Modelling Somatic Hypermutations with Reinforcement Learning for PD-1 and Pembrolizumab

submitted by

Anmol Singh (225HSBB004)

to

Institute of Bioinformatics and Applied Biotechnology

in partial fulfilment of the requirements for

Master of Science in Biotechnology and Bioinformatics

(degree awarded by **Bangalore University**, Bengaluru)

under the guidance of

Dr. Nithya Ramakrishnan

Prof. Subha Srinivasan

www.ibab.ac.in

Biotech Park, Electronic City Phase I

Bengaluru - 560100

DECLARATION BY THE STUDENT

I hereby declare that the thesis "*Modelling Somatic Hypermutations with Reinforcement Learning for PD-1 and Pembrolizumab*" is a bona fide and genuine research work carried out by me, between 23rd December 2023 and 4th June 2024 at IBAB, Bengaluru, under the guidance of *Dr Nithya Ramakrishnan*, Assistant Professor, Information Theory, Algorithms and Machine Learning in Biology and *Prof. Subha Srinivasan*, IBAB Chair, Genomics.

Date: 03.06.2024

Place: IBAB, Bengaluru

Anmol Singh

Anmol Singh

CERTIFICATE BY THE SUPERVISOR

This is to certify that the thesis "*Modelling Somatic Hypermutations with Reinforcement Learning for PD-1 and Pembrolizumab*" represents research work done by *Mr. Anmol Singh* in partial fulfillment of the requirements for M.Sc. in Biotechnology and Bioinformatics at IBAB, Bengaluru, under our guidance.

Date: 03/06/2024

Place: Bangalore

Dr Nithya Ramakrishnan

Prof. Subhashini Srinivasan

ACKNOWLEDGMENTS

I express my deepest gratitude and special thanks to my guides Dr Nithya Ramakrishnan and Prof. Subhashini Srinivasan who took time to hear me out, guide and keep me on the correct path, and allowed me to bring it to a good conclusion. Dr Nithya Ramakrishnan helped me regarding the reinforcement learning, while Prof. Subhashini Srinivasan provided guidance related to structural biology and protein-protein interactions, without which the project would not have a base to build upon.

I express my thanks to the lab members of the research groups of my guides, especially Mr. Balakrishna, Miss Namita, Miss Apoorva, and Mr. Yash who helped me during the project and gave important and necessary advice to overcome the problems I faced in the project and life during this time.

I thank all the professors who helped me acquire the required knowledge and understanding of biology, mathematics, and computer science which helped me to confidently work on this project. I especially thank Prof. R Srivatsan, our mathematics, statistics, and R programming professor, for his guidance and teachings during my course.

I would also like to thank my mother for always listening to my rambles about this project (and other topics) and by explaining my work to her, I have acquired a better understanding of reinforcement learning and structural biology. Needless to say, I would not have the courage to pursue my postgraduate course if not for her support. This thesis is dedicated to her.

This work was partially supported by the Department of Electronics, IT, BT, and S&T of the Government of Karnataka and I am grateful for their support.

ABBREVIATIONS

RL	Reinforcement Learning
SHM	Somatic Hypermutations
PD-1	Programmed Cell Death Protein-1
Pembro	Pembrolizumab
DQL	Deep Q-Learning
ANN	Artificial Neural Network
PRODIGY	PROtein binDIng enerGY prediction
BA	Binding Affinity
AF	AlphaFold2
RMSD	Root Mean Square Deviation
aa	Amino Acids

ABSTRACT

The process of engineering antibodies with high affinity towards an antigen is very expensive and time-consuming in traditional wet approaches. Bioinformatic Tools and computational approaches can be and have been used for antibody design to reduce the time and cost. Reinforcement learning is similar to natural selection, as in by trial-and-error method, the better actions (mutations) remain in the population and those harmful are removed over time.

