Sequence Design of a Hydrophobic-Hydrophilic Polymer

Sally Jiao

ChE 210D Final Project, December 11, 2019

1 Abstract

The efficient inverse design of polymer and peptide sequences is an important and challenging problem, enabling design of novel folded proteins, macrostructures, ligand-binding complexes, etc. Here, a simple hydrophobic-hydrophilic polymer model is used to compare various optimization strategies. The genetic algorithm converges more quickly than gradient descent, though a bottleneck for both is the cost of simulations. Fitting a surrogate function may be able to significantly accelerate the optimization.

2 Introduction

The design of protein or other polymer sequences to achieve specific structures and functional properties is of great interest for a large variety of applications, including pharmaceuticals, industrial formulations, surface-mediated interactions, etc. Simulations provide a promising route to approach this problem, allowing for direct observation of molecular structures and ability to sweep over a parameter of interest, and, when paired with an optimization algorithm, enable automated molecular design. Inverse design through molecular simulation is, however, complicated by the high dimensionality of the problem (the number of possible sequences grows exponentially with the number of monomers), the difficulty of the forward evaluation (connecting sequence to either structure or a functional property), and the challenges related to accurate modeling of realistic polymers and peptides. Simplified models can, however, reduce the complexity of the problem and allow for faster forward evaluations, while at the same time elucidating underlying design rules. Here, a simple model of a hydrophobic-hydrophilic (HP) polymer is used to test various optimization algorithms.

3 Methods

3.1 Model

An off-lattice model by Stillinger et al. [3] was used to simulate a single HP polymer. Equation 1 gives the energy function. In this model, the bonded potential is an angle potential (where the angle is defined as shown in the inset of Figure 1), while bond lengths are fixed at 1. The Lennard-Jones-like non-bonded term depends on ξ_i which is 1 for a hydrophobic bead and -1 for a hydrophilic bead. Thus, the coefficient of $r_{i,j}^{-6}$ is -1 (strongly attractive) for hydrophobic-hydrophobic interactions, -1/2 (weakly attractive) for hydrophilic-hydrophilic interactions, and 1/2 (weakly repulsive) for hydrophobic-hydrophilic interactions. A cutoff of 2.5 was used for the non-bonded interactions.

$$U = \sum_{i,j,k} \frac{1}{4}(1 - \cos\theta_{i,j,k}) + \sum_{i+1<j} 4(r_{ij}^{-12} - \frac{1}{8}(1 + \xi_i + \xi_j + 5\xi_i\xi_j)r_{i,j}^{-6}) \tag{1}$$

This model was initially simulated in two dimensions [3], but later work (e.g. finding minimum energy structures) simulated the model in three dimensions [2], as in this present study.

3.2 Simulation

Configurational Bias Monte Carlo was used to simulate the polymer, following the off-lattice procedure in Frenkel and Smit [1] with a single move: regrowth of one end of the polymer from the middle bead (offset by 1 for the case of an even number of beads). The end that is regrown is chosen randomly. For each regrown bead, 20 trial segments are generated. All simulations were run at a dimensionless temperature of 1.0, unless otherwise noted. Acceptance ratios were typically around 0.6 for a 15-bead chain and 0.8 for a 10-bead chain. Longer chains would thus likely require a more extensive set of regrowth moves (e.g. regrowing a shorter length of the end of the chain instead of always regrowing an entire half of the chain).

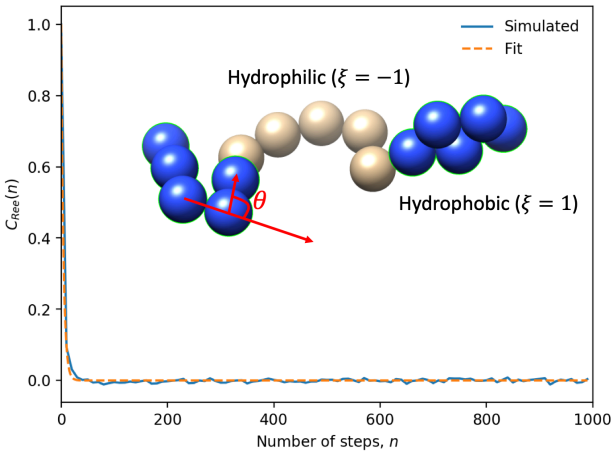

Figure 1: The autocorrelation function for the end-to-end distance decays quickly. The correlation "time" computed from an exponential fit is 4.3 steps.

