3.3 Protein Design Algorithm

To investigate the impact of van der Waals packing on membrane protein association, I opted for a high-throughput design approach. I created a sequence search algorithm that can design thousands of homodimer MP structures using MSL v. 1.1, an open source C++ library that is freely available at http://msl-libraries.org (Kulp et al., 2012). I coupled this algorithm with a structural backbone refinement program also built in MSL. Below, I detail the algorithm alongside experiments and tests that aided in its development.

A diagram of a protein structure

Description automatically generated with medium confidence3.3.1 Analysis of membrane protein PDBs

**Figure 3.1 MP helix-helix density distribution. A)** The geometric terms for homodimer proteins: Distance (x-shift, d), crossing angle (θ), axial rotation (ω), vertical shift (z-shift, Z). **B)** Helix-helix interactions extracted from the PDB in September 2019. Plotted against the angle and distance of each interaction, and density map defined using kernel density estimation.

To computationally design homodimers, I first extracted backbone geometries from all unique MPs found in the Orientations of Membrane Proteins (OPM) database (Lomize et al., 2006). To ensure that I don’t extract redundant helical pairs, the MP structures from OPM were trimmed by sequence similarity. Only unique structures with less than 30% sequence similarity were analyzed (Steinegger & Söding, 2017). We then developed a program in MSL that reads in a structure and identifies helical segments. This program extracts helical segments found within the membrane by measuring the Phi-Psi angles of quadruplets of Cα carbons. Any helical segments composed of at least 13 AAs in length are extracted as individual helices, and the distance is measured between Cα carbons on each unique helical pair. Any two helices with at least 3 Cα carbons within 9Å of each other are extracted as an individual helical pair. I then extracted two parameters, the distance (x-shift, **d**) and the angle (crossing angle, **θ**) (Figure 3.1A), which were plotted as a scatterplot and analyzed using kernel density estimation (Figure 3.1B).

The density plot identifies the most common interaction motifs for dimeric proteins. We expect regions of high density to correlate with designability: By applying my design algorithm to the geometries most often found in nature, they are more likely to successfully interact. First, it’s important to define how these helical geometries are commonly referred to in scientific literature. When viewing dimers from the perspective on the right side of Figure 3.1A, the helix closer to the eye can denote a naming characteristic. In this image, the helix in front is pointing up and to the right. We refer to these dimers with a negative crossing angle as right-handed. The opposite is true for positive crossing angles, where the helix in front is pointing up and to the left. We refer to these dimers as left-handed.

There are three high density regions present in the membrane protein helix-helix dataset. The first design region is present in the left-handed region. Helical pairs interact frequently in the range between 8.5 to 10 Å of interhelical distance and 20 to 40° of crossing angle. Because there is only a single patch of high density, we refer to this region as the left-handed design region (Left). The other two high density regions are found in the right-handed region. Helical pairs are found at a much broader range of distances from 6.5 to over 10 Å of interhelical distance. The region with the most density is found between 7.75 to 9.5 Å interhelical distance and -30 to -60° crossing angle. We refer to this as the right-handed design region (Right). Finally, the third region corresponds to a known dimerization motif called GASright, which is characterized by short interhelical distance (6.5-7.5 Å) and crossing angles of -25 to -55°. The GASright ­is a well characterized and known to be stabilized by a combination of van der Waals packing and the formation of interhelical weak hydrogen bonds between helices (Anderson et al., 2017; Mueller et al., 2014). I decided to design this region as a control, allowing me to compare the stability A black and white grid

Description automatically generatedbetween proteins stabilized solely by van der Waals packing and GASright.

**Figure 3.2 Template Geometry Grid.** To determine the axial rotations and z-shifts that would be favorable for protein design, I created a grid of randomized geometries that were assessed for possible clashes at the dimer interface.

To design homodimer proteins, there are two additional geometric features: the rotation of the helix (axial rotation, **ω**) and the vertical shift in the membrane (z-shift, **Z**) (Figure 3.1A). To determine these features for the corresponding angles and distances, I created a grid of template geometries for each design region (Figure 3.2). I used MSL to place poly-Ala sequences at each geometry and assessed for clashing at the interface by measuring the van der Waals energy. If the structure is clashing with the small AA Ala at the interface, then the structure is less likely to be designable as it would not be able to accommodate larger AAs. Any structures that corresponded to an energy less than 100 were saved, allowing some leeway for potential clashes in each design region that could be mitigated with backbone refinement. I plotted the saved geometries on density maps and extracted the ranges of axial rotations and z-shifts **(S1, S2, and S3)**. Finally, I generated 1000s of geometries for each design region, where the angle and distance are chosen from the membrane protein density map, and the rotation and z-shift are chosen from the identified ranges where clashing did not occur in poly-Ala sequences. These geometries were used as input backbone templates for protein design.

A close-up of a pie chart

Description automatically generated3.3.2 Choosing amino acids for membrane protein design

**Figure 3.3 Membrane Protein AA Percent Composition. A)** Percent composition of TM-helical regions for all sequences and consensus sequences (Liu et al., 2002). **B)** Re-distributed percent AA composition of only AAs chosen for design, calculated from nonredundant TM-helical pair sequences.