We present a reinforcement learning (RL) model to mimic Somatic Hypermutations (SHM), a natural process for selecting high affinity antibodies. In this model, the agent can learn to preferentially mutate interacting amino acids in the antibody, leading to affinity maturation. The model has the potential to select a higher binding affinity antibody than the initial antibody-antigen complex. We have used the Pembrolizumab-PD-1 (5b8c) complex to optimize the RL model.

We use Q-Learning in RL to model SHM on a reduced state space to provide better binding affinity antibodies. We validated the structure of the mutated antibodies selected by the RL model using AlphaFold2 and C-alpha distance plots to check for proper folding of chains and protein-protein interactions. This study provides a proof of concept that RL can be used for modelling the biological process of SHM and can further be employed for creating novel antibodies.

Keywords: Antibody prediction, Q-learning, Deep Q-Learning, Deep Reinforcement Learning, AlphaFold2, protein-protein interaction, immunotherapy

TABLE OF CONTENTS

List of Figures

Figure 22. A. AF predicted structure of state DRRTSYTYYNSNTNRYR aligned with Pembro (control) state, **B.** Cα distance plot of intra chain A and chain B, **C.** Cα distance plot of inter chain A vs chain B, **D.** Cα distance plot of inter chain A vs chain C and chain B vs chain C

List of tables

1. INTRODUCTION

In this section, we introduce the major approaches and address the following specific questions:

- What are somatic hypermutations? Why are they important for our immune system?
- What is immunotherapy and how is this study related to it? Where is Pembro used in immunotherapy?
- What is reinforcement learning? Why is it a good strategy for the simulation of SHM?

1.1 Somatic Hypermutations

The B cells express cell-surface receptors called immunoglobulins (Ig). These Ig consists of two heavy chains and two light chains. These polypeptides are encoded in three Ig loci, the heavy chain (IgH), the κ-light chain (Igκ), and the λ-light chain (Igλ). These loci consist of variable and constant(C) regions. The variable region of the heavy chain is composed of variable(V), diversity(D), and joining(J) genes. On the other hand, the light chain only has V and J genes (Odegard and Schatz, 2006).

Our Immune system is well-equipped to produce Ig of high specificity towards foreign substances, called antigens. This ability is the result of somatic recombination of a small set of gene segments, called V(D)J recombination. This process can produce around 10^7 different antibody specificities. But, the antibodies created by V(D)J recombination only bind to the antigens by modest affinity, there is a need to improve the resultant Ig to make it bind to the antigens with high affinity and specificity (Papavasiliou and Schatz, 2002).

The diversification of Ig is caused by two distinct diversification processes, class switch recombination (CSR), and the somatic hypermutations (SHM) (Odegard and Schatz, 2006).

The SHM introduces point mutations on the variable regions, the antibodies with higher affinity for the antigen will proliferate and survive. With successive cycles of mutations and proliferation of selected B cells, this results in high-affinity antibodies. This process is called affinity maturation (Papavasiliou and Schatz, 2002). The mutations in SHM are mainly point mutations, but insertions and deletions are also observed sometimes (Odegard and Schatz, 2006).

In our proposed model, only point mutations at the peptide level are taken into consideration.