The end-to-end distance (the distance from the first bead to the last bead) was saved every 10 steps. Because half of the polymer is regrown every step, the end-to-end correlation "time" is very fast, as shown by the autocorrelation function for the end-to-end distance plotted in Figure 1. An exponential, $f(n) = \exp(-n/\tau_c)$, was fit to the autocorrelation function, yielding a correlation "time" of $\tau_c = 4.3$ steps. As such, no additional advanced sampling techniques were used to simulate this system. For all analyses and optimization algorithms, the end-to-end distance distribution was generated by binning the end-to-end distances in bins of width 0.5 spanning 0 to 10.

3.3 Optimization

3.3.1 Gradient-free: genetic algorithm

A simple genetic algorithm was implemented:

1. Initial population generation: an initial population of 10 chains was generated randomly (for each chain, each bead was chosen to be -1 or 1, with equal probability).
2. Fitness calculation: a fitness score is computed for each chain in the current population. A chain's fitness is the inverse of the relative entropy, $1/S_{rel}$, where S_{rel} is given in equation 2 and is the relative entropy between the end-to-end distribution generated by simulating that chain and a target end-to-end distance distribution. A value of 0.001 was added to each bin of the distributions for numerical stability. The simulations used to compute the fitnesses were run for 5000 steps.
$$\int P_T(R_{ee}) \ln \frac{P_T(R_{ee})}{P_M(R_{ee})} dR_{ee} \tag{2}$$
3. Selection: out of the fittest 20%, two parents are randomly selected, weighted by fitness.
4. Crossover: two children are generated from the parent sequences. See Appendix for details.
5. Mutation: for each child, each bead in its sequence is randomly flipped with a probability $\gamma = 0.1\exp(-j/100)$ where j is the generation number.
6. Repeat: the two children are added to the population and the process begins again with step 2, ending after 200 repetitions (i.e. 200 generations).

3.3.2 Gradient-based: gradient descent

If a derivative of the objective function (here, S_{rel} given in equation 2) can be computed with respect to the parameters (ξ_i), then gradient-based optimization methods can be used. Instead of restricting ξ_i to the set $-1, 1$, they are here treated as continuous variables. The analytical form of the derivative of the objective function is given in equation 3 where the subscript M denotes that the quantity is computed in the ensemble of the simulated system and where the derivative of the potential energy is given in equation 4.

$$\frac{\partial S_{rel}}{\partial \xi_i} = \beta \left(\left\langle \frac{P_T(R_{ee})}{P_M(R_{ee})} \frac{\partial U}{\partial \xi_i} \right\rangle_M - \left\langle \frac{\partial U}{\partial \xi_i} \right\rangle_M \right) \tag{3}$$

$$\frac{\partial U}{\partial \xi_i} = \sum_{j \neq i, i-1, i+1} -\frac{1}{2} r_{ij}^{-6} (1 + 5\xi_j) \tag{4}$$

In order to compute the derivative, an initial simulation of 5000 steps was used to compute the end-to-end distance distribution, and then another simulation of 5000 steps was used to compute the derivatives. For each step, the gradient is computed, and the bead types ξ are updated with: $\xi_\mathbf{i} \leftarrow \xi_\mathbf{i} + \gamma \frac{\partial S_{rel}}{\partial \xi_i}$ where the step size is $\gamma = 0.1$. The initial guess is randomly drawn for each bead independently from a uniform distribution over $[-1, 1)$.

