optimized for different conformational states. We select to stabilize State 59 and State 0 separately for two different optimized sequences. We then compare energy landscapes and shifted probability plots.

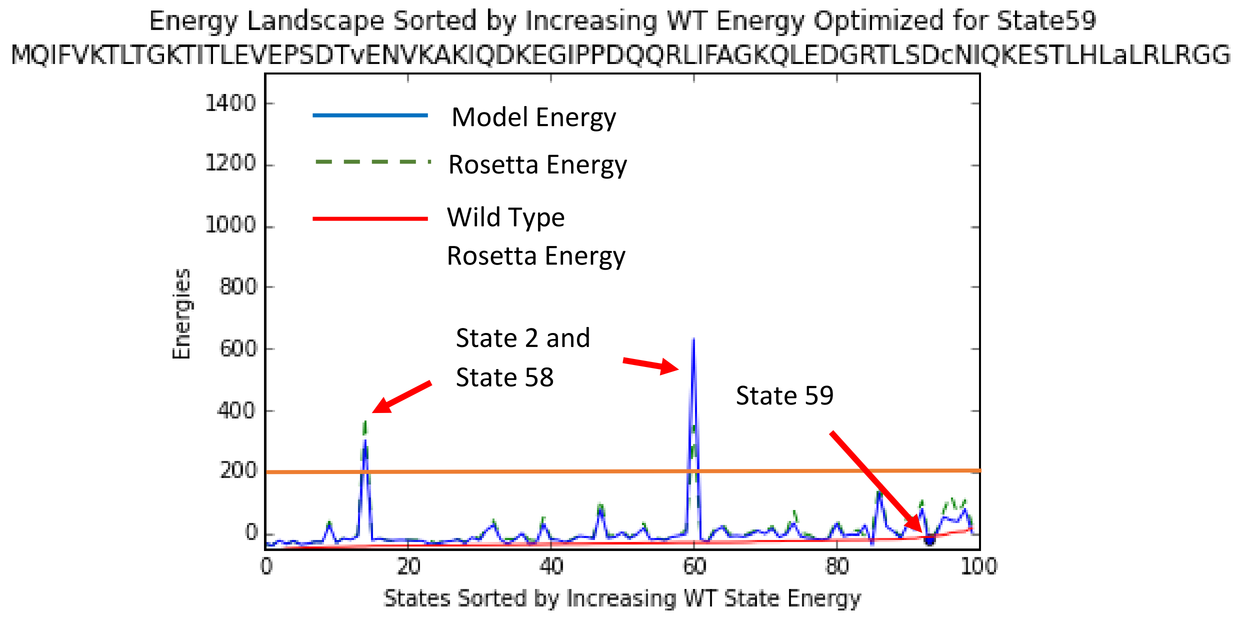


Figure 21: Energy Landscape of a Triple Mutant Optimized for State 59
Energy landscape of specified sequence. There seems to be very low destabilization occurring in this optimization. As only two states have been destabilized above a value of 200 REU. State 59 (optimized state) also seems to have an energy below its wild type energy.

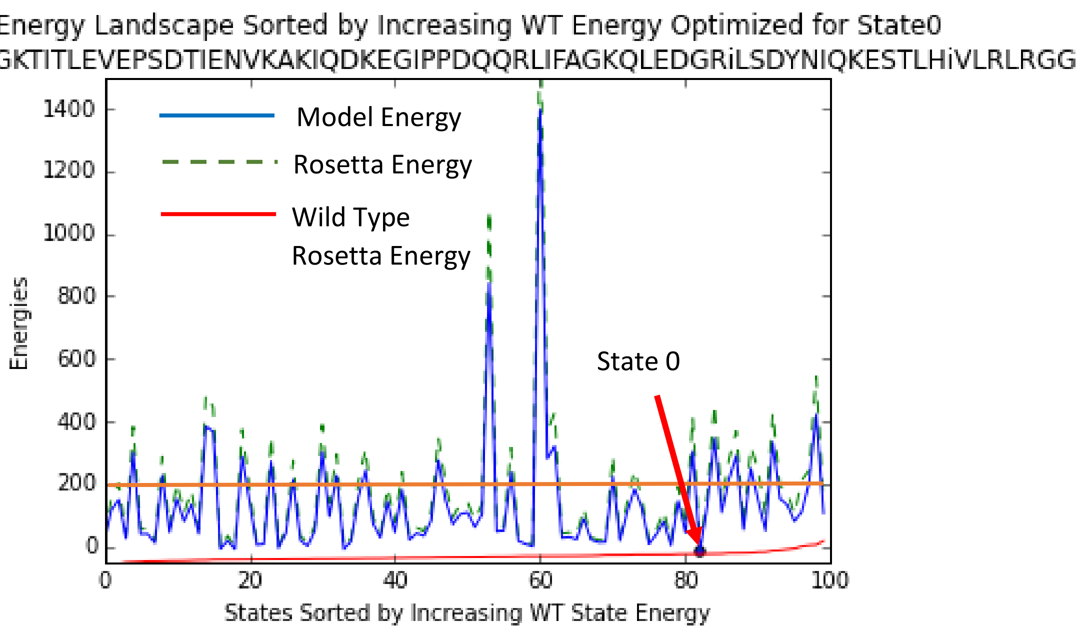


Figure 22: Energy Landscape of a Triple Mutant Optimized for State 0
Energy landscape of specified sequence. There seems to definitely be much more destabilization occurring especially when compared against Figure 21.

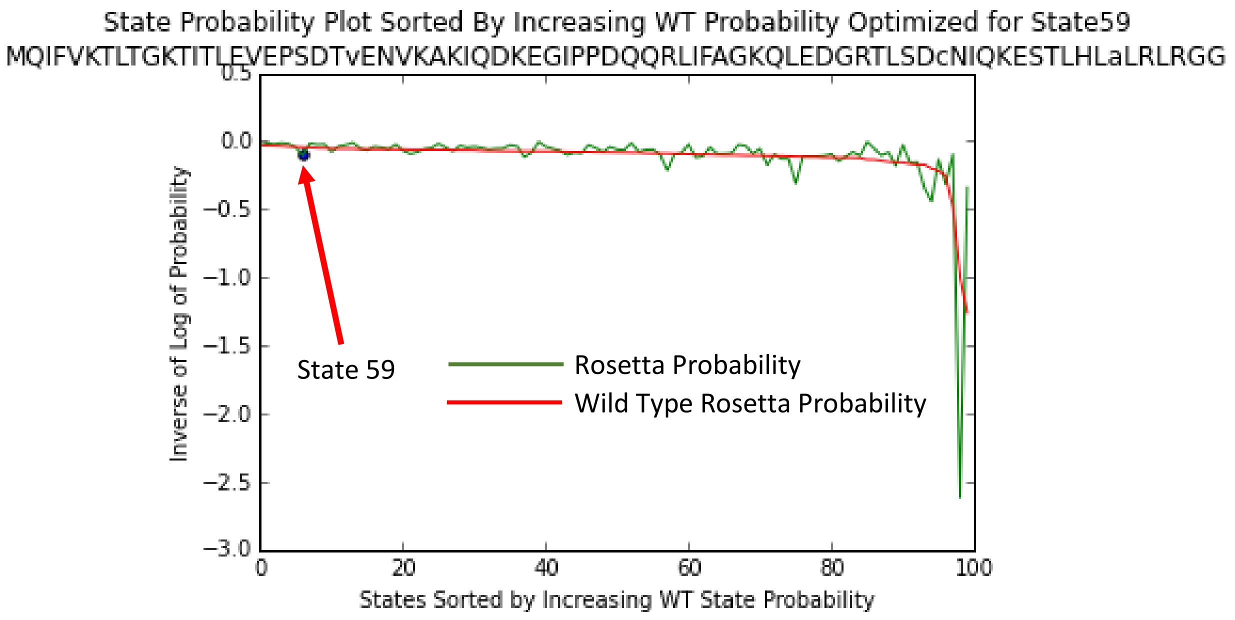


Figure 23: Shifted Probability Plot of a Triple Mutant Optimized for State 59
Shifted Probability Plot of the specified mutant based on Rosetta energy values for each state. We see that our optimized state, State 59, seems to have been stabilized from the wild type sequence at least slightly as compared with other state stabilizations. There is a greater probability on the right side of the figure (State 52). State 59 in this mutation only seems to be slightly more probable when compared to the wild type sequence.