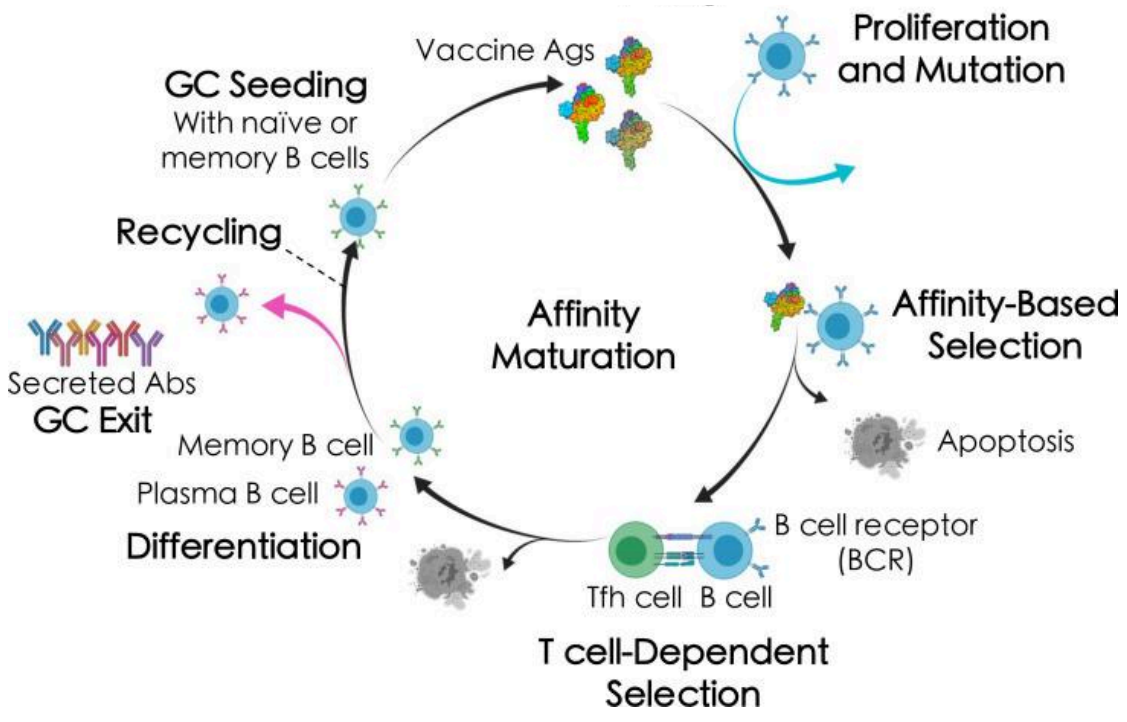


Figure 1. A broad overview of the affinity maturation (AM) process by which antibodies (Abs) evolve in a germinal center (GC) reaction (Faris J et al., 2022)

Assumptions:

We make the following assumptions at the start of our study:

- The relative configuration of the PD-1-Pembro complex remains the same irrespective of the type of amino acids mutations at the 17 positions
- The 3D folds of light and heavy chains are not disrupted by the mutations in loops on Pembro.

1.2 Immunotherapy and Pembrolizumab

Immunotherapy is a technique to utilize a patient's immune system to target cancer cells and destroy them. It is one of the widely used techniques in cancer treatment. Even though this is a revolutionary technique, there are still many challenges in the clinical scenarios. Since the tumor-host interactions are heterogeneous, the immunotherapy response can differ. The tumor microenvironment (TME) can also affect the immunotherapeutic response and immune evasion.

There are multiple types of immunotherapies used currently. These include immune checkpoint inhibition (ICI), adoptive cellular therapy (CAR T-cell therapy), and cancer vaccination. Pembro comes under ICI and is explained in more detail below.

The T cells contain evolutionarily conserved regulatory markers that are like checkpoints to regulate the activation of T cells. After the early activation, the T cells upregulate the inhibitory receptor cytotoxic T lymphocyte antigen 4 (CTLA4) and then to programmed cell death 1 (PD-1) which bind to ligands B7-1, B7-2, and PD-L1 or PD-L2. These ligands are presented by tumor cells, myeloid cells, regulatory T cells (Tregs), and antigen-presenting cells (APCs), which reduce cytotoxic T-cell activation, resulting in immune suppression and tumor growth. However, after treatment with ICI, inhibition is

released and cancer cells are targeted and destroyed by the primed and activated cytotoxic T cells.

Pembro, an IgG4 anti-PD-1 checkpoint inhibitor antibody, was one of the first FDA-approved therapy drugs for melanoma. Pembro was very successful in melanoma patients and is still used in ICI therapy. We are using Pembro as a model IgG antibody to perform SHM and possibly find a higher affinity antibody (Peterson et al., 2022).