4 Results

End-to-end distributions for some exemplary sequences are given in Figure 2, where T indicates a hydrophobic bead, and H indicates a hydrophilic bead (thus, $T_5H_5T_5$ is a chain of length 15 with 5 hydrophobic bead on either end surrounding a middle section of 5 hydrophilic beads). The distributions behave as expected, with the completely hydrophobic polymer the most collapsed, the polymer with hydrophobic ends showing two preferred end-to-end distances, presumably one in which the hydrophobic ends are interacting, and the T_8H_7 the most extended, likely due to repulsion between the two ends of the polymer. To generate these distributions, simulations were run for 500,000 steps. While these long simulations generate smooth end-to-end distance distributions, they are expensive, so shorter simulations are used in the inner objective function / fitness evaluation of each optimization algorithm, as described above. This comes with some loss of accuracy, (see Figure 5 in the Appendix).

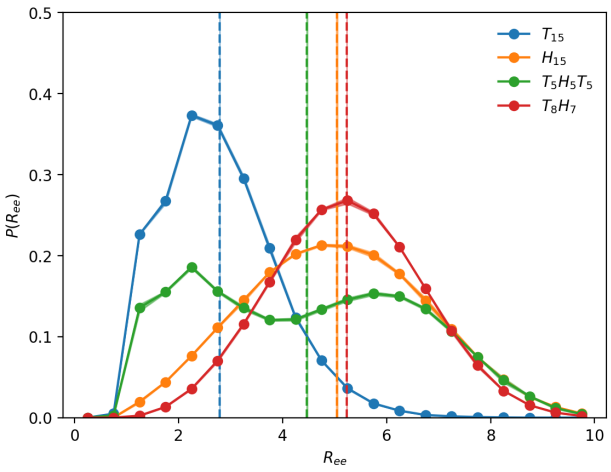

Figure 2: The end-to-end distance distributions show that the completely hydrophobic polymer is most collapsed, while a "diblock copolymer" with half hydrophobic and half hydrophilic beads is most extended. Shaded regions give the standard deviation from three independent runs.

4.1 Optimizing with the genetic algorithm

Figure 3i shows the results of genetic algorithm optimizations of four different target end-to-end distributions: the distributions computed from simulations of $T_5H_5T_5$ (hydrophobic ends), T_8H_7 (diblock copolymer), and $THTHTHTHTHTHT$ (alternating), and a fictitious distribution generated from three gaussian distributions. For each, the target distribution (blue circles) is compared to the optimized distributions, both the one obtained from the fitness calculation performed for the optimal sequence during the genetic algorithm run (orange dashed line) and one from a longer simulations of the optimal sequence (green triangles). The diblock copolymer case is an interesting test for the genetic algorithm, because both T_8H_7 and H_7T_8 should be optimal sequences, but when combined as "parents" to produce a child distribution, will result in a polymer with either hydrophobic or hydrophilic ends, which should be much more collapsed (due to aggregation of the ends) than the diblock copolymer (where the ends are repulsive). Still, the algorithm converges on a reasonable approximation of the distribution. The algorithm performs less well for the fictitious distribution, but it is unclear if any sequence would actually be able to generate such a distribution. In this case, the slight differences between the "Optimized" and "Refined" distributions suggest that another shortcoming of the algorithm is that the shorter simulation used to compute the fitness function can return an overestimate of the true fitness due to error in the simulated end-to-end distribution.

Figure 3ii shows that the efficiency of the genetic algorithm is sensitive to parameters of the algorithm, such as the mutation rate and the length of the simulation used to compute the fitness of a given sequence. For these systems, a decaying mutation rate was more efficient than a fixed mutation rate (further into the optimization, refinement of the already discovered minima is prioritized over discovery of new minima).

