DEVELOPMENT OF A PROTEIN FOLDING ENVIRONMENT FOR REINFORCEMENT LEARNING

by

Wenhao Gao

A thesis submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering.

Baltimore, Maryland

July, 2020

Abstract

The prediction of the three-dimensional protein structures from amino acid sequences has been a long-standing challenge in computational biophysics. In the last decade, considerable progress has been made by leveraging evolutionary information and deep neural networks. These efforts, however, have been focusing on solving the scoring problem. In this thesis, I introduce a Reinforcement Learning (RL) environment based on PyRosetta to solve the sampling problem directly. I argue that the sampling of protein conformations can be formulated as a Markov Decision Process (MDP) and addressed by an RL algorithm. I describe the observation, action, and reward specification for the protein folding environment and propose a graph-based representation of the states. I present initial baseline results of applying Proximal Policy Optimization (PPO) to solve this protein folding environment. This protocol can be easily generalized to other protein modeling tasks, such as loop modeling and protein design. The protein modeling domain poses a new challenge for RL, representing a new class of difficult problems with scientific value.

ABSTRACT

Primary Reader and Advisor: Dr. Jeffrey J. Gray

Secondary Reader and Advisor: Dr. Jeremias Sulam

Acknowledgments

I wish to express my sincere gratitude for my advisors, Dr. Jeffrey J. Gray, and Dr. Jeremias Sulam, who have always been supporting me. This project started from nothing previous but only my naive idea, and I didn't have positive results in an extended period. So I appreciate the patience and confidence they kept on me throughout the whole project. Without the two open-minded and helpful advisors, I might not have had the chance to try this idea.

I also wish to express my appreciation to Dr. Sai Pooja Mahajan, who has been the PyRosetta mentor of me during the project. We had many helpful discussions about the project and another review paper.

Last but not least, I would like to thank all members of Gray Lab. Everyone is so friendly and helpful that I had many inspiring discussions academically and received help in private life as well. Without them, I cannot finish this project.

Contents

List of Figures

Chapter 1

Introduction

The prediction of three-dimensional protein structure from its amino acid sequence, *i.e.*, protein folding, has been a grand challenge in computational biophysics for decades [1]. As the basic units of life that are responsible for various biological activities, proteins have an incredible variety of three-dimensional structures determined by the combination and order in which twenty amino acids thread the polymer chain. The atomic-level structures of proteins and complexes are necessary to understand their function and modulate or engineer them. While sophisticated experimental techniques primarily determine protein structure, computational structure prediction has been employed as an alternative to overcome the high cost and limitation of the experimental approaches.

The current methodology for computational protein folding is largely based

on Anfinsen's thermodynamic hypothesis [2]. It states that the native structure of a protein must be the one with the lowest free-energy, governed by the energy landscape of all possible conformations associated with its sequence. Finding the lowest-energy state is challenging because of the immense space of possible conformations available to a protein, also known as the "sampling problem" or Levinthal's paradox [3]. Furthermore, the approach requires accurate free energy functions to describe the protein energy landscape and rank different conformations, referred as the "scoring problem." Following this problem formulation, score functions [4] that estimate the free energy of protein conformation and Monte Carlo (MC) sampling approaches [5, 6] that employ random conformational moves have been widely used to predict the protein structures [1]. Though the methods have advanced dramatically in the past decade and have many successful applications [1,7], the accurate template-free protein structure prediction is still challenging.

The recent advances in deep learning have opened up new avenues in many areas [8–10], including protein folding [11–14]. In the most recent Critical Assessment of Structure Prediction (CASP13 held in 2018) [7], a biennial community experiment to determine the state-of-the-art in protein structure prediction, deep-learning-based methods accomplished a striking improvement in model accuracy, especially in the "difficult" target category where comparative modeling (starting with a known, related structure) is ineffective. Most

methods employ convolutional neural networks to predict the residue-residue contact/distance map and add it as a constraint to the score function to make a better-shaped energy funnel. These approaches are changing the problem of protein folding from a free energy optimization to supervised learning, with amino acid sequences as input and three-dimensional structure as targets. However, unlike the conventional score functions usually with physical meanings, most of the currently used statistical models are still "black boxes." The structures with lower scores in these cases are closer to the ones with predicted structural features, such as distance map, but not necessarily the ones with lower experimental free energy. Sometimes the sampled non-native structures' scores are lower than the near-native ones [15], which means the near-native structures would be rejected even when sampled.

In this work, rather than learn a constraint in score function, we employ deep learning models to enhance the sampling of protein structures. We formulate the protein folding problem as a Markov Decision Process (MDP) [16] and solve it with Reinforcement Learning (RL) algorithms [17]. RL methods have been used to solve optimization problems for high-dimensional structured data, such as small organic molecules [18] and computational chips [19]. Applying RL methods to fold proteins is rare, and previous attempts have focused on the two-dimensional hydrophobic-polar model [20–22]. Here, we combined RL methods with a graph representation of proteins to directly solve

three-dimensional protein structure. Besides, from the control process perspective, this work can also be seen as training an Artificial Intelligence (AI) agent to make a series of "moves" to fold a protein, similar to playing Foldit game [23, 24].

Chapter 2

Methods

2.1 Problem Formulation

2.1.1 Metropolis-Hastings Monte Carlo

The conventional approach, Metropolis-Hastings Monte Carlo (MHMC) [5, 6] is a type of Markov Chain Monte Carlo (MCMC) sampling algorithm that employs randomly selected conformational changes to sample a series of conformations from a probability distribution. This algorithm describes the sampling of protein conformation as a Markov process. A Markov process (S, P) is uniquely defined by the states $S = \{x_1, x_2, \dots\}$ and transition probability $P(x'|x)$ between each states. In the context of protein folding, the state space includes all conformations of the inquiry protein. The transition probability

from state x to another state x' is defined as:

$$P(x'|x) = \min\left\{1, e^{-\Delta E/kT}\right\} \tag{2.1}$$

where k is the Boltzmann constant, T is the temperature in Kelvin, $\Delta E = E(x') - E(x)$ is the change of free energy during this conformational change. Within each step of MHMC, a random conformational change, *i.e.*, a "move", is selected, then accepted or rejected as determined by the transition probability from the initial to the final state. When the move leads to a conformation with lower free energy (favorable), then the $\Delta E < 0$ and the transition probability $P(x'|x) = 1$, which means we always accept that move. If the move leads to a conformation with higher free energy (unfavorable), then we accept the move with a probability of $P(x'|x) = e^{-\Delta E/kT}$. If there is an unique stationary distribution of protein conformations, $u(x)$, by repeating this process, this Markov chain converges to stationary points with $u(x_2) = u(x_1)e^{-\Delta E_{21}/kT}$, under the assumption of microscopic reversibility [25]. In this way, we can sample a collection of protein conformations from the Maxwell-Boltzmann distribution.

Despite the significant progress with the MC approach in the last several decades, conformation sampling is still a bottleneck of accurate structure prediction. The free energy of the near-native structures is usually lower than the ones we sampled from the MC algorithm [15], which means the score func-

tion distinguishes the near-native and non-native conformations successfully, but we failed to sample the near-native states. One reason is that, under this Markov chain setting, transitions between different conformations are treated equally, including the ones lead us to the unfolded states. That makes us spend time trying unproductive moves during the exploration. Another reason is that the MC approach aims to obtain an ensemble property while we are formulating the protein folding problem as a global optimization. Although we can apply gradient-based optimization during an MC sampling, the primary purpose is still obtaining an ensemble with local minima only.