Liu et al. 2002 identified the percent composition of AAs as found in all TM-helical regions in membrane protein sequences (Figure 3.3A). Inspired by the previous literature, I determined the composition of AAs found in my nonredundant extracted helical pair dataset. Since I aim to study the effect of sidechain packing on association, I chose only to design with a subset of AAs to decrease the potential for association by alternate forces at the interface. To prevent the formation of disulfide bridges, the two sulfur containing AAs (Cys and Met) were removed (**cite**). AAs with the potential to form charged interactions (Lys and Arg) were also excluded (**cite**). Histidine, often forming cation-π interactions, was excluded. AAs that often form hydrogen bonds (Asp, Glu, Asn, Gln) were removed, with the exclusion of Ser, Tyr, and Thr due to how frequently we found them in our TM helical pairs. Finally, Pro, which is known to form kinks in helices, was excluded from the design pool. This left me with 10 AAs to use for protein design (Figure 3.3B). Each of these AAs was allowed to be designed along the interface during the sequence search described in section 3.3.5.

A screenshot of a cell phone

Description automatically generated3.3.3 Defining the Interface

**Figure 3.4 Solvent Accessible Surface Area (SASA).** Side view and top-down view of helical dimers. SASA is determined by the amount of area on both helices that is not in contact. Can be thought of similarly to rolling a ball along the surface of the protein structure: The ball can only access parts of the protein not in contact, or outside of the X (the most buried region of the dimer interface).

To reduce heterogeneity in protein expression, I designed the interface of a standardized TM helix of 21 AAs consisting of a poly-Leu backbone, a strategy previously applied to study the association of GASright proteins (Anderson et al., 2017). In my first protein design run, I used Solvent Accessible Surface Area (SASA) to identify the interfacial positions of a protein set at a geometry from my membrane protein analysis (Figure 3.4). The SASA was calculated for each position on the dimer and the interface defined as the 8 positions with the least amount of access to the solvent, or the most buried positions. Although we were successfully able to design sequences that associate, our energy score showed little correlation to association. However, we found that sequences with similar interfaces had better A video game screen with a black background

Description automatically generatedcorrelation with our energy score (**S4**).

**Figure 3.5 Protein design interfaces.** Sequence design was performed along a poly-Leucine backbone sequence with defined interfaces for each design region. Interface designated by positions with X.

For subsequent design runs, I standardized the interface for each region (Figure 3.5). The GASright was previously studied using similarly standardized backbones, so I chose to apply the same 8 positions as the interface. Since Right designs have similar geometric parameters outside of a larger x-shift, I used the same 8 interfacial positions as GASright. For the Left designs, I visually inspected a variety of backbone templates with unique crossing angles and defined 3 interfaces. These interfaces accounted for known left-handed association motifs including the leucine zipper, knobs-into-holes, and coiled coils (Ash et al., 2004; Bornberg-Bauer et al., 1998; Walshaw & Woolfson, 2003). Standardizing the interfaces permitted me to come up with a consistent mutational strategy to assess my proteins for their association at the given interface that is described in section 3.3.6.

3.3.4 Developing the energy terms

To predict the stability of my designed proteins, I applied the same set of energy terms used previously by CATM: CHARMM\_VDW for van der Waals packing, SCWRL4\_HBOND for hydrogen bonding, and CHARMM\_IMM1 to estimate the interactions found in the membrane environment (Anderson et al., 2017)(cites). These terms are computed for each protein during the sequence search to find the best interfacial sequence for the input geometric structure. To determine the stability of my designed dimers, I computed the interaction energies of the dimer structure and two monomers, and then subtracted the monomer energy from the dimer energy:

However, computing the monomer energy for each sequence during the sequence search is time consuming, resulting in a bottleneck in the algorithm and limiting the number of sequences I could design. To account for this, I developed an energy term that estimates the monomer energy of each sequence: BASELINE\_MONOMER.

A graph with a line graph

Description automatically generated with medium confidence

**Figure 3.6 Developing the BASELINE\_MONOMER term.** A baseline energy term was developed to increase computational speed, estimating the stability of the sequence as a monomer. A strong correlation was found between the actual Monomer energy (x-axis) and the BASELINE\_MONOMER (y-axis).

The BASELINE\_MONOMER term was created by measuring the energy of the previously mentioned terms for each individual amino acid on a monomeric helix. I computed the energetics of 10000 random sequences and measured the self and pair energies for each amino acid. The self-energy represents the energy contribution for an individual amino acid to the protein stability alone, while the pair energies represent the energy contribution between any two interacting amino acids (**cite**). I measured the pair energies for all AA pairs on the sequence and found that there was no longer a pair interaction between amino acids more than 10 away from each other. From each iteration, I calculated the average of all self and pair energies and saw a strong correlation between the measured monomer energy and the BASELINE\_MONOMER term (Figure 3.6). This BASELINE\_MONOMER term was made for only the subset of amino acids used in design (Figure 3.3B) and would need to be rerun to establish the term for any additional amino acids. This term was helpful in decreasing computational time, enabling us to design 1000s of sequences within a week.