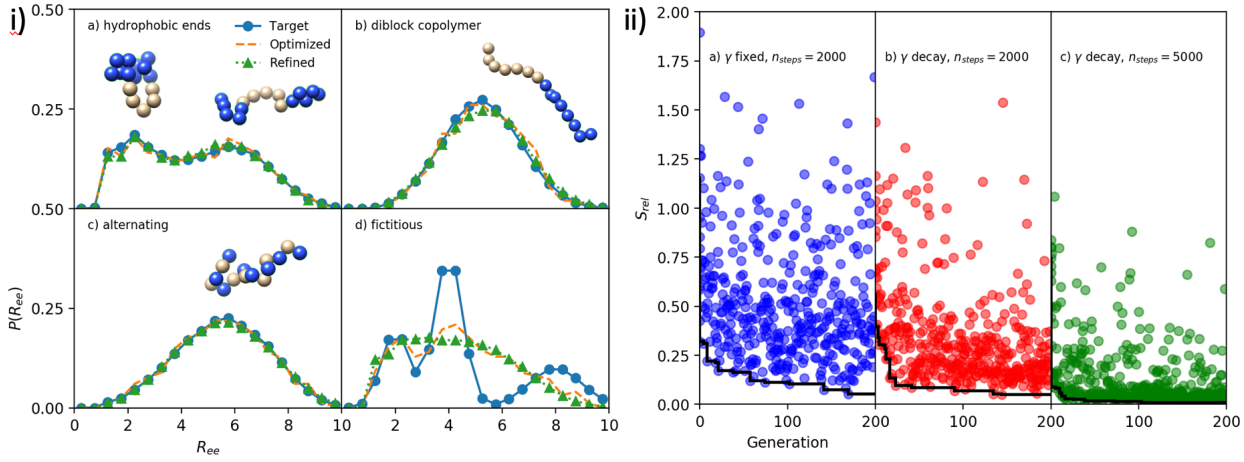

Figure 3: i) The end-to-end distance distributions of the target and optimal (as found by the genetic algorithm) distributions show good agreement for the cases where the target distribution is known to be that of a specific sequence (a-c). A fictitious distribution (d) is more difficult for the algorithm. ii) The trajectories (dots show the S_{rel} for each child in a given generation, black lines show the most optimal S_{rel} found up to that generation) show that the genetic algorithm is sensitive to the parameters of the algorithm.

4.2 Optimizing with gradient descent

Figure 4i shows the results from four separate runs of gradient descent optimization where the target distributions were distributions computed from simulations of a) H_{15} (completely hydrophilic) b) $T_5H_5T_5$ (hydrophobic ends), c) $THTHTHTHTHTHTHT$ (alternating), and d) T_8H_7 (diblock copolymer). Each gradient descent run is half as expensive as one genetic algorithm run (each genetic algorithm run is 200 generations with two simulations performed per generation, while each gradient descent run is 100 steps with two simulations performed per step). However, in general, gradient descent performs less well than the genetic algorithm. While it is able to converge to the target distributions for some runs for some target distributions, for other targets (all hydrophilic and diblock copolymer), none of the runs converge to the target. Examination of the trajectories of the four runs shown in Figure 4ii (the colors match the colors of the distributions) shows that for these cases, none of the trajectories starting from randomly chosen initial sequences consistently converge to the target sequence ξ_i values (shown as dashed black lines) and in some cases, seem to converge to the inverse. This phenomenon suggests the search space is non-convex. Starting from an initial sequence that is designed to be close to the target sequence (blue and orange curves) does lead to convergence to the target sequence (in most cases). Thus, when starting at a randomly generated initial sequence, the algorithm is likely simply optimizing to a local minimum in S_{rel}.