2.1.2 Markov Decision Process

Here, we describe the protein folding problem as a Markov Decision Process (MDP) [16]. An MDP, specified by the tuple (S, A, P, R), is a Markov chain with actions (A) and rewards (R) that provides a mathematical framework for the decision making process. Actions are defined as the moves transitioning one state to another. Rewards are real-valued numbers given to the decision-maker to distinguish which states and/or actions are preferred. The objective of this process is to optimize the accumulative reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{2.2}$$

where γ is a discount factor and R_i is the reward from time step i. In the context of protein folding, the definition of state is the same as above. The action space contains all legal moves that transform a protein conformation to another. The transition probability $P_a(x'|x)$ is the probability of taking move a from conformation x to conformation x'. The reward R is a score that evaluates the quality of current protein conformation space (in our case, the opposite of the free energy). Solving such an MDP means finding a series of conformations that maximize the reward received by the RL agent, which is naturally an optimization problem.

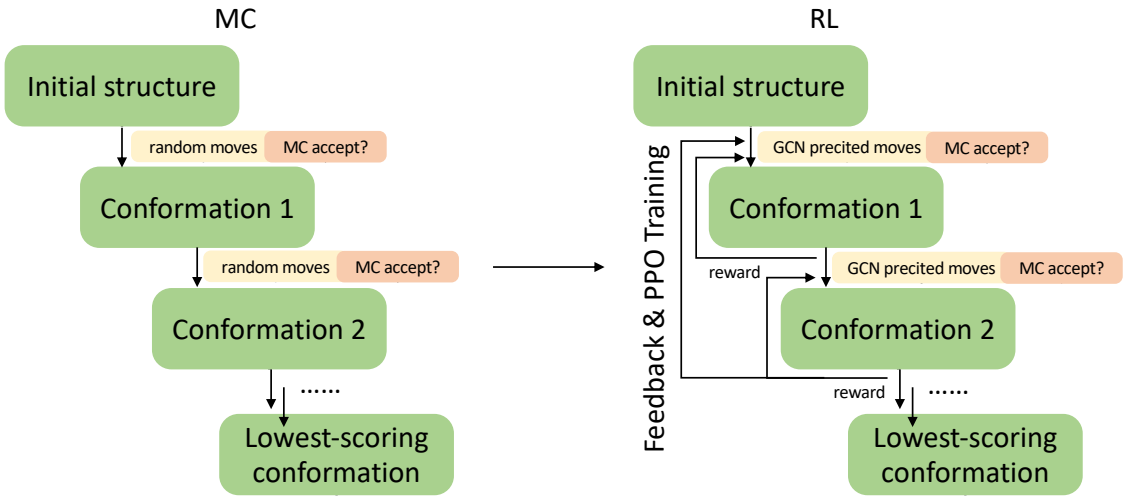


Figure 2.1: Comparison between Metropolis-Hastings Monte Carlo (labeled as MC) and a Markov decision process solved by reinforcement learning (labeled as RL).
This figure shows the major difference between MC and RL: moves are selected by an neural agent (GCN in our case), and moves and states have rewards. The MC accept means determining to accept or reject a move based on the MC criterion.

2.1.3 Reinforcement Learning

Reinforcement learning (RL) [17] is a class of approaches to solving MDPs that combines the concept of dynamic programming and machine learning. An agent (decision maker) is trained to explore an environment (problem) in an RL setting, aiming to maximize the received reward. An agent can be divided into two parts: value evaluation and policy estimation.

A value function evaluates how "good" an action or a state is:

$$V(s) = \mathbb{E}[G_t|S_t = s] \quad \text{or} \quad Q(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a] \tag{2.3}$$

where $V(s)$ is called state value, $Q(s,a)$ is called action-state value. They estimate the expectation of accumulative reward, G_t, we can get from the current state or action-state pair. A policy, $\pi(a|s)$, describes the behavior of the agent:

$$\pi(a|s) = P[A_t = a|s_t = s] \tag{2.4}$$

where $\pi(a|s)$ assigns a probability to each action in the action space given the current state. When we have an accurate value function, the policy simply assigns a probability of 1 to the action that maximize the expected reward and 0 to the others, which can be achieved by a greedy search.

Conventional dynamic programming methods, such as value or policy it-

eration [26], become infeasible when the problem is too complicated. In that case, estimating the value or policy with statistical models provides a solution. During the training, the agent predicts actions and receives rewards from the environment to tell whether it is a "good" move. The models are trained with the data generated from these trial-and-error interactions with the environment.

2.2 Implementation

2.2.1 Protein Environment Description

In this section, we specify a step-wise protein folding procedure that constitutes our learning environment. For simplicity, we only consider the coarse stage protein folding. All proteins are in centroid mode (ignoring the side chain of residues), and score3 [27] is selected as score function. Note that this environment can be easily extended by adding more elaborated movers and more accurate score functions.

State Space: We define the state at step t, s_t, as the conformation of the protein at this step. The protein conformations are represented as graphs that are fully observable to the RL agent. We will discuss the detail of this graph representation in Section 2.2.2. Every learning process starts from an extended

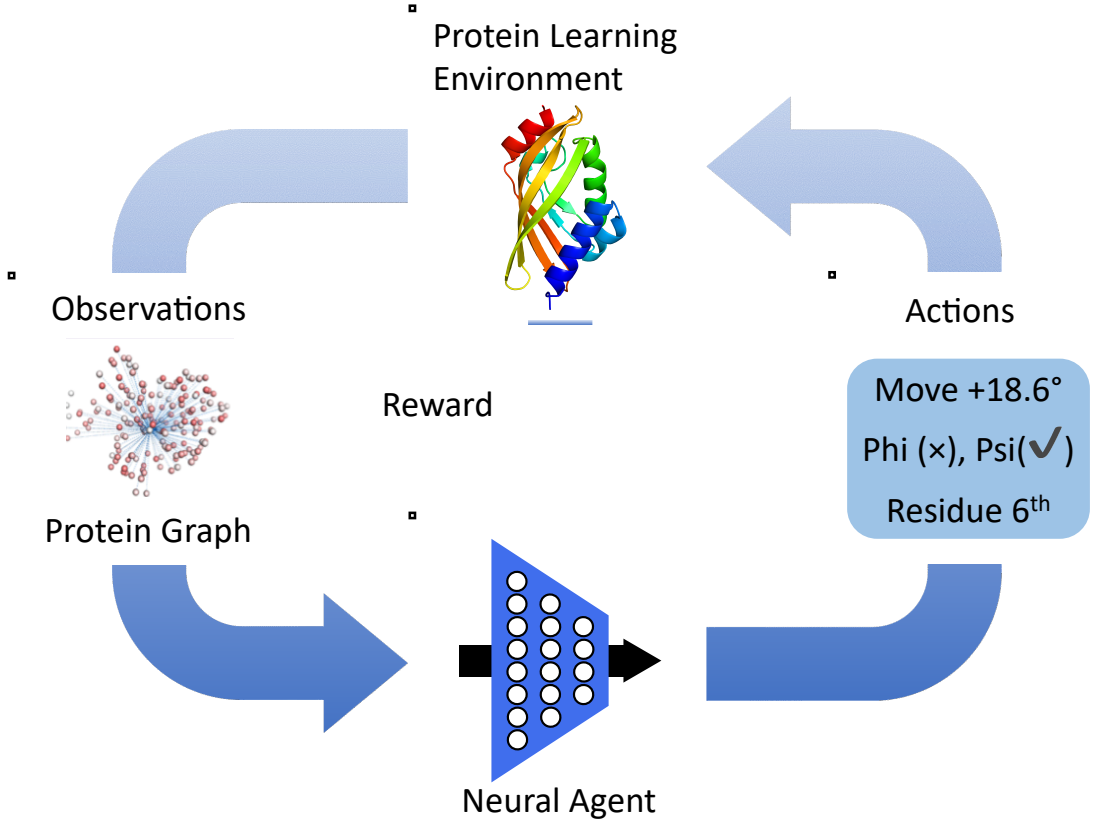


Figure 2.2: The schematic of our protein folding environment.
This figure shows the protein folding environment with its components plugged into a neural agent.

conformation where all torsion angles are set to 180.0 degrees.