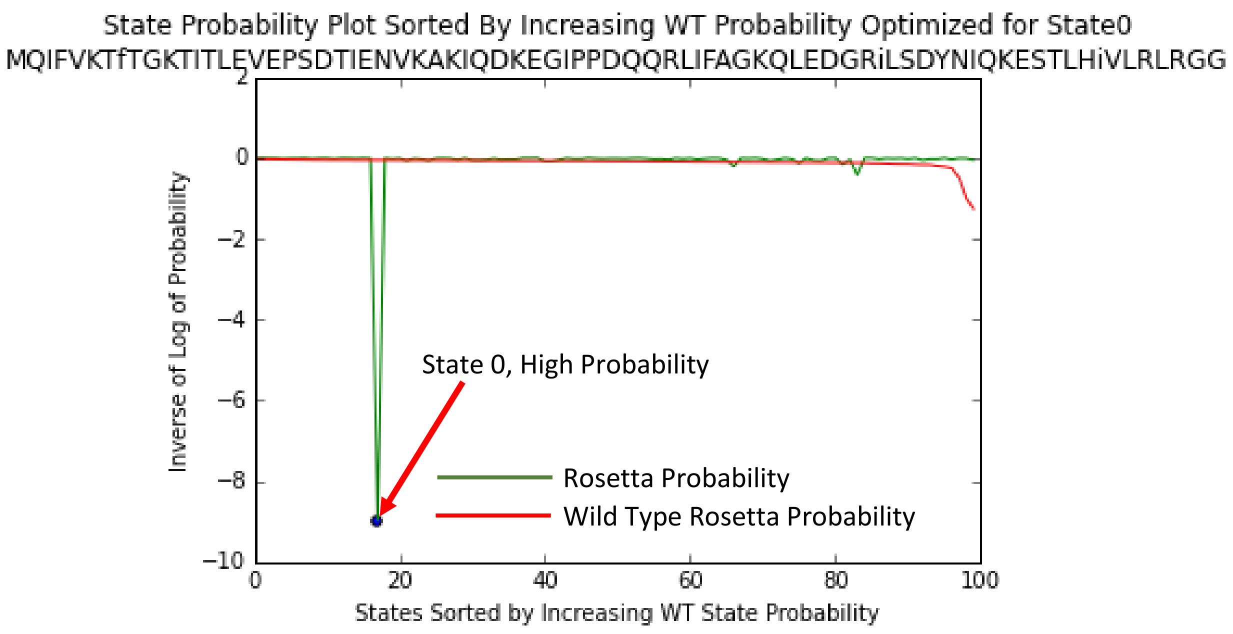
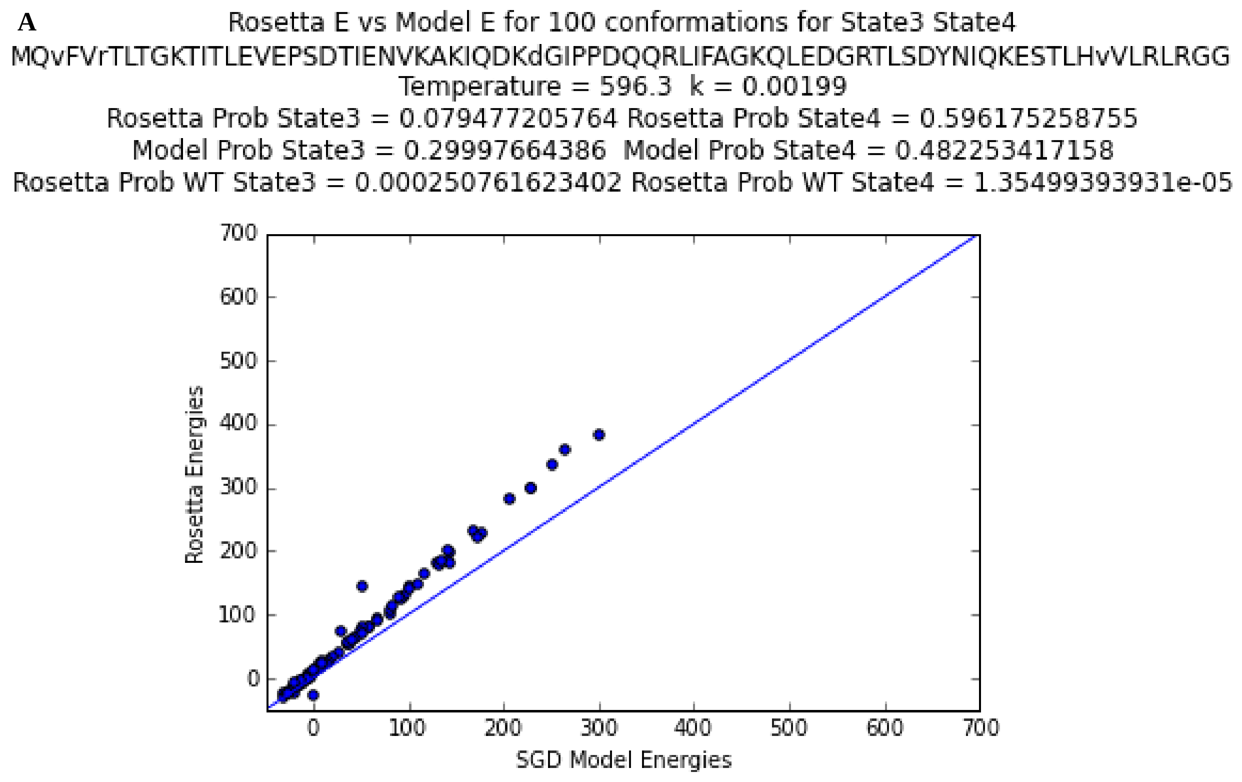


Figure 24: Shifted Probability Plot of a Triple Mutant Optimized for State 0
Shifted Probability Plot of the specified mutant based on Rosetta energy values for each state. When the green line is lower than the wild type red line that means that specific state has a higher probability of existing for the given sequence. We see that our optimized sequence for State 0 seems to have a much higher probability compared with the wild type sequence. Also, it seems to be one of the only states that actually has a higher probability after the optimization.

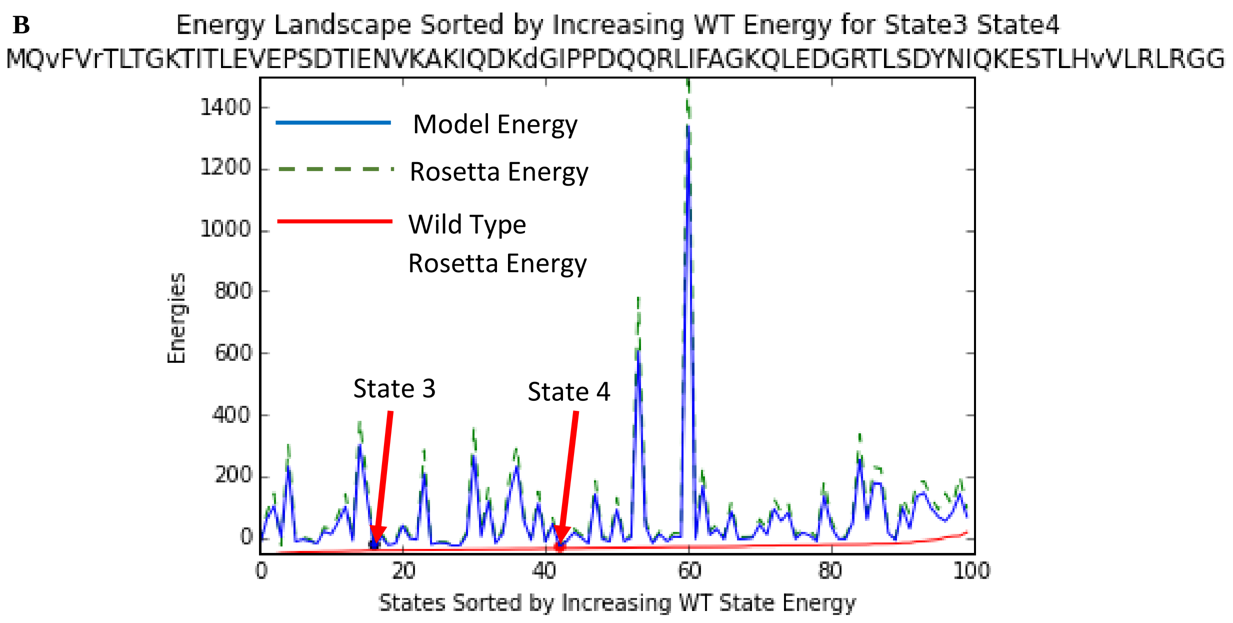
In Figure 21 we showcase a designed energy landscape of a triple mutant stabilized for State 59. We do not see much destabilization across all conformational states. We see typical State 2 destabilization and also some destabilization in State 58. In Figure 22 we showcase a designed energy landscape of a triple mutant optimized for stabilizing State 0. We see the typical peak at State 2 but also see a rather large peak at State 74. Also, looking at the orange line going across at an REU value of 200 we see that the State 0 optimized sequence (Figure 22) has multiple States that are destabilized past an REU of 200 while the State 59 optimized sequence (Figure 21) only has two states past an REU value of 200. We present this comparison to showcase that we can drastically alter designed energy landscapes from the wild type sequence and also that designed energy landscapes can be significantly different from each other depending on which state is being optimized. Optimized probabilities for each state also appear to be successful as seen in Figure 23 for State 59 and Figure 24 for State 0. In Figure 23 we see that State 59 is slightly stabilized as the blue dot is at least somewhat underneath the red wild type inverted probabilities. However, in Figure 24, we see substantial stabilization as compared with all other states for the given sequence. State 0 is definitely the preferred state as it has the highest probability (lowest inverse shifted probability) by far when compared to all other states. Only two other states seem to have a higher probability than the wild type sequence when looking at Figure 24.