Another issue that I encountered was that many of my initial designs were often composed of only 2-3 different AAs. This result could impact our protein expression and insertion, as natural membrane protein sequences are often made of a diverse set of AAs. To account for this sequence diversity, I developed a SEQUENCE\_ENTROPY term that outputs an energy based on how similar a sequence is to the composition of a natural membrane protein sequence (Figure 3.3). To convert the composition of AAs in a membrane sequence to an energy term, I utilized the following equation based on the Boltzmann entropy formula:

where R is the gas constant and T is temperature defaulted to 298K (RT = 0.592). To compute the sequence entropy, I needed to calculate the probability that the sequence occurs in membrane. First, the number of each AA (#AA) is counted within the sequence. Using these values, I then calculated the number of possible permutations for the sequence. This is determined using the following equation:

where n is the number of positions, which is divided by the factorial for #AA in the sequence multiplied, or the total number of combinations possible. The probability is computed using the frequency of each AA in membrane protein sequences (freq\_AA) to the power of the number of each AA in the sequence multiplied by the permutations:

This probability is inserted into the sequence entropy equation, returning a value that can be applied as an energy term for each sequence. My algorithm utilizes this term as an additional energy, predicting the stability of my designed homodimer sequences and adjusting the energy by the likelihood that they are found in natural membrane proteins sequences.

A video game screen with a black background

Description automatically generated3.3.5 Sequence Search

**Figure 3.7 Sequence Search Example.** A random position on the input sequence (red) is selected and switched to another AA. The energy is then calculated for the new sequence and compared to the energy of the previous sequence. Sequences with more stable energies (more negative) are always accepted.

For each input geometry, the algorithm first defines the best sequence using the Self Consistent Mean Field (SCMF) theory as implemented in MSL. This method estimates the conformational entropy of each design AA as a probability that it is present within the dimer interface (Koehl & Delarue, 1994). The sequence from the SCMF is then run through a MC sequence optimization: A random position on the interface is switched to a random AA, and the energy is calculated using the combination of the CATM energy terms and the developed energy terms defined in section 3.4.3.

Each energy term can be multiplied by an optional weight, meaning the total energy can be changed proportionately by the given weights. Previous research in our lab defaulted the weights of the CATM energy terms to 1, which was repeated in this study. To determine if SEQUENCE\_ENTROPY performed better at different weights, I ran a test using weights of 1, 5, 10, 50, and 100 and calculated the AA composition in these designs. Weights greater than 10 were found to be optimal for mimicking the AA composition found in TM helical pairs. I chose to use 10 as it resulted in the SEQUENCE\_ENTROPY term affecting the total energy at the same order of magnitude as the other terms. The equation for the total energy is as follows:

The total energy is used for the acceptance criteria, and the MC searches for a multitude of sequences before reaching an energetic minimum. The sequences accepted during the search are saved into an output trajectory file alongside the energy. The sequence with the best total energy is saved, and that single design undergoes backbone refinement.

3.3.6 Backbone Refinement

After initially starting with a specific backbone template, the newly designed sequence undergoes an MC based structural refinement procedure. The structure undergoes MC backbone perturbations, where one of the four inter-helical parameters (Figure 3.1A: x-shift, crossing angle, axial rotation, z-shift) is chosen and shifted during each cycle. The total energy for the refined structure is used as the acceptance criteria, with the BASELINE\_MONOMER energy being replaced by the computed monomer energy and the SEQUENCE\_ENTROPY term no longer applied:

The backbone refined geometry can be input into the sequence search to find other designable sequences. The sequence, energetics, geometries, and their corresponding structures are output to a folder for analysis (**Table Sx**).

A screenshot of a video game

Description automatically generated3.3.7 Mutating the interface

**Figure 3.8 Clash and Void mutations.** Mutations to Ile (top) results in interfacial positions on one helix overlapping with atoms on the opposite helix. Mutations to Ala (bottom) results in holes at the interface. Each of these mutations was expected to decrease association.

For each protein, I made point mutations on the interface corresponding to interfacial positions expected to decrease association. I created a program that accepts the structure and interface as inputs, individually mutates each interfacial position to a given amino acid, and outputs the energy and structure. The first mutation is a clash mutant, where an interfacial position was mutated to an Ile which can protrude into the opposing helix, often disrupting the ability for a protein to associate. The second mutation is a void mutant, where an interfacial position was mutated to an Ala, aiming to decrease the amount of packing at the interface. We expect these mutants to enable us to determine if our proteins associate by the designed interface. Clashing mutants that had the highest energy (least stable) and void mutants with the most interfacial SASA were chosen for experiments (Figure 3.8). To evaluate the energy of my mutants, I created a separate backbone refinement program that improves the structure and energetics described in section 3.3.6. This updated algorithm was used for determining energies for all sequences and is detailed in supplementary figure S7.