Action Space: To directly compare to the baseline *RandomTorsionMover* in Rosetta [28], we define a combinatorial action space. We first predict which torsion angle should be moved, and then predict how many degrees we should move (see Figure 2.2).

State Transition Dynamics: We incorporate the transition probability of MHMC into our environment, called MC dynamics. The moves that lead to

higher free energy states would be rejected with a certain probability. For comparison purposes, we also tested deterministic dynamics where all transition probabilities are 1 once picked.

Reward Design: As we formulate the protein folding as an infinite-step process, there are no intermediate and final states. We define the opposite of free energy as the reward so that the RL agent's purpose is to minimize the free energy of conformations.

2.2.2 Agent Architecture

We represent states (protein conformations) as attributed graphs that preserve all necessary information and are transferable between proteins. We map each residue to a node, and an edge links each pair of nodes. The node features include one-hot encoded amino acid type, backbone torsion angles (ϕ, ψ, ω), normalized direction vector (C to C_α), and Rosetta one-body energy terms (*e.g.*, residue-wise contribution to the potential energy caused by Van der Waals interaction). The edge features include relative sequence position (*i.e.*, the gap $|i-j|$ for i^{th} and j^{th} node), distance lifted into a radial basis, normalized relative direction, a quaternion representation of relative rotation [29], hydrogen bond term and Rosetta residue-pair energy terms.

The graph representation is amenable to the Graph Neural Network (GNN) paradigm [29, 30], which has recently emerged as a powerful framework for

non-Euclidean data. In our implementation, we employ the edge-conditioned convolutional operator from a message passing neural network [31, 32]:

$$\boldsymbol{x}_i' = \mathbf{GRU}(\boldsymbol{x}_i, \sum_{j \in \mathcal{N}(i)} \boldsymbol{x}_j \cdot \boldsymbol{h}_{\boldsymbol{\phi}}(\boldsymbol{e}_{\boldsymbol{i,j}})), \tag{2.5}$$

where $\boldsymbol{x}_i$ is the embedding of the i^{th} node, $\boldsymbol{e}_{\boldsymbol{i,j}}$ is the edge feature between the i^{th} and the j^{th} nodes, $\boldsymbol{x}_i'$ is the updated embedding of the i^{th} node, $\mathcal{N}(i)$ includes the indices of all nodes connected to the i^{th} node, **GRU** is the Gated Recurrent Unit introduced in [33] and $\boldsymbol{h}_{\boldsymbol{\phi}}$ is a neural network. The state value is obtained through a global pooling operator based on iterative content-based attention mechanism [34], and the actions are obtained from multi-layer perceptrons (see Figure 2.3). Further details are provided in the code.

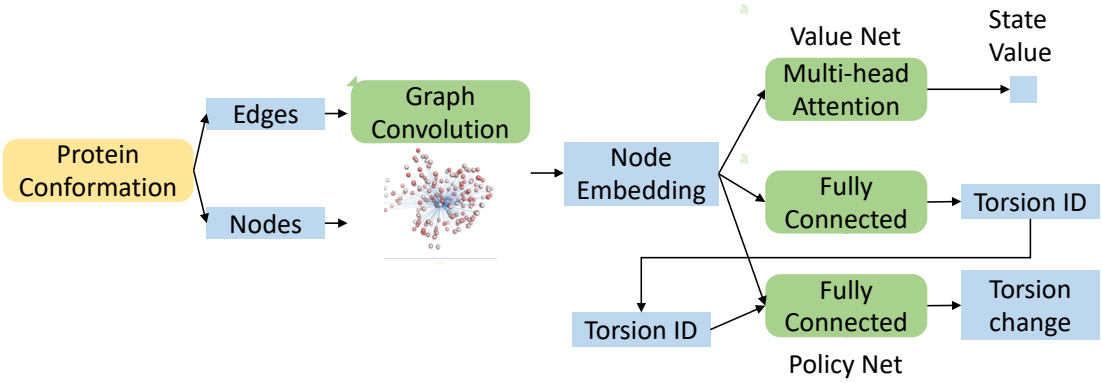


Figure 2.3: The architecture of neural agent.
Yellow block represent the protein object. Blue ones are tensors. Green blocks are the networks.

2.2.3 Proximal Policy Gradient

Because of the continuous nature of the action space, we need a policy gradient based method that optimizes the policy networks directly. Here we adopt Proximal Policy Optimization (PPO) [35], one of the state-of-the-art policy gradient methods. The main objective function of PPO is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \textbf{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.6}$$

where θ represents the parameters of the agent described by the policy $\pi_\theta(a_t|s_t)$. $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ denotes the probability ratio of the new and old policy, and $\hat{A}_t = \hat{Q}_t - \hat{V}_t$ is the estimated advantage. In the second term, the probability ratio is clipped to the range of $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyper-parameter (say 0.2). The final objective function takes the minimum of the clipped and unclipped terms so that we are maximizing a lower bound, *i.e.*, a pessimistic bound, on the unclipped objective. This modification is made to avoid excessively large policy update, hence to make the training process stable.

2.2.4 Knowledge Transfer

Our goal is to train a network that enables domain-adaptive policies for protein folding, which means that a single policy can fold multiple proteins, with only slight modification. Training such a policy is challenging, and pre-training

the networks with an expert policy is known to improve the training stability, sample efficiency, and performance [19, 36, 37]. In the protein folding problem, any experimentally solved protein structure can be viewed as an expert folding trajectory. We create a dataset comprising 2011 chains with different topology sampled from the CATH Protein Structure Classification database [38]. In each iteration, we randomly sample a chain and make a move on it (select one backbone torsion and change a random value). The prediction target is then the inverse move that leads to the native structure. The ground truth of the state value is defined as the negative of the score function. We divide the training into two stages: value learning and policy learning. The loss function of the two stages are:

$$\begin{aligned} L_1 =& 5 \times MAE(v) + CE(a_1) + MSE(a_2) \\ L_2 =& 0.01 \times MAE(v) + 5 \times CE(a_1) + MSE(a_2) \end{aligned} \tag{2.7}$$

where MAE stands for mean absolute error, CE stands for cross-entropy, MSE stands for mean squared error. v is the state value, a_1 is the torsion index, and a_2 represents how much we should change at this torsion.

Chapter 3

Experiments

To test the baseline policy gradient algorithm's performance on the protein folding task, we select three simple proteins to experiment with (Trp-Cage, a 20-residue α miniprotein, PDB ID: 1L2Y [39]; 30-residue terminal domain of the Carp Granulin-1, mainly β strand, PDB ID: 1QGM [40]; 31-residue designed coiled coil, PDB ID: 1COI [41]). For each protein, we ran experiments with conventional MHMC (labeled as MC), RL described above (labeled as RL), RL with deterministic dynamics (labeled as RL+deterministic), and RL starting from a pre-trained agent (labeled as RL+Pre). The neural agents were updated once every 20 action steps. All experiments except RL with deterministic dynamics were run for three independent trajectories for each case, and 10^6 action steps for each trajectory.