Next we compare energy landscapes of dual state optimized sequences. For this optimization, we forced the probability of each selected state towards a value of .5 through minimization of the dual state objective function (as opposed to 1 for single state optimization). Below we present a sequence optimized to stabilize State 3 as well as State 4.

A


Rosetta E vs Model E for 100 conformations for State3 State4
MQvFVrTLTGKTITLEVEPSDTIENVKAKIQDKdGIPPDQQRLIFAGKQLEDGRTLSDYNIQKESTLHvVLRLRGG
Temperature = 596.3 k = 0.00199
Rosetta Prob State3 = 0.079477205764 Rosetta Prob State4 = 0.596175258755
Model Prob State3 = 0.29997664386 Model Prob State4 = 0.482253417158
Rosetta Prob WT State3 = 0.000250761623402 Rosetta Prob WT State4 = 1.35499393931e-05

B



C

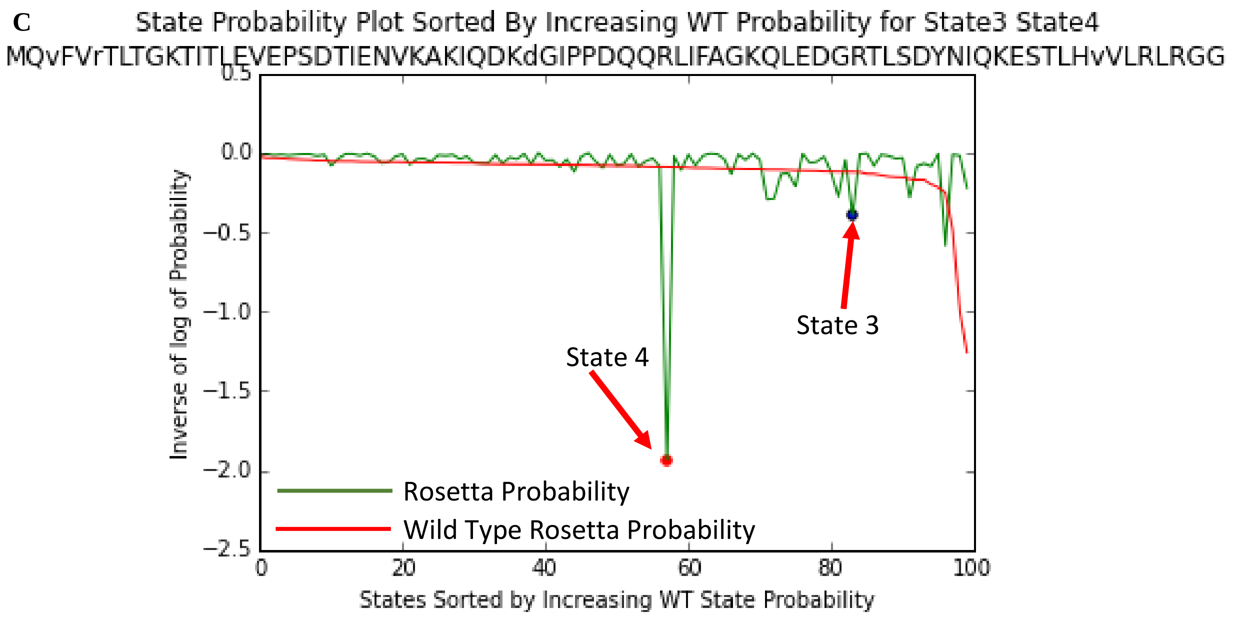
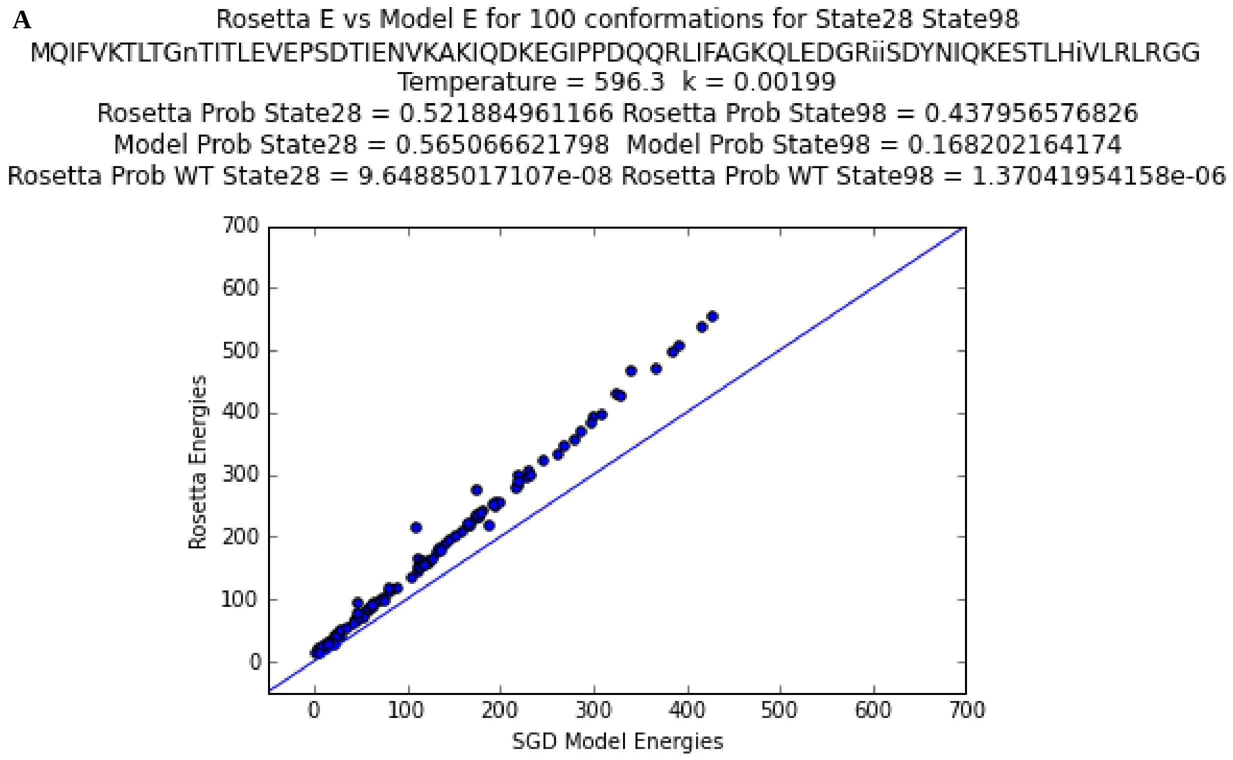