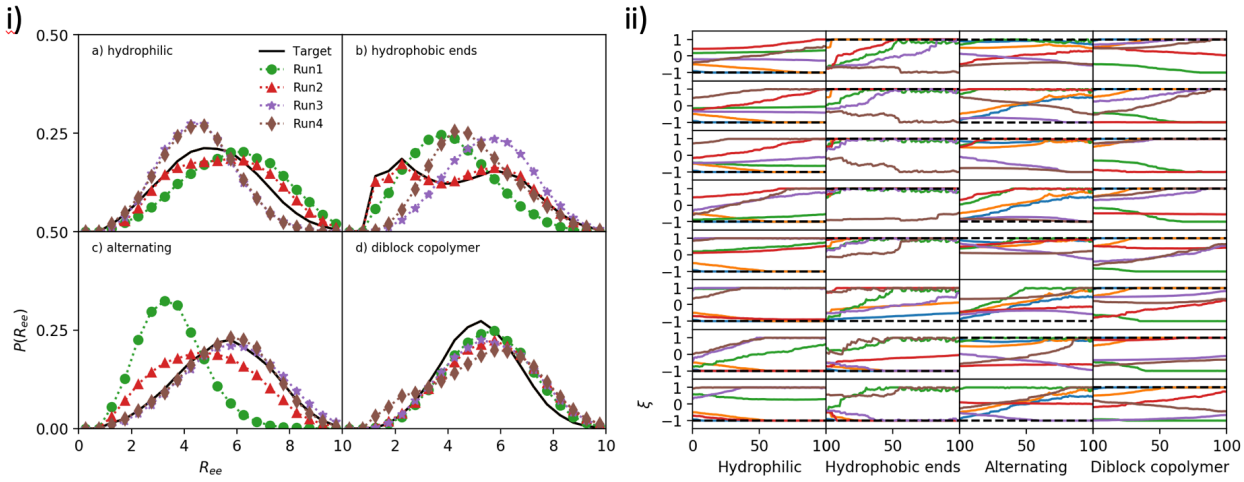

Figure 4: i) Comparison of target (black) and optimized end-to-end distributions for four different runs of the gradient descent algorithm, all starting from randomly generated initial sequences (green, red, purple, brown) shows that in general, gradient descent is not able to converge to the target distribution under these conditions. ii) The trajectories of the algorithm starting from those randomly generated initial sequences (green, red, purple, brown) show that the algorithm likely is converging to a local minimum, as opposed to the global minimum (black dashed lines). Initializing the algorithm with a guess that is closer to the globally optimal sequence produces better convergence (blue, orange).

The animation shows how an optimization algorithm might change the sequence of the hydrophobic-hydrophilic polymer in order to more closely approximate the target distribution. For each sequence (three are shown in this animation), a simulation is run and the end-to-end distribution becomes more refined / smoother the longer the simulation. Once the simulation finishes, the algorithm computes the distance between the two distributions and changes the sequence accordingly and a simulation of the new sequence begins. I do not own the music played in the animation.

References

[1] D. Frenkel and B. Smit. *Understanding Molecular Simulation.* Academic Press, 2002.

[2] S. Y. Kim, S. B. Lee, and J. Lee. Structure optimization by conformational space annealing in an off-lattice protein model. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 72(1):1–6, 2005.

[3] F. H. Stillinger, T. Head-gordon, C. L. Hirshfeld, T. B. Laboratories, and M. Hill. Toy model for protein folding. 48(2), 1993.

I would like to acknowledge Pratyush Kumar, Koty McAllister, William Jiao, Nicholas Yang, and Scott Shell for helpful discussions and ideas.

5 Appendix

5.1 Genetic algorithm details

Fitness function: $1/S_{rel}$ is used as the fitness function and maximized (as opposed to minimizing S_{rel}) so that the probability that a sequence is chosen as a parent can be related directly to its fitness.

Crossover: After the crossover step, one child has the same sequence as one of its parents, up until a randomly selected crossover point, after which it has the same sequence as its other parent. The opposite is true for the other child.

5.2 $P(r)$ uncertainty as a function of simulation length

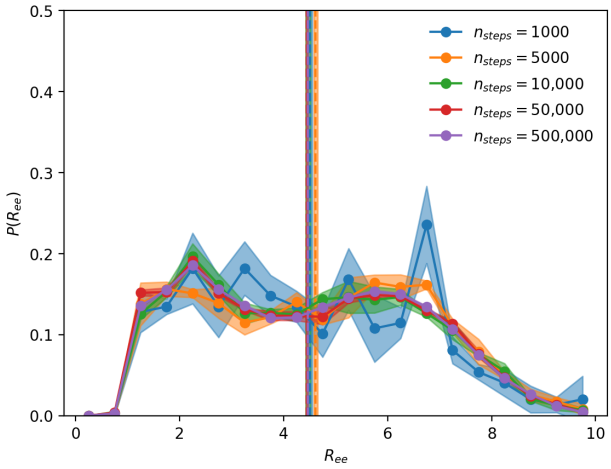

Figure 5: The uncertainties in the generated end-to-end distance distributions grow as the number of steps decreases, as expected, for the $T_5H_5T_5$ polymer. At $n_{steps} = 5000$, the doubly peaked structure is still captured.