3.1 Overall Performance

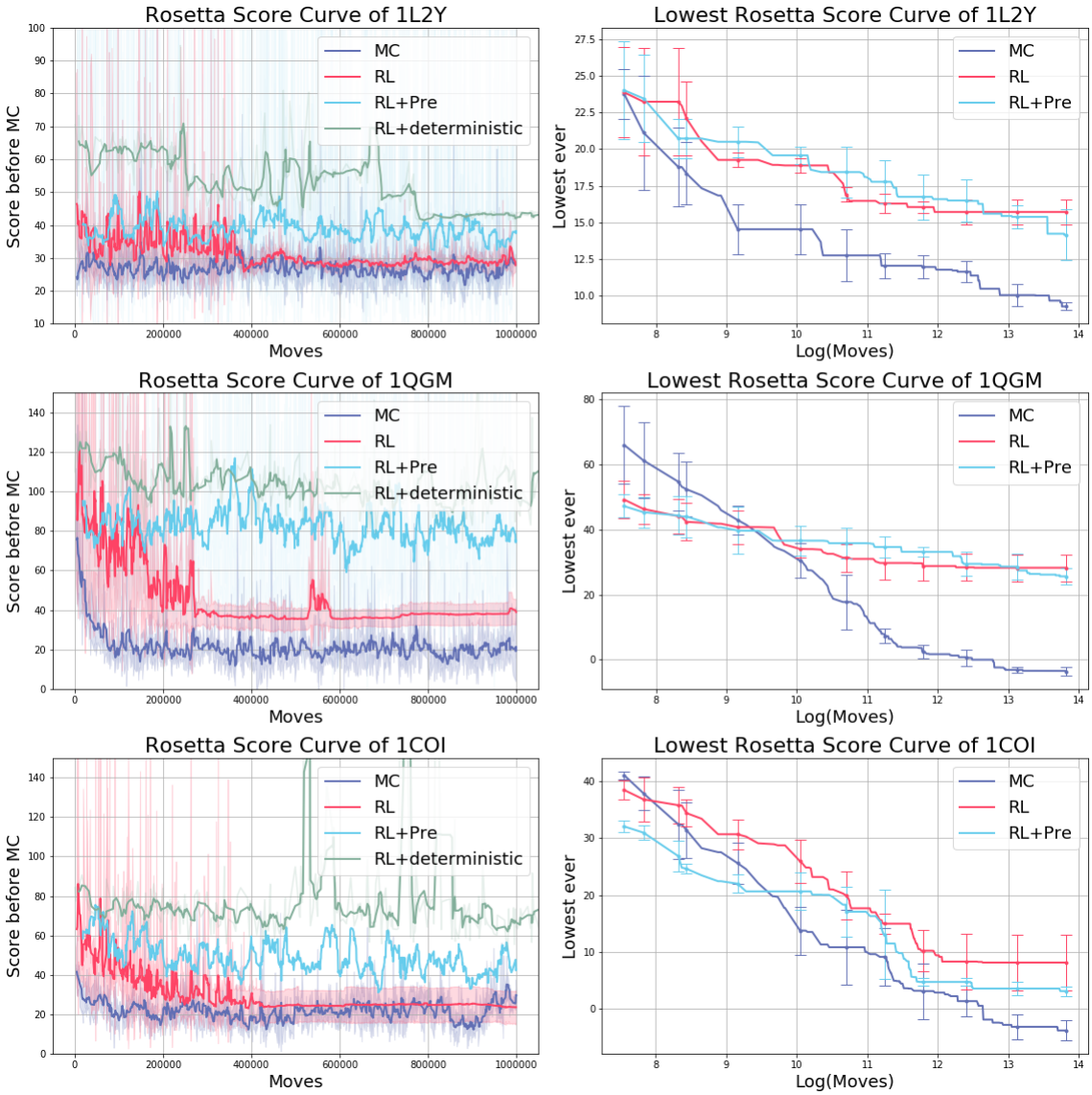


Figure 3.1: Rosetta Score During Training.
Each row shows the result of one protein. The left column shows the score (negative of the reward) before applying the MC criterion for each step; the right shows the lowest score seen before the current step. For each experiment, three independent trajectories were run, and the solid lines are the mean value of the three. Standard deviation is shown with faint color (Left) and error bars (Right).

The overall performance of RL on the three proteins is shown in Figure 3.1. Comparing with the conventional MC, all of the agents trained with MC dynamics converged to comparable scores after 400,000 action steps. However, the lowest scores of these agents are higher than the corresponding lowest scores found by the conventional MC algorithm. This result is due to that the RL agents converge at local minima and stop exploring, while the MC algorithm does not have a stop criterion and keeps exploring. From Figure 3.2, we can see the RL agents stopped to propose conformations with increased free energy after convergence, while the MC did so routinely. Though some of the converged conformations are close to the native ones (see Figure 3.3a for an example, the overall fold is correct, and the agent successfully folds the vital helical structure), this result is not satisfactory, especially considering the additional computational overhead.

The inability of the RL agents to find a lower score can be attributed to two reasons. First is the policy gradient methods themselves are easy to converge to local minima. Although policy gradient methods provide a flexible framework (*i.e.* directly learn a parametrized policy) to address the complex control problems by performing a standard gradient descent optimization, the total loss is usually non-convex with respect to the parameters [42]. As previously reported [43], even for some simple problems that are solvable by classical methods, policy gradient methods may converge only to local minima. Another

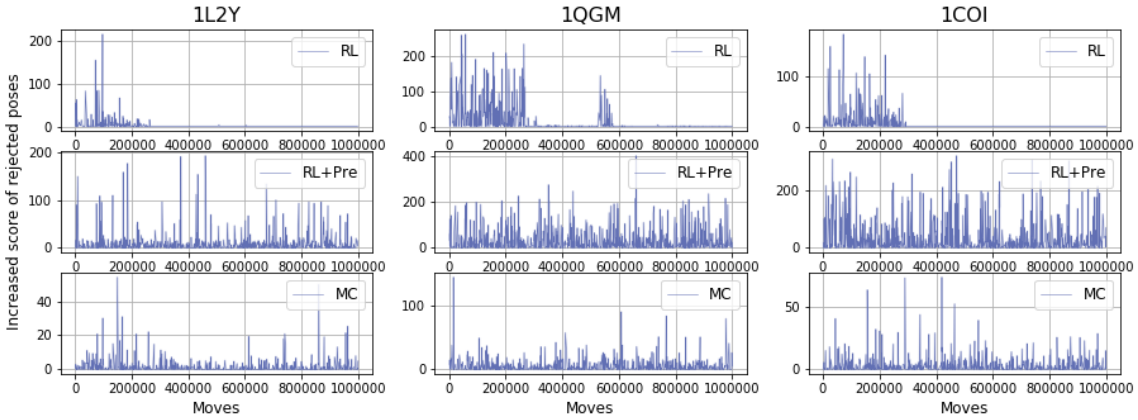
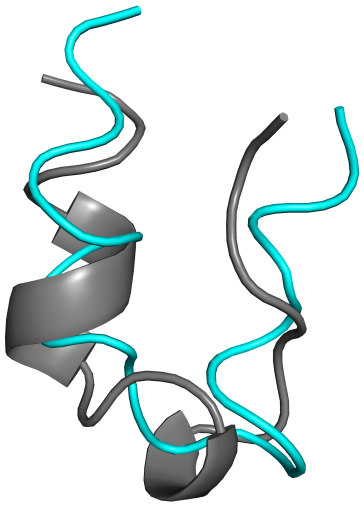


Figure 3.2: Increased Free Energy During Training.
The free energy increase before MC rejection. Each column represents a protein (1L2Y, 1QGM, 1COI), and each row represents a type of experiment (RL, RL+Pre, MC).

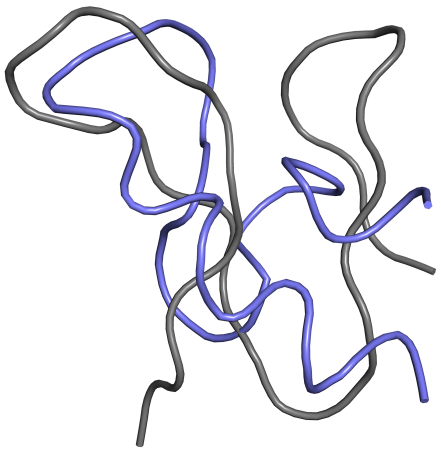
reason is that the free energy landscapes of proteins are usually complex and high-dimensional [1], even though we are using a centroid mode score function that should be smoother than the all-atom ones.