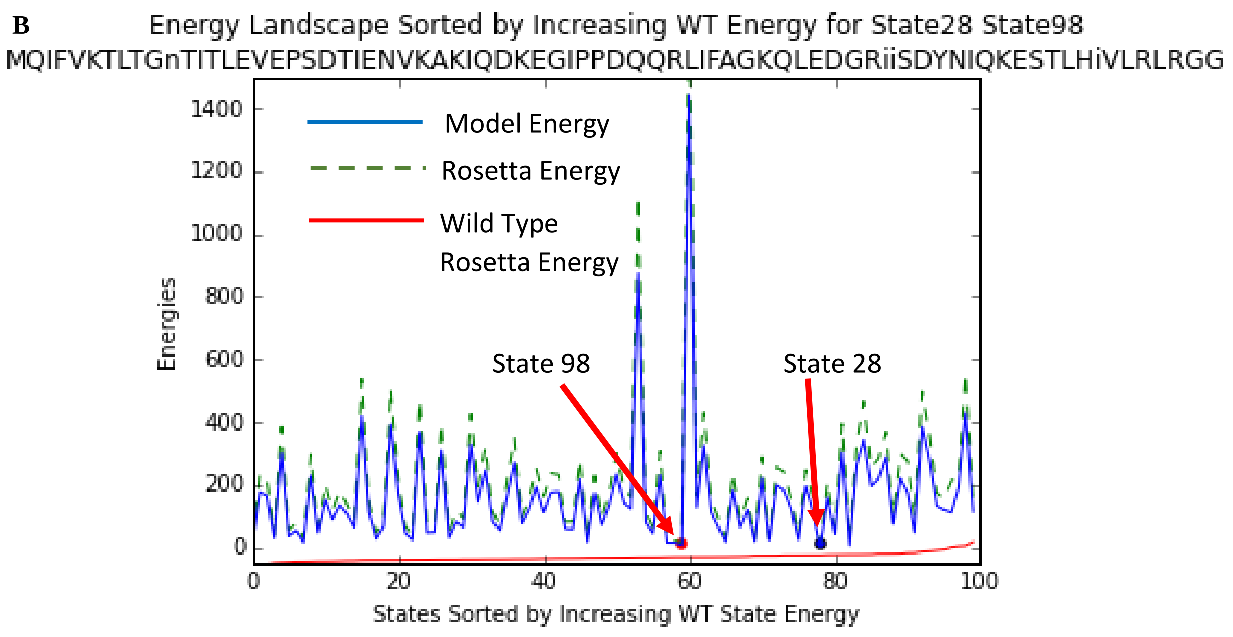
Figure 25: Triple Mutant Optimized Sequence Stabilizing State 3 and State 4
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see good linear agreement especially at low energies. As energy raises our models seem to underestimate energy values (B) Energy landscape of specified sequence. There does not seem to be substantial destabilization although some is definitely present and we see that State 3 and State 4 seem to be optimized at values around zero REU. (C) Shifted Probability Plot of specified sequence. Both optimized states seem to be at least slightly optimized above the associated wild type probabilities. State 4 seems to be substantially more probable while State 3 is at least slightly more probable for the new sequence.

Based on Figure 25-B as well as Figure 25-C and the data it seems that we were successful in stabilizing State 3 and 4. We see the typical destabilization at State 2 and State74. Energy values for both of the optimized states seem to be at or near the wild type energy line as well. We also see good accuracy when comparing the Rosetta energies and model energies for each of the 100 states (Figure 25-A). In Figure 25-C we see that the inverse shifted probability for both states has also been decreased (regular probability increased) from the wild type sequence inverse shifted probability. State 4 is substantially more probable while State 3 is at least slightly more probable. This also makes sense when compared to the data present above Figure 25-A as State 4 has a probability of 0.59 while State 3 only has a probability of 0.079.

A



B



C

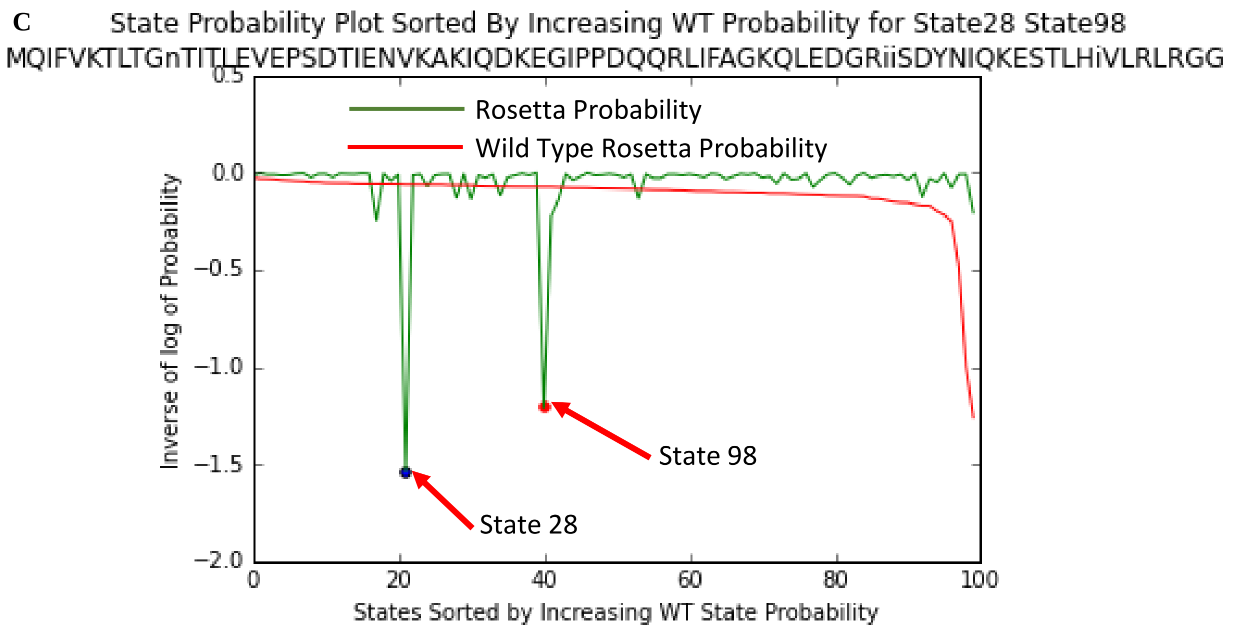
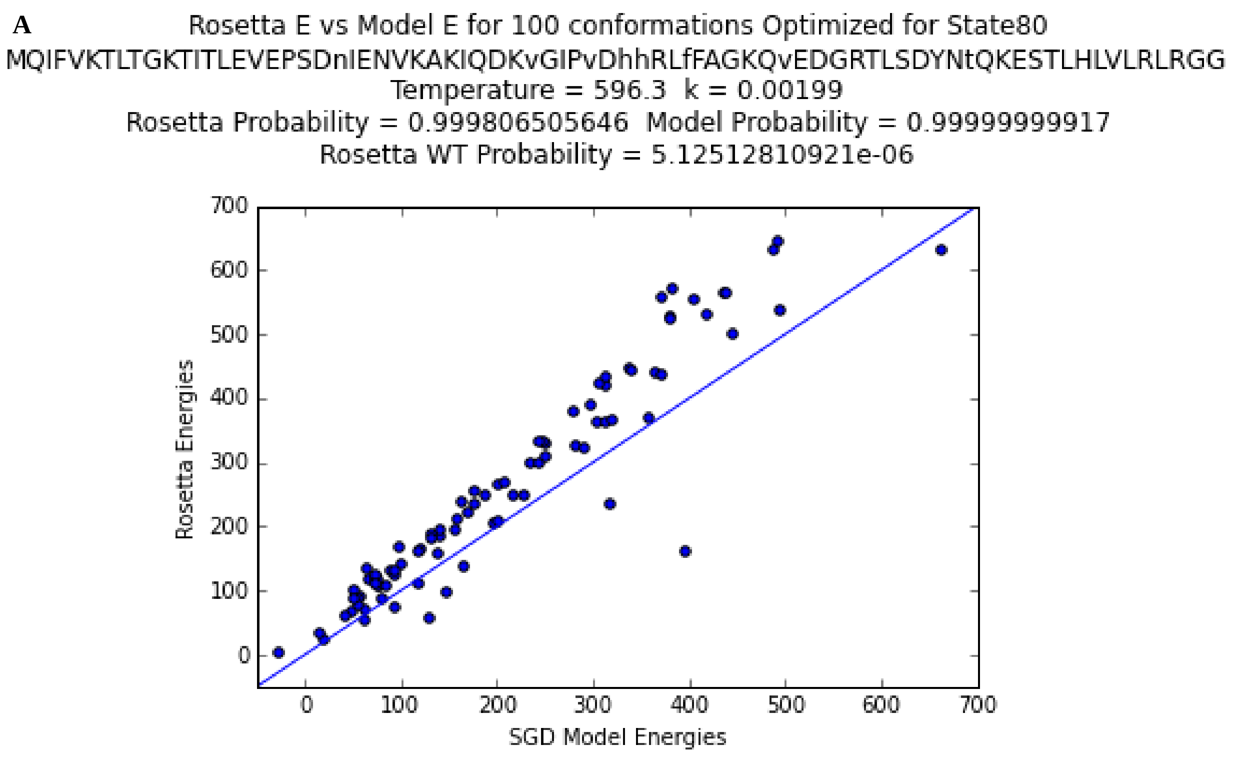


Figure 26: Triple Mutant Optimized Sequence Stabilizing State 28 and State 98
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see good linear agreement especially at low energies. As energy raises our models seem to underestimate energy values (B) Energy landscape of specified sequence. There seems to be substantial destabilization especially at State 2 (typical). Other than that in (B) we see that our models follow the general Rosetta energy trends. (C) Shifted Probability plot for specified sequence. We see that both State 28 and State 98 have been successfully stabilized. Both shifted values are below the wild type shifted probability line while various other states have been destabilized.