5.3 Finding a surrogate function

In both the optimization algorithms, the expense of the forward pass (the simulation) was one of the limiting factors. Finding a surrogate function that can accurately estimate $P(r)$ or S_{rel} and thus can serve as a much faster substitute for running a simulation can significantly speed up the optimization, allowing us to search a much larger fraction of design space. However, finding this surrogate function is non-trivial. Figure 6 shows the results of fits of S_{rel} (computed between a simulated distribution and the reference end-to-end distribution of $T_5H_5T_5$) with linear combinations of various features of the sequence (e.g. number of hydrophobic groups, cluster distribution of hydrophobic beads etc.) for a dataset comprising 100 randomly generated sequences and their distributions simulated for n_{steps}. Simply fitting S_{rel} to a linear combination of the sequence (panel a) ($\sum_i c_i \xi_i$ where c_i are the coefficients of the linear fit) actually produces less error (measured as RMSD) than fitting (panel b) to the number of hydrophobic groups or (panel c) to a linear combination of a "coarse-grained" representation of the sequence, where each set of three beads is represented by 1 if there are more hydrophobic beads or -1 if there are more hydrophilic beads. However, a linear fit of a vector describing the cluster distribution of hydrophobic and hydrophilic

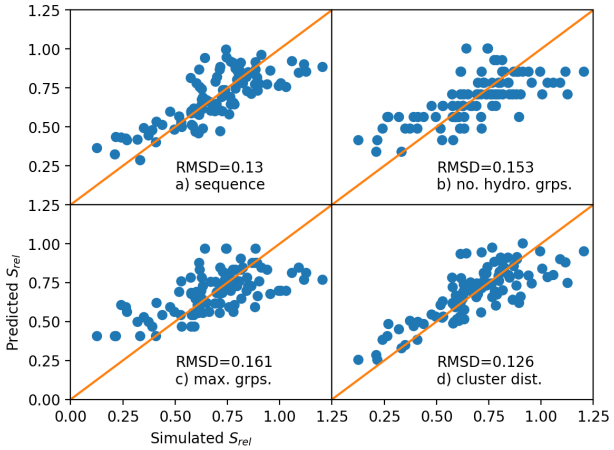

Figure 6: Fits of S_{rel} to linear combinations of various features generated from the sequence (see text for details) shows that a linear fit to the sequence itself outperforms many naively engineered features.

groups gives slightly less error than just a fit of the sequence. The vector that represents the "cluster distribution" described in the text is a concatentation of two 15-element vector. The first is the cluster distribution for hydrophobic beads. The second is the cluster distribution for hydrophilic beads. For each, the element x_i in the distribution is the number of clusters of length i in the sequence, where a cluster is a set of consecutive hydrophobic or hydrophilic beads.

5.4 Other improvements

1. The genetic algorithm actually converges to a fairly optimal sequence relatively quickly, and therefore ends up repeating sequences. Currently, the algorithm has no "memory" of past sequences and simply recomputes a fitness by running another simulation. Thus, a simple way to make the algorithm more efficient would be to keep some memory of sequences generated and the computed fitnesses (a hash table can be used for fast lookups). A possible downside is that repeated fitness measurements for the same sequence reduce the impact of measurement error. At the very least, however, the lookup table could be used to combine all measurements of the fitness of a sequence into a more refined estimate. It could also provide a way to address another issue: the genetic algorithm does not currently understand that a sequence and its reverse are identical and therefore should have the same fitness. Keeping a lookup table and entering a sequence and the reverse (symmetrically identical) sequence with the fitness computed a single time for one of those sequences would somewhat reduce the search space.
2. Currently, the gradient descent step traverses some fixed step size γ in the direction of steepest descent. Alternatively, a line search could be used, though the performance of the line search in cases where there is measurement error would need to be evaluated.