3.2 Effect of MC Dynamics

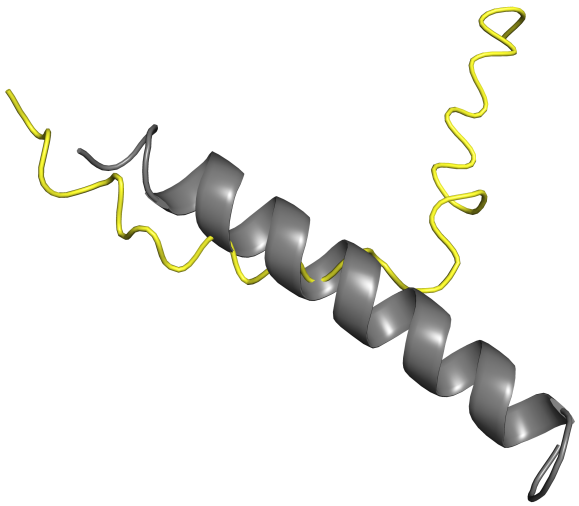
Compared with the agent trained under MC dynamics, most of the agents trained with deterministic dynamics developed a stable policy to fold the proteins within 1,000,000 action steps. The agent converged only in the case of 1L2Y and failed to converge to a stable state in the remaining two cases. This result implies that the MC dynamics accelerates the agents' convergence by rejecting unstable states. But on the other hand, this dynamics may also con-



(a) Trp-Cage, PDB ID: 1L2Y



(b) Termini of the Carp Granulin-1, PDB ID: 1QGM



(c) A designed coiled coil, PDB ID: 1COI

Figure 3.3: The converged structures.
Each structure comes from one trajectory and is aligned to its native conformation (colored in gray).

tribute to the agents' convergence to local minima. As the MC dynamics rejects the unstable states, the agents are restricted to the states neighboring around the stable ones. The conventional MC relies heavily on the inherent randomness to overcome the energy barrier and explore a larger conformational space. Therefore, leveraging randomized value [44] or intrinsic rewards [45] to enhance the exploration of the RL algorithm should improve overall performance. Besides, from the case of 1L2Y, the RL agents could potentially result in a converged policy, taking more data. Thus improving the sample-efficiency of the RL algorithms, such as employing experience replay [46], may also help.

3.3 Effect of Pre-Training

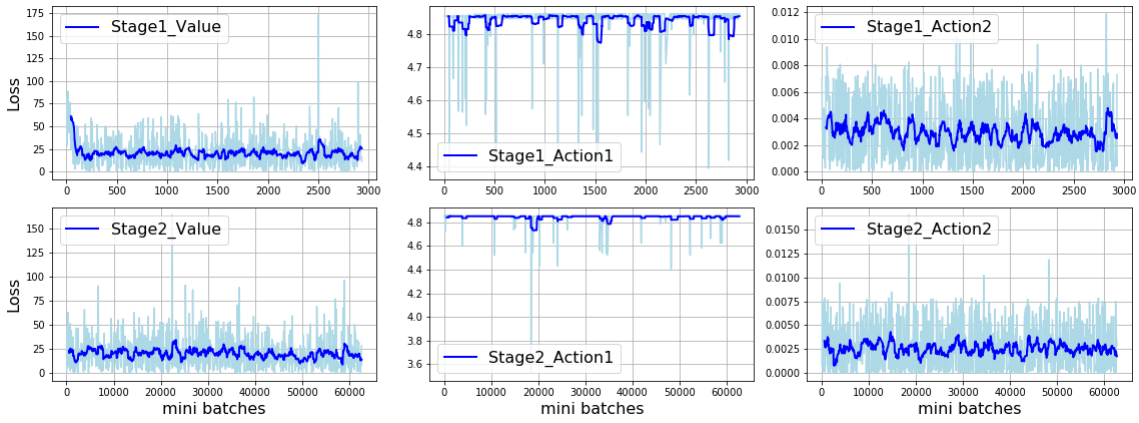


Figure 3.4: Learning Curves of Pre-trained Models.
The first row represents the first stage of training, with emphasis on value learning. The second row represents the second stage of training, with emphasis on policy learning. Within each row, the three columns (from left to right) represent the loss of value network, network to predict which torsion should change and network to predict how many degrees should change.

In all cases, the agents starting from pre-trained parameters failed to develop a stable folding policy. We attribute this failure to two reasons. First is the inappropriate action space definition. From the learning curves of the pre-training (see Figure 3.4), we can see the loss of the value network showed a decreasing trend in the first stage, indicating the network architecture is capable of learning the Rosetta score function. The loss of policy networks, on the other hand, didn't show a decreasing trend, mainly due to the non-linearity of the action concerning the observation. When a collision or void happens, the "correct" move is to twist the torsion angles in the loop between the two chains to make the chains get further or closer as a whole (see Figure 3.5). However, according to our representation of the state space, the activation signal, such as high repulsive energy terms, can only be observed around the collision site. That makes the prediction of "correct" actions extremely difficult. Possible solutions to this problem include a more elaborated action space setting (*e.g.* move the segment around the site as a whole instead of a single torsion angle), or a better featurization of protein conformation (*i.e.* including the first derivative of the energy as features). Besides that, the second reason is the limitation in the training data. As we only move one torsion angle from the native conformation when we created the dataset, only the last steps in the whole folding trajectories were included.

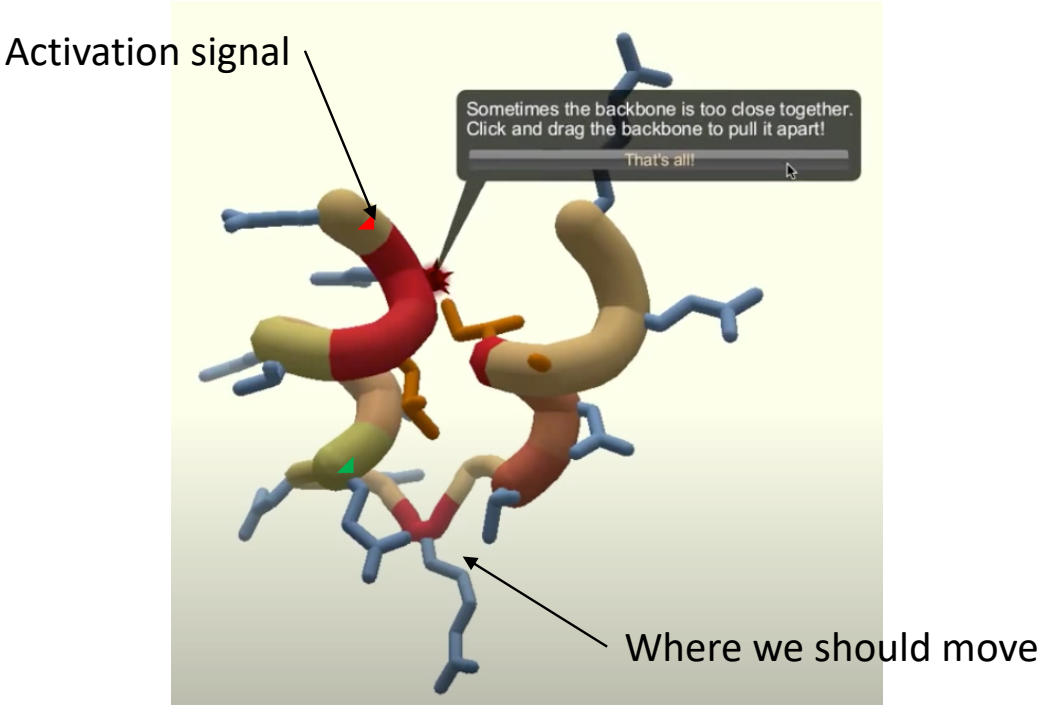
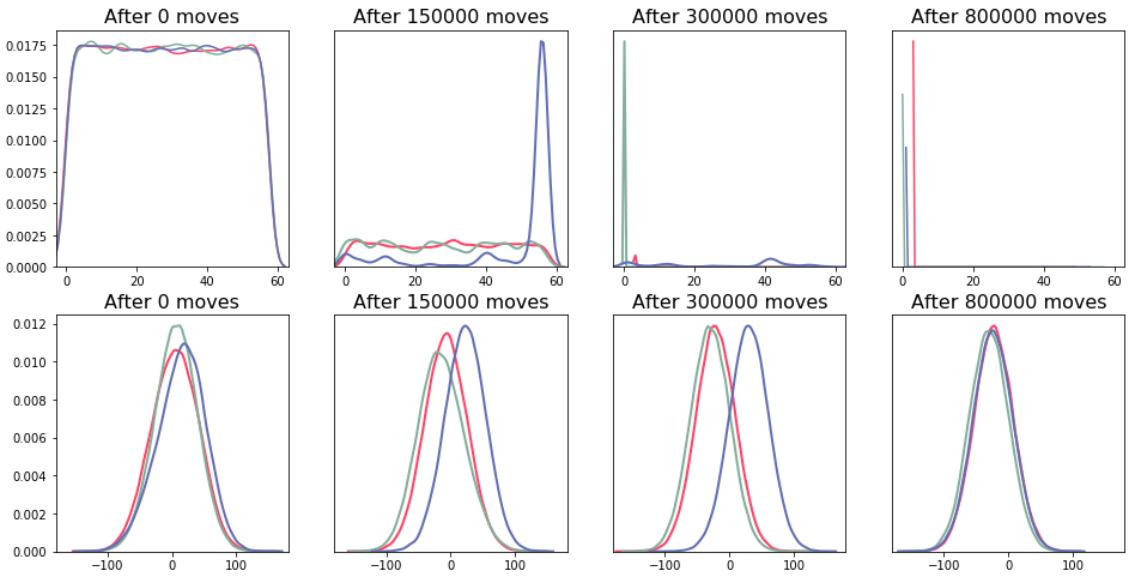