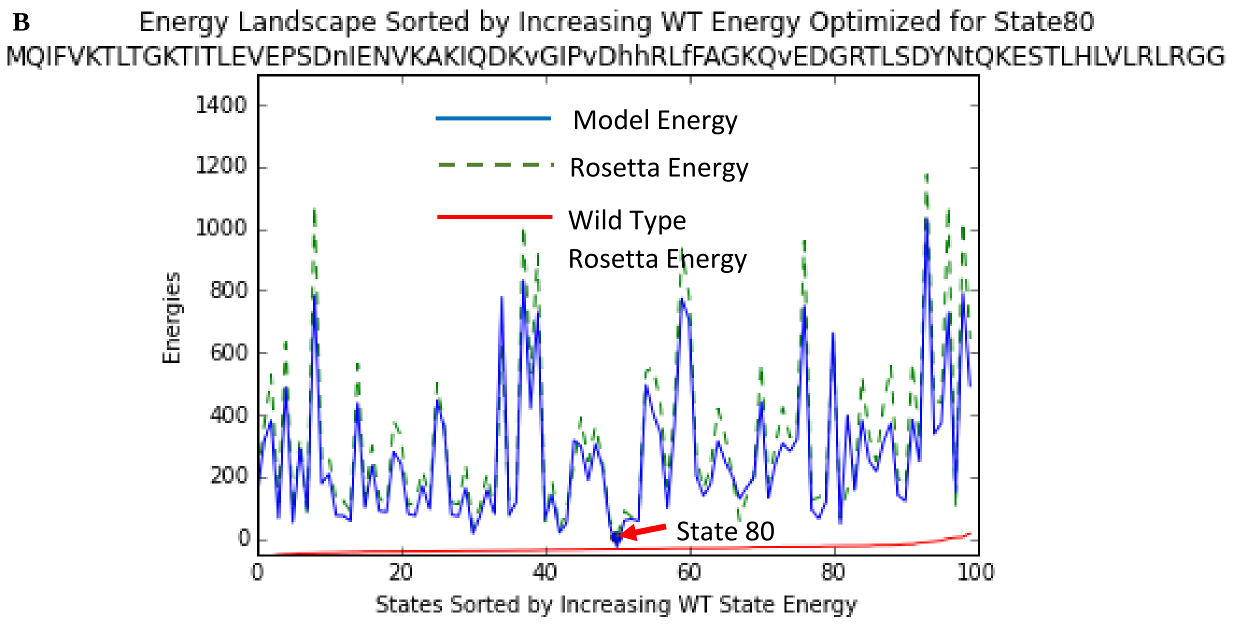
In Figure 26 we present plots for a dual optimized sequence stabilizing State 28 and State 98. We again see the typical destabilization in State 2 but we were successfully able to both target states. Both states are below the shifted wild type probability line in Figure 26-C. When looking at Figures 26-B and C we see that in Figure 26-B all states have been destabilized. Our target states definitely have lower energies when compared to off target states, but even our target states seem to be destabilized from the wild type. However, in Figure 26-C we see that both of our target states have higher probability values (lower shifted values). We believe this is largely due to all other states being substantially destabilized. So, instead of our algorithm successfully stabilizing the target states, it seems that we have destabilized other states and that plays into the relative higher probability of the target states. Ultimately, our energy functions and optimization algorithms tend to hold up in regards to two state optimization though it is difficult to pick out trends in the landscape across optimized states.

Finally, we showcase that we can design sequences with a higher mutation threshold than simply triple mutants. Many more designed sequences for specified conformational states can be generated with any number of allowable mutations and potentially stabilizing any number of states. A larger number of mutations from the wild type sequence will likely correspond with less accurate predictions from our models. This is likely due to the higher energy sequences associated with larger numbers of mutations from the wild type sequence. Higher energy sequences result in our models being less accurate given that our models were trained on quadruple mutant data (with relatively low energies). We showcase a sequence optimized for State 80 (Figure 27). The optimized sequence was allowed to have 8 mutations from the wild type sequence.

A


Rosetta E vs Model E for 100 conformations Optimized for State80
MQIFVKTLTGKTITLEVEPSDnIENVKAKIQDKvGIPvDhhRLfFAGKQvEDGRTLSDYNtQKESTLHLVLRLRGG
Temperature = 596.3 k = 0.00199
Rosetta Probability = 0.999806505646 Model Probability = 0.99999999917
Rosetta WT Probability = 5.12512810921e-06
Rosetta Energies
SGD Model Energies
0
100
200
300
400
500
600
700

B


Energy Landscape Sorted by Increasing WT Energy Optimized for State80
MQIFVKTLTGKTITLEVEPSDnIENVKAKIQDKvGIPvDhhRLfFAGKQvEDGRTLSDYNtQKESTLHLVLRLRGG
Model Energy
Rosetta Energy
Wild Type Rosetta Energy
State 80
Energies
States Sorted by Increasing WT State Energy

C

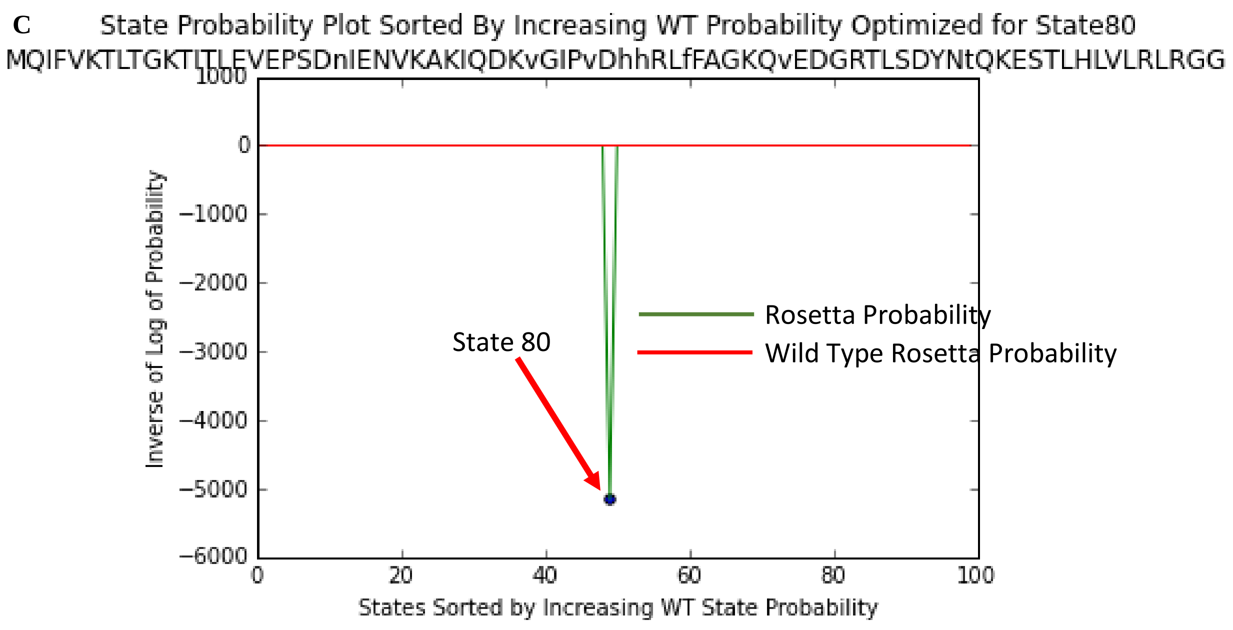
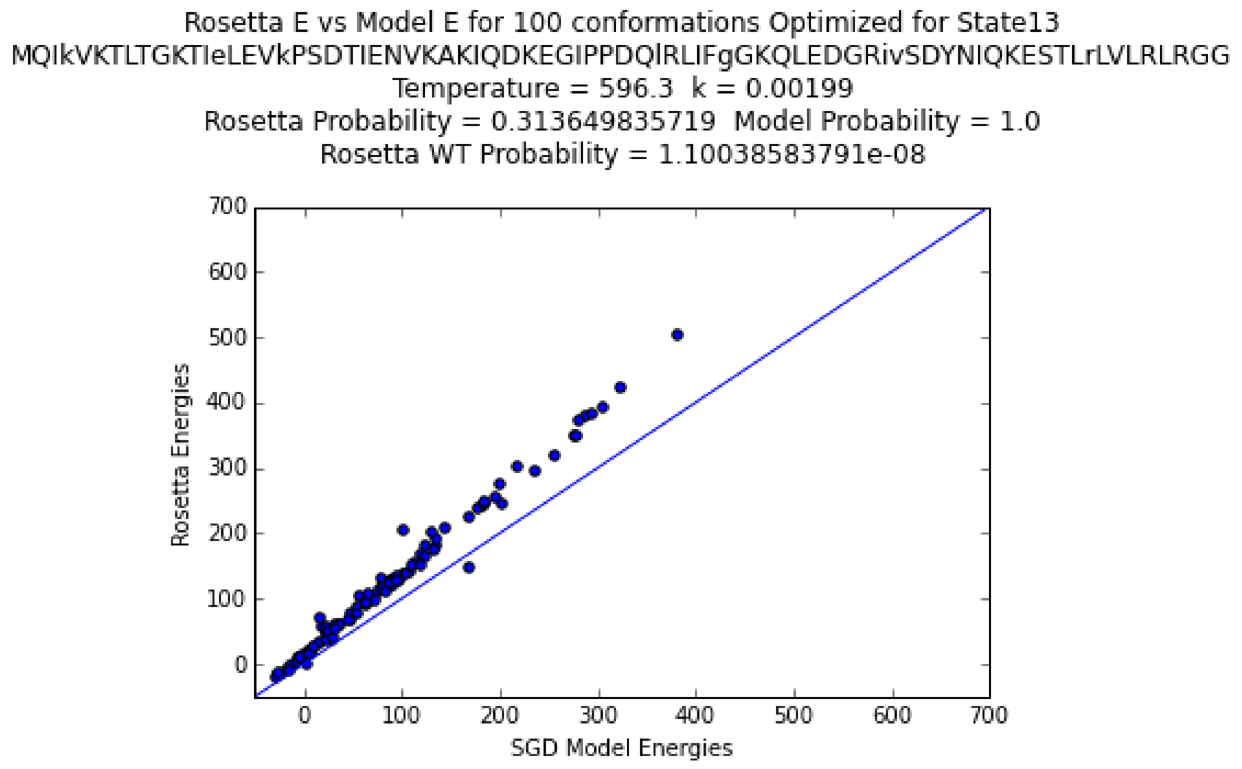


Figure 27: Eight Mutation Optimized Sequence Stabilizing State 80
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see decent linear agreement especially at low energies but there is definitely more deviations when compared with the triple mutant optimized sequences. As energy raises our models seem to underestimate energy values although at a few energies our models overestimate energy values (B) Energy landscape of specified sequence. We do see good agreement in the energy landscape however, at high energies we do see some disagreements. There seems to be substantial destabilization, likely due to their being 8 mutations from the wild type sequence. (C) Shifted Probability plot for specified sequence. Many of the states are substantially destabilized. We see that our optimized state, State 80 is the only visible state on the plot. It is stabilized below the shifted wild type probability. However, this is likely due to the vast majority of other states being destabilized.

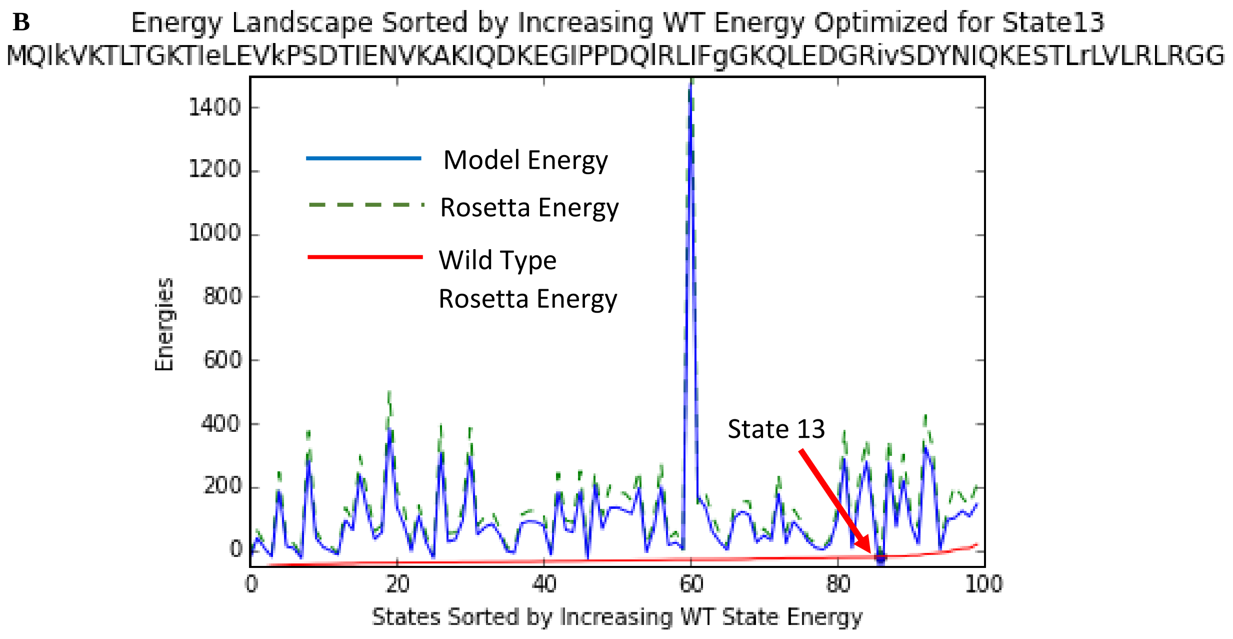
In Figure 27-A and Figure 27-B we see that our models are less accurate for this sequence with higher mutations. However, we still get correlated energy values that trend along the line of slope 1. This higher inaccuracy of our models is likely attributed to the higher energies associated with this sequence at various conformational states. Given the higher number of mutations we see significantly more destabilization from state to state. This makes sense as more mutations from the wild type sequence would correspond with higher overall energy values. On the energy landscape plot we can also clearly see that State 80 has been stabilized while all other states have been substantially destabilized as State 80 is the lowest value in the energy landscape.

In Figure 27-C, we see that the probability for nearly all of the states is highly unlikely and the only likely state is our optimized state 80. Essentially, the only state that is likely to exist is our optimized state of State 80. We also see that State 80 is a good amount below the shifted and inverted wild type probability line in Figure 27-C. This extremely high probability we see is likely attributed to the large amount of destabilization in nearly all of the other states (energy landscape plot destabilization). So in relative terms, State 80 is the only probable state given the vast destabilization of most of the other states. The energy value for State 80 (4.1 REU) is above the wild type value but in Figure 27-B it is the only state with an energy value relatively close to the wild type energy level.