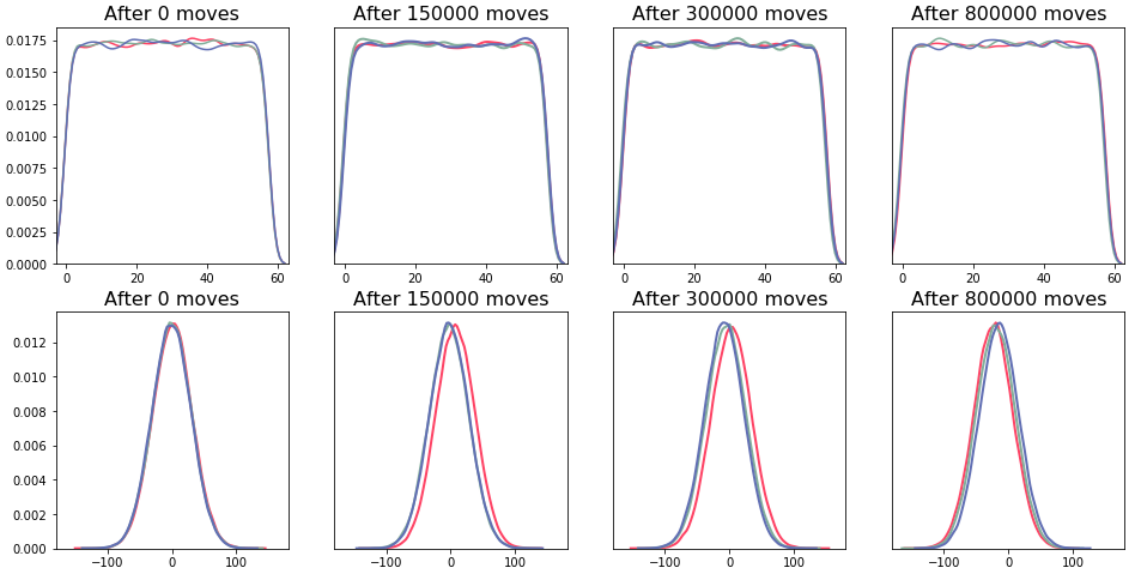
Figure 3.5: The non-linearity of observation and action The red circle represents the collision of the two chains. The green circle represents the correct move that lead to a relaxed conformation. The original figure comes from the screenshot of the Foldit game.

3.4 Training Process

To further analyze what the agent learns from the protein folding environment, we plot the predicted action distribution of policy networks during the training process (see Figure 3.6). For the agent starting from random parameters, we can see that the predicted torsion index is distributed uniformly at the beginning of the training process. As the training progresses, the predicted torsion index begins to concentrate on one side of the amino acid chain. After convergence, the policy network almost only predicts one residue near



(a) RL start from random parameter



(b) RL start from pre-trained parameter

Figure 3.6: Agent training progression.
The distribution of action choice during the training on 1COI. In each sub-figure, the first rows show the distribution of prediction of torsion angles (index ranging from 0 to 57); the second rows show the distribution of predicted value to change. a. is the curves of RL trained on 1COI from random parameters. b. is the curves of RL trained on 1COI from pre-trained parameters. Each run includes three independent trajectories in different color.

termini. Meanwhile, the distribution of the predicted angle value to change doesn't show an obvious trend during training, which means the agents converge by learning to only move residues near to termini after convergence so that it will not disturb the core structure.

Chapter 4

Conclusion and Outlook

In this work, we targeted the protein folding problem and formulated it as an MDP. We introduced a protein folding learning environment based on a backbone torsion moving setting and applied a baseline RL algorithm to solve it. The RL agents performed comparable to a conventional MC approach in multiple tasks and learned to only move terminal residues without disturbing the core structure after convergence, showing that the idea of applying the RL algorithm to solve the protein folding problem is feasible. The RL agents' convergence to local minima is partially due to the lack of exploration, and this problem can be relieved by methods such as adding intrinsic reward [45] or simulating multiple trajectories synchronously. The pre-trained agents failed to converge in the experiments due to multiple reasons. The way we created expert policy data only included one step before native structures, and the

relationship between the current features and the outputs is extremely complicated. We need to either change the current action setting or change the featurization of protein to make the problem learnable.

Although we only tested our protocol in the coarse protein folding problem, this RL protocol can be generalized to solve a wide range of protein modeling and design problems by changing the environment setting. For example, actions only moving a loop region would lead to a loop modeling environment; actions moving a ligand-protein would lead to a rigid-body docking environment; actions changing the amino acid sequence given a structure would lead to a fixed backbone design environment. Also, because of the immense search space and the complex reward surface, the domain of protein modeling poses challenging but scientific meaningful problems for the RL algorithms. Therefore, we expect to see more exploration of applying RL algorithms in protein modeling and more powerful algorithms that can solve the structure both quickly and accurately.