A



B



C

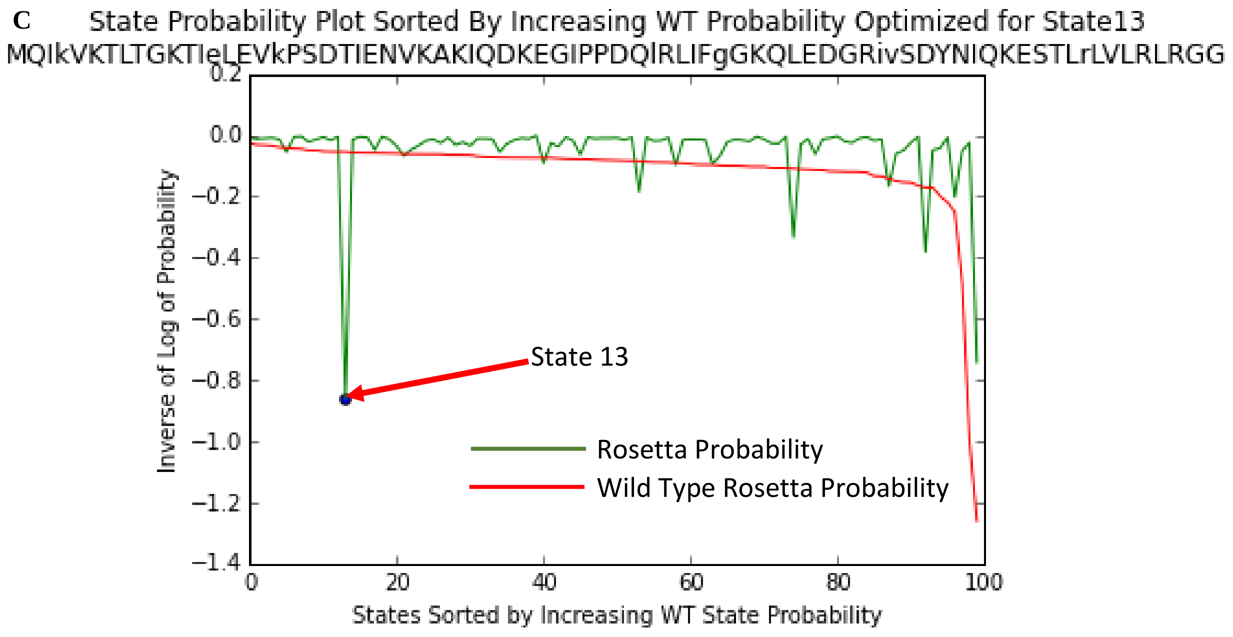


Figure 28: Eight Mutation Optimized Sequence Stabilizing State 13
(A) Rosetta Energy vs No Pair No Weight Model Energy for all 100 conformations for specified sequence. We see good linear agreement especially at low energies. As energy raises our models seem to underestimate energy (B) Energy landscape of specified sequence. The energy landscape plot again shows our model energies following the Rosetta energy value trends. (C) Shifted Probability plot for specified sequence. The shifted probability plot has a few states from the mutation that are more probable when compared to the wild type sequence. We see that our optimized state, State 13, has the highest probability when compared to all other states in the mutated sequence.

In Figure 28 we showcase an 8 mutation designed sequence for State 13. Again, we see general linear agreement when comparing our model energies and the Rosetta energies (Figure 28-A). This sequence optimization (State 13) appears more accurate as compared with the previous optimized sequence (State 80) and this is largely due to the lower energy values present in most of the conformational states across the energy landscape. Essentially, there is less destabilization occurring across all states. This is also presented in the Shifted Probability Plot (Figure 28-C) as we can actually see other states present in the plot. We see that State 13 is the most probable state in the landscape while four other states are also more probable when we compare wild type sequence with the new mutation.

Ultimately, we could showcase any sequence optimization that selects to stabilize either one state or a pair of states for any allowable number of mutations. As we increase the number of mutations we see less agreement with Rosetta energy values and our models. Also, with increased mutation threshold for designed sequences we definitely see substantially more destabilization from state to state including the optimized state. This is due to the higher energies associated with higher mutation numbers. The highly probable states in this high mutation case seems to largely be stabilized by the destabilization of many of the other states. Also, at this high mutation threshold we see that the energy value of our stabilized state usually increases relative to the wild type sequence. Running longer optimizations over a larger number of sequences could potentially improve the energy value of the optimized state. In the triple mutation case we sometimes get lower energy values for the optimized state/states, if not they are generally pretty close to the wild type values. Among the optimized sequences presented (3 or 8 mutations per sequence) we show across all states that our models have generally good performance and agreement with Rosetta values. Although our models tend to underestimate some of the high energy values we believe this is not extremely relevant as these destabilized states would be seen as off target states that are not as important. We see that our models follow the general energetic landscape trends when compared with Rosetta energy values. Therefore, as long as our models can stay accurate for the optimized state at low energy levels then we generally see accurate results as a whole.

Conclusions

Potential Problems and Improvements

Our models are able to provide relatively good accuracy during optimization; however, there are some issues with model accuracy when optimizing certain states. Some of our models tend to underestimate specific sequences that have low energy scores (from Rosetta). This provides an energy score from our models that seems extremely ideal but it is an underestimated value from the true Rosetta energy value. This is a problem during optimization because it comes up as a false positive for an optimal (maximum) probability. This generally happens with a high number of mutations from the wild type sequence during optimization. There are potentially ways to bypass or fix these inaccuracies. First, the most obvious fix is we could simply only optimize for lower number of mutations such as triple or quadruple mutants. Secondly, we could find inaccurate states by running one or two optimizations per state and looking for designed sequences that are inaccurate compared with the true Rosetta energy values. We could then go back and re-train weighted models for inaccurate states with their own specifically tailored weight distribution based on the energy distribution of said state. This could provide more accurate results for low energy sequences which could provide better accuracy during optimization. Thirdly, we could select to run optimizations with models that include the 4 angstrom distance two-body parameters. This change was seen to slightly improve model accuracy and it should provide more accurate results across all states. At low mutation numbers these changes do not seem to be a necessity. However, they could be implemented for more accurate results as well.

Expansion of Computational Method

This computational method could certainly be expanded to larger protein systems and conformational spaces. This is largely due to our online learning approach for model training which makes our approach not limited by system memory. Therefore, the biggest issue is generating accurate sequence-energy models for each state. Further expansion of this method could also be to implement more complex design objectives. The simplistic design objectives we use could definitely be expanded upon. For example, one could optimize for more than two states up to as many states as one would want. Also, one could include that energy for the stabilized state(s) has to be lower than the previous iterations energy or the wild type energy. This could ensure more stabilization of selected state(s). Other design objectives could include the classification of specific states as off-target, target, and transition states. With more biological information about specific conformations one could make these decisions. Essentially, specific subfunctions of a protein or enzyme could be mapped to specific conformational states. For example, substrate binding and product release could have specified conformations mapped to them from MSM's. Then these conformations associated with each subfunction can be optimized for and the energy transition between them could potentially be controlled through use of a transition conformation. One could assign certain energetic constraints during optimization for specified on-target, off-target, and transition conformations. This would ultimately control the energetic transition between conformational states. If one wanted to attempt to improve stability of a specific subfunction then the associated state could be selected as target and could be optimized for. On the contrary if this function wanted to be turned off one could label the state as off-target and select to stabilize other states. Sequences would then be optimized to selectively ‘turn off’ certain functions or destabilize certain states related to specific functions or to stabilize states related to these functions. Larger areas of sequence space could also be searched through using methods such as branch and bound optimization. We only cycle through around one million sequences for each optimized sequence. Using a branch and bound optimization method would allow searching through a much larger space of sequences.

Model Accuracy Testing

Further accuracy testing of designed sequences could also be done. For a designed sequence we show that we can generate an energy landscape plot through scoring the mutated sequence with our models as well as scoring the mutated sequence with an energy function (Rosetta and OpenMM). This is a good comparison for design but a more valid accuracy testing technique could be to compare our landscape plots as well as our shifted probability plots with an energy landscape plot and a shifted probability plot from an MSM of the designed mutation. We could score the energies of this new MSM and compare with our designed energy landscape plots and probability plots to see if the new MSM has energy values from new states that map to the energy values of our models energy landscape plots.

Accurately Designed Energy Landscapes

Through this new protein engineering method we showcase the ability to completely and accurately redesign energy landscapes for ubiquitin mutants up to 8 mutations away from the wild type sequence. This is largely due to the online learning approach we take allowing us to train all computational models at the same time. We were able to generate accurate models for all 100 separate structures. Putting all of these models together during the design phase, our models maintained good enough accuracy at low energy to stabilize either one target state or a pair of target states while destabilizing other off-target states.

References

1. Burcu Turanli-Yildiz, C.A., Z. Petek Cakar, *Protein Engineering Methods and Applications*. 2012: InTech.
2. Kuhlman, B., et al., *Design of a novel globular protein fold with atomic-level accuracy.* Science, 2003. **302**(5649): p. 1364-8.
3. Jiang, L., et al., *De novo computational design of retro-aldol enzymes.* Science, 2008. **319**(5868): p. 1387-91.
4. Lassila, J.K., *Conformational diversity and computational enzyme design.* Curr Opin Chem Biol, 2010. **14**(5): p. 676-82.
5. Baker, D., *An exciting but challenging road ahead for computational enzyme design.* Protein Sci, 2010. **19**(10): p. 1817-9.
6. van den Berg, B.A., et al., *Protein redesign by learning from data.* Protein Eng Des Sel, 2014. **27**(9): p. 281-8.
7. Yue, K. and K.A. Dill, *Inverse protein folding problem: designing polymer sequences.* Proc Natl Acad Sci U S A, 1992. **89**(9): p. 4163-7.
8. Boehr, D.D., R. Nussinov, and P.E. Wright, *The role of dynamic conformational ensembles in biomolecular recognition.* Nature chemical biology, 2009. **5**(11): p. 789-796.
9. Henzler-Wildman, K. and D. Kern, *Dynamic personalities of proteins.* Nature, 2007. **450**(7172): p. 964-972.
10. Babor, M., D.J. Mandell, and T. Kortemme, *Assessment of flexible backbone protein design methods for sequence library prediction in the therapeutic antibody Herceptin-HER2 interface.* Protein Sci, 2011. **20**(6): p. 1082-9.

11. Murphy, G.S., et al., *Increasing sequence diversity with flexible backbone protein design: the complete redesign of a protein hydrophobic core.* Structure, 2012. **20**(6): p. 1086-96.

12. Davey, J.A. and R.A. Chica, *Multistate approaches in computational protein design.* Protein Sci, 2012. **21**(9): p. 1241-52.

13. Zhou, F., et al., *Coarse-graining protein energetics in sequence variables.* Phys Rev Lett, 2005. **95**(14): p. 148103.

14. Vijay-Kumar, S., C.E. Bugg, and W.J. Cook, *Structure of ubiquitin refined at 1.8 A resolution.* J Mol Biol, 1987. **194**(3): p. 531-44.

15. Kiss, G., et al., *Computational Enzyme Design.* Angewandte Chemie International Edition, 2013. **52**(22): p. 5700-5725.

16. Bowie, J.U., R. Luthy, and D. Eisenberg, *A METHOD TO IDENTIFY PROTEIN SEQUENCES THAT FOLD INTO A KNOWN 3-DIMENSIONAL STRUCTURE.* Science, 1991. **253**(5016): p. 164-170.

17. Li, X., Z. Zhang, and J. Song, *Computational enzyme design approaches with significant biological outcomes: progress and challenges.* Computational and Structural Biotechnology Journal, 2012. **2**: p. e201209007.

18. Lutz, S., *Beyond directed evolution - semi-rational protein engineering and design.* Current opinion in biotechnology, 2010. **21**(6): p. 734-743.

19. Liu, Y. and B. Kuhlman, *RosettaDesign server for protein design.* Nucleic Acids Research, 2006. **34**(Web Server issue): p. W235-W238.

20. Rohl, C.A., et al., *Protein Structure Prediction Using Rosetta*, in *Methods in Enzymology*. 2004, Academic Press. p. 66-93.

21. Lazaridis, T. and M. Karplus, *Effective energy function for proteins in solution.* Proteins: Structure, Function, and Bioinformatics, 1999. **35**(2): p. 133-152.

22. Kortemme, T., A.V. Morozov, and D. Baker, *An Orientation-dependent Hydrogen Bonding Potential Improves Prediction of Specificity and Structure for Proteins and Protein–Protein Complexes.* Journal of Molecular Biology, 2003. **326**(4): p. 1239-1259.

23. Dantas, G., et al., *A large scale test of computational protein design: Folding and stability of nine completely redesigned globular proteins.* Journal of Molecular Biology, 2003. **332**(2): p. 449-460.

24. Gerstein, M., A.M. Lesk, and C. Chothia, *Structural Mechanisms for Domain Movements in Proteins.* Biochemistry, 1994. **33**(22): p. 6739-6749.

25. Pande, V.S., K. Beauchamp, and G.R. Bowman, *Everything you wanted to know about Markov State Models but were afraid to ask.* Methods, 2010. **52**(1): p. 99-105.

26. Bowman, G.R., et al., *Progress and challenges in the automated construction of Markov state models for full protein systems.* J Chem Phys, 2009. **131**(12): p. 124101.

27. Voelz, V.A., et al., *Molecular simulation of ab initio protein folding for a millisecond folder NTL9(1-39).* J Am Chem Soc, 2010. **132**(5): p. 1526-8.

28. Bowman, G.R., *Improved coarse-graining of Markov state models via explicit consideration of statistical uncertainty.* J Chem Phys, 2012. **137**(13): p. 134111.

29. Wipf, A., *Mean Field Approximation*, in *Statistical Approach to Quantum Field Theory: An Introduction*. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 119-148.

30. Apgar, J.R., et al., *Cluster expansion models for flexible-backbone protein energetics.* J Comput Chem, 2009. **30**(15): p. 2402-13.

31. Grigoryan, G., et al., *Ultra-fast evaluation of protein energies directly from sequence.* PLoS Comput Biol, 2006. **2**(6): p. e63.

32. Hart, K.M., et al., *Modelling proteins' hidden conformations to predict antibiotic resistance.* Nat Commun, 2016. **7**: p. 12965.

33. Van Der Spoel, D., et al., *GROMACS: fast, flexible, and free.* J Comput Chem, 2005. **26**(16): p. 1701-18.

34. Jorgensen, W.L., et al., *Comparison of simple potential functions for simulating liquid water.* The Journal of Chemical Physics, 1983. **79**(2): p. 926-935.

35. Bowman, G.R., X. Huang, and V.S. Pande, *Using generalized ensemble simulations and Markov state models to identify conformational states.* Methods, 2009. **49**(2): p. 197-201.

36. Kaufmann, K.W., et al., *Practically useful: what the Rosetta protein modeling suite can do for you.* Biochemistry, 2010. **49**(14): p. 2987-98.

37. Eastman, P., et al., *OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation.* J Chem Theory Comput, 2013. **9**(1): p. 461-469.

38. Reinsel, G.C. and R.P. Velu, *Multivariate Linear Regression*, in *Multivariate Reduced-Rank Regression: Theory and Applications*. 1998, Springer New York: New York, NY. p. 1-14.

39. Bottou, L., *Large-Scale Machine Learning with Stochastic Gradient Descent*, in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, Y. Lechevallier and G. Saporta, Editors. 2010, Physica-Verlag HD: Heidelberg. p. 177-186.

40. Bottou, L., *On-line Learning and Stochastic Approximations*, in *On-Line Learning in Neural Networks*, D. Saad, Editor. 1999, Cambridge University Press: Cambridge. p. 9-42.

41. Draper, N.R. and H. Smith, *Applied Regression Analysis*. 1981: Wiley.

42. Oliphant, T.E., *Python for Scientific Computing.* Computing in Science & Engineering, 2007. **9**(3): p. 10-20.

43. Rossum, G.v. and F.L. Drake, *The Python Language Reference Manual*. 2011: Network Theory Ltd. 150.

44. Walt, S.v.d., S.C. Colbert, and G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation.* Computing in Science & Engineering, 2011. **13**(2): p. 22-30.

45. Hunter, J.D., *Matplotlib: A 2D Graphics Environment.* Computing in Science and Engg., 2007. **9**(3): p. 90-95.

46. Thain, D., T. Tannenbaum, and M. Livny, *Distributed computing in practice: the Condor experience: Research Articles.* Concurr. Comput. : Pract. Exper., 2005. **17**(2-4): p. 323-356.

47. Edgar, R.C., *MUSCLE: multiple sequence alignment with high accuracy and high throughput.* Nucleic Acids Res, 2004. **32**(5): p. 1792-7.

48. Jenuth, J.P., *The NCBI. Publicly available tools and resources on the Web.* Methods Mol Biol, 2000. **132**: p. 301-12.

49. Landau, L.D., E.M. Lifshit͡s, and L.P. Pitaevskiĭ, *Statistical Physics*. 1980: 世界图书出版公司.

50. Kellogg, E.H., A. Leaver-Fay, and D. Baker, *Role of conformational sampling in computing mutation-induced changes in protein structure and stability*. Proteins, 2011. **79**(3): p. 830-8.

51. Bottou, L., *Stochastic Gradient Descent Tricks*, in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G.B. Orr, and K.-R. Müller, Editors. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 421-436.

52. Schrodinger, LLC, *The PyMOL Molecular Graphics System, Version 1.8*. 2015.