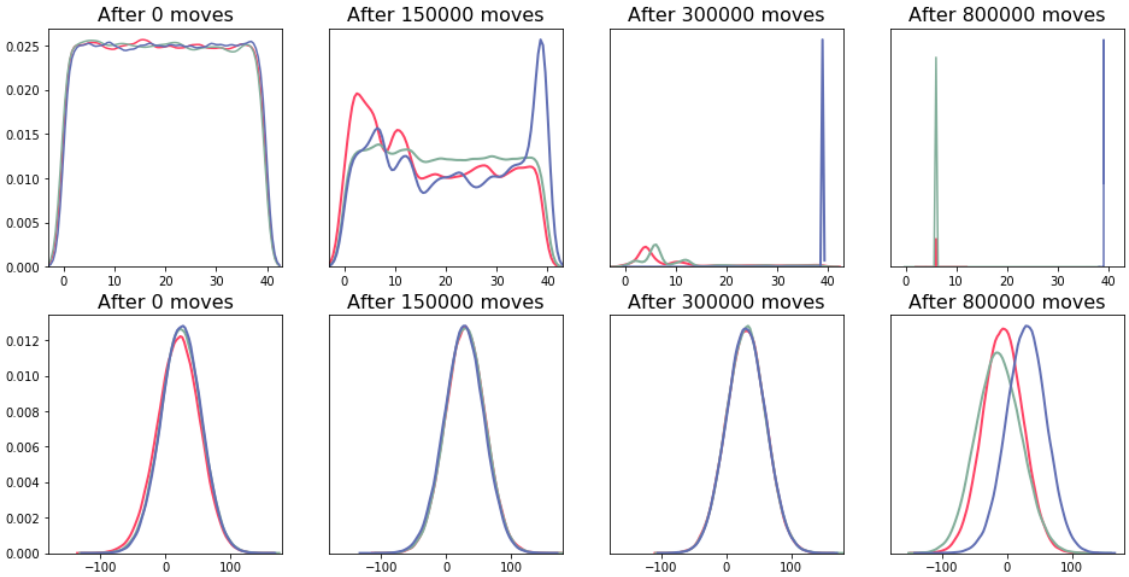
Appendix A

Code Availability

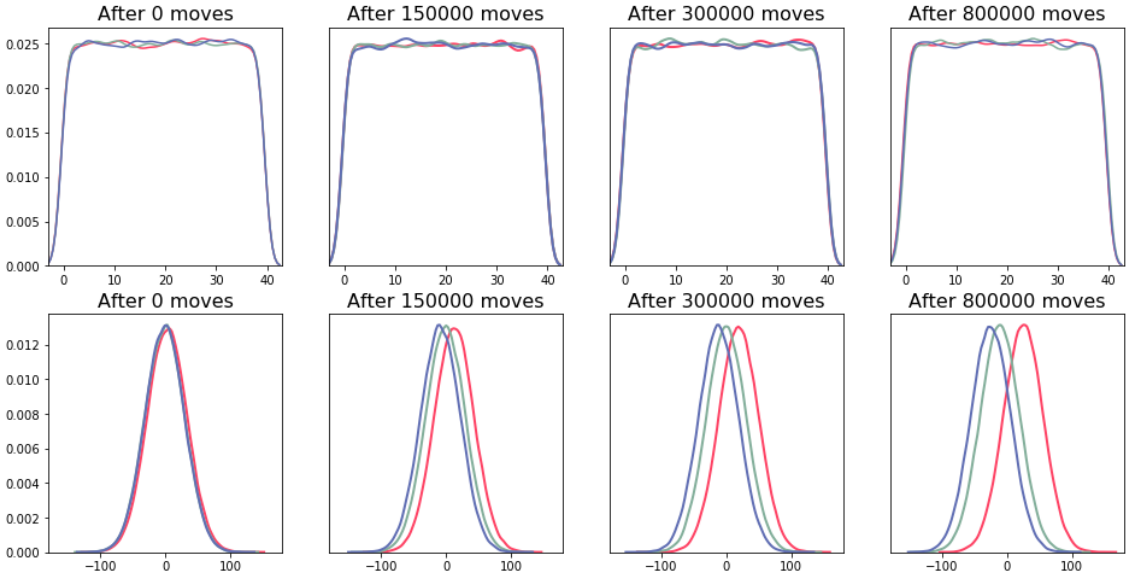
All code and data can be found at `https://github.com/wenhao-gao/Protein_Folding_Env`.

Appendix B

Supporting Information

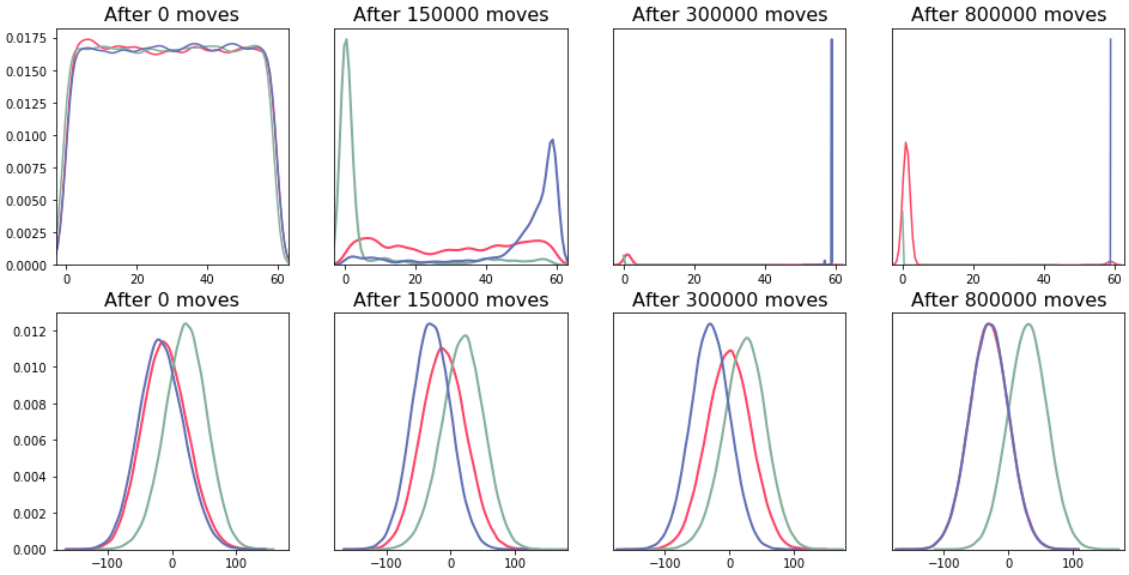


(a) RL start from random parameter

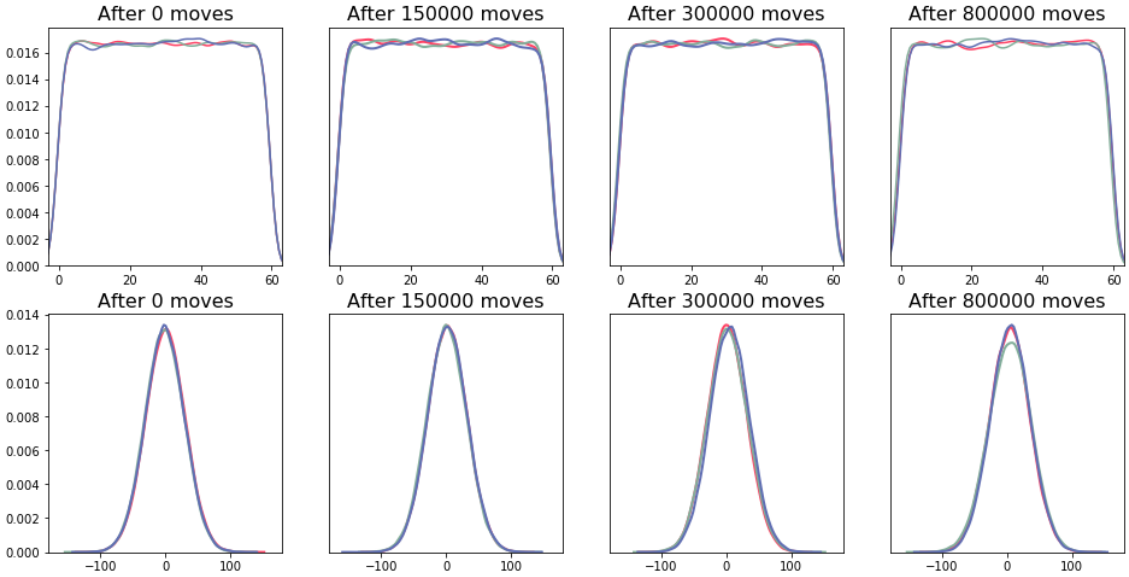


(b) RL start from pre-trained parameter

Figure B.1: Agent training progression.
The distribution of action choice during the training on 1L2Y. In each sub-figure, the first rows show the distribution of prediction of torsion angles (index ranging from 0 to 39); the second rows show the distribution of predicted value to change. a. is the curves of RL trained on 1L2Y from random parameters. b. is the curves of RL trained on 1L2Y from pre-trained parameters. Each run includes three independent trajectories in different color.



(a) RL start from random parameter



(b) RL start from pre-trained parameter

Figure B.2: Agent training progression.
The distribution of action choice during the training on 1QGM. In each sub-figure, the first rows show the distribution of prediction of torsion angles (index ranging from 0 to 59); the second rows show the distribution of predicted value to change. a. is the curves of RL trained on 1QGM from random parameters. b. is the curves of RL trained on 1QGM from pre-trained parameters. Each run includes three independent trajectories in different color.

Bibliography

[1] B. Kuhlman and P. Bradley, "Advances in protein structure prediction and design," *Nature Reviews Molecular Cell Biology*, vol. 20, no. 11, pp. 681–697, 2019.

[2] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, no. 4096, pp. 223–230, 1973.

[3] C. Levinthal, "Are there pathways for protein folding?" *Journal de Chimie Physique*, vol. 65, pp. 44–45, 1968.

[4] R. F. Alford, A. Leaver-Fay, J. R. Jeliazkov, M. J. O'Meara, F. P. DiMaio, H. Park, M. V. Shapovalov, P. D. Renfrew, V. K. Mulligan, K. Kappel *et al.*, "The rosetta all-atom energy function for macromolecular modeling and design," *Journal of Chemical Theory and Computation*, vol. 13, no. 6, pp. 3031–3048, 2017.

[5] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and

E. Teller, “Equation of state calculations by fast computing machines,” *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[6] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[7] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult, “Critical assessment of methods of protein structure prediction (casp)–round xiii,” *Proteins: Structure, Function, and Bioinformatics*, 2019.

[8] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[9] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing,” *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, p. 354, 2017.

[11] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu, “Accurate de novo prediction of protein contact map by ultra-deep learning model,” *PLoS Computational Biology*, vol. 13, no. 1, p. e1005324, 2017.

[12] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, pp. 1–5, 2020.

[13] J. Yang, I. Anishchenko, H. Park, Z. Peng, S. Ovchinnikov, and D. Baker, "Improved protein structure prediction using predicted inter-residue orientations," *bioRxiv*, p. 846279, 2019.

[14] W. Gao, S. P. Mahajan, J. Sulam, and J. J. Gray, "Deep learning in protein structural modeling and design," *arXiv preprint arXiv:2007.08383*, 2020.

[15] J. A. Ruffolo, C. Guerra, S. P. Mahajan, J. Sulam, and J. J. Gray, "Geometric potentials from deep learning improve prediction of cdr h3 loop structures," *bioRXiv*, p. 940254, 2020.

[16] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *Advances in Neural Information Processing Systems*, pp. 6410–6421, 2018.

[19] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae *et al.*, "Chip placement with deep reinforcement learning," *arXiv preprint arXiv:2004.10746*, 2020.

[20] G. Czibula, M.-I. Bocicor, and I.-G. Czibula, "A reinforcement learning model for solving the folding problem," *International Journal of Computer Technology and Applications*, vol. 2, no. 01, 2011.

[21] Y. Li, H. Kang, K. Ye, S. Yin, and X. Li, "Foldingzero: Protein folding from scratch in hydrophobic-polar model," *arXiv preprint arXiv:1812.00967*, 2018.

[22] R. Jafari and M. M. Javidi, "Solving the protein folding problem in hydrophobic-polar model using deep reinforcement learning," *SN Applied Sciences*, vol. 2, no. 2, p. 259, 2020.

[23] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović *et al.*, "Predicting protein structures with a multiplayer online game," *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.

[24] B. Koepnick, J. Flatten, T. Husain, A. Ford, D.-A. Silva, M. J. Bick, A. Bauer, G. Liu, Y. Ishida, A. Boykov *et al.*, "De novo protein design by citizen scientists," *Nature*, vol. 570, no. 7761, pp. 390–394, 2019.

[25] G. N. Lewis, "A new principle of equilibrium," *Proceedings of the National*

Academy of Sciences of the United States of America, vol. 11, no. 3, p. 179, 1925.

[26] A. Antos, C. Szepesvári, and R. Munos, "Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, no. 1, pp. 89–129, 2008.

[27] K. T. Simons, I. Ruczinski, C. Kooperberg, B. A. Fox, C. Bystroff, and D. Baker, "Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins," *Proteins: Structure, Function, and Bioinformatics*, vol. 34, no. 1, pp. 82–95, 1999.

[28] J. K. Leman, B. D. Weitzner, S. M. Lewis, J. Adolf-Bryfogle, N. Alam, R. F. Alford, M. Aprahamian, D. Baker, K. A. Barlow, P. Barth *et al.*, "Macromolecular modeling and design in rosetta: recent methods and frameworks," *Nature Methods*, pp. 1–14, 2020.

[29] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola, "Generative models for graph-based protein design," *Advances in Neural Information Processing Systems*, pp. 15 820–15 831, 2019.

[30] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in Neural Information Processing Systems*, pp. 6530–6539, 2017.

[31] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[32] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[33] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[34] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[37] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmas-

ter level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[38] I. Sillitoe, N. Dawson, T. E. Lewis, S. Das, J. G. Lees, P. Ashford, A. Tolulope, H. M. Scholes, I. Senatorov, A. Bujan *et al.*, "Cath: expanding the horizons of structure-based functional annotations for genome sequences," *Nucleic Acids Research*, vol. 47, no. D1, pp. D280–D284, 2019.

[39] J. W. Neidigh, R. M. Fesinmeyer, and N. H. Andersen, "Designing a 20-residue protein," *Nature structural biology*, vol. 9, no. 6, pp. 425–430, 2002.

[40] W. Vranken, Z. Chen, P. Xu, S. James, H. Bennett, and F. Ni, "A 30-residue fragment of the carp granulin-1 protein folds into a stack of two β-hairpins similar to that found in the native protein," *The Journal of peptide research*, vol. 53, no. 5, pp. 590–597, 1999.

[41] N. L. Ogihara, M. S. Weiss, D. Eisenberg, and W. F. Degrado, "The crystal structure of the designed trimeric coiled coil coil-vald: implications for engineering crystals and supramolecular assemblies," *Protein Science*, vol. 6, no. 1, pp. 80–88, 1997.

[42] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *arXiv preprint arXiv:1908.00261*, 2019.

[43] J. Bhandari and D. Russo, "Global optimality guarantees for policy gradient methods," *arXiv preprint arXiv:1906.01786*, 2019.

[44] I. Osband, B. Van Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions." *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019.

[45] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning," in *Advances in neural information processing systems*, 2005, pp. 1281–1288.

[46] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

Vita

Wenhao Gao grew up in Beijing, China. Before his coming to Johns Hopkins University, he obtained his B.S. in Chemistry from Peking University. In Johns Hopkins University, he majored in Chemical and Biomolecular Engineering and endeavored to apply artificial intelligence technology to automate the chemical and biological discovery process. Under the supervision of Dr. Jeffrey J. Gray and Dr. Jeremias Sulam, he completed this thesis about applying reinforcement learning algorithm to fold proteins. After finishing his Master's degree, Wenhao will further pursue a PhD in Chemical Engineering in Massachusetts Institute of Technology.