1.3 Binding Affinity

We use binding affinity (BA) as a measure to evaluate the antibody and antigen interactions quantitatively and qualitatively.

We generalize our Ab-Ag interactions to protein-protein interaction (PPI) so we can understand them better. Given the complex structure, these interactions can also be explained as a function of atomic distances and characteristics, but this does not model the behavior accurately. It is the affinity of these interactions under various biological conditions, such as temperature, pH, and protein and substrate concentration that are critical in describing the interactions.

The binding affinity is explained in terms of the equilibrium dissociation constant (K_d), which can either be measured experimentally at the equilibrium of the reaction or be derived from the Gibbs free energy of dissociation ΔG and the reaction kinetics. Experimentally, several methods such as surface plasmon resonance (SPR), isothermal calorimetry (ITC), and titration by fluorescence apply if K_d is in the micromolar to nanomolar range. Although these methods have limitations. Normally, two or more methods are used under similar conditions to determine the K_d. Since in our model, we do not have a dataset of earlier experimentally determined results, computational methods of estimating the binding affinity from the given high-resolution structure is the only option.

These methods may use several parameters into account including the intermolecular non-covalent contacts (Kastritis P et al., 2011), and also the non-interacting surface (Kastritis P et al., 2014) and create linear, nonlinear models or machine learning models to estimate the binding affinity and compare them with the experimentally determined values to validate if their models work.

1.4 Reinforcement Learning

1.4.1. Basic Overview

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns what to do in an environmental condition to maximize an expected reward. It is described as a set of sequential Markov decision processes (MDP). The basic idea is that an agent that is interacting with an environment over time can sense the state of the environment and take actions to affect the state to achieve a goal over time **t**. Markov decision processes have multiple components: an agent, the environment, the state of the environment, the action taken by the agent, and the reward. We use **S** for a set of states, **A** for a set of actions, and **R** for a set of rewards (Sutton, R. S. and Barto A. G., 2018).

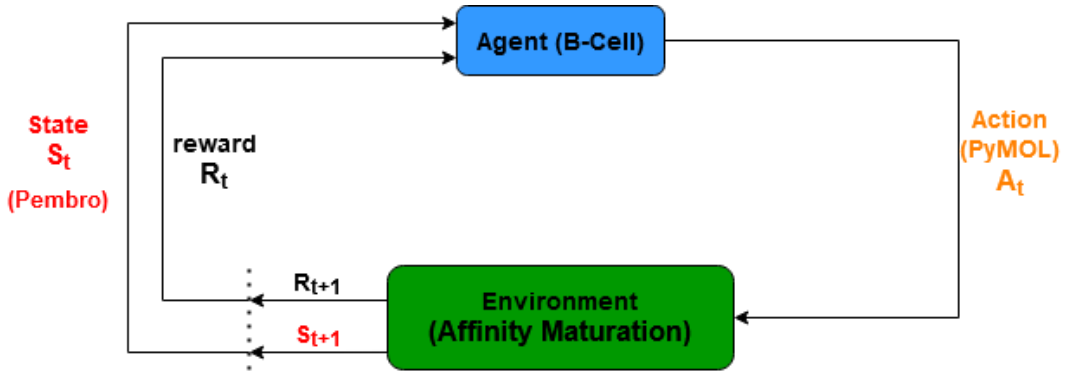


Figure 2. The interaction of agent and environment in a Markov decision process (MDP) for SHM/AM model (adapted from Sutton, R. S. and Barto A. G., 2018)

It is important to note that an agent aims to maximize the expected return of rewards. For our study the MDP consists of:

- ***Agent***: B-cells
- ***Action***: Point mutations on Pembro (using PyMOL)
- ***Environment***: Affinity Maturation of Pembro-PD-1
- ***State***: Pembro with mutated amino acids

1.4.1.1. Discounted Reward

We get the discounted return of rewards ($\mathbf{G_t}$) to account for the disparity between immediate and future rewards. This way the agent will care about immediate rewards more than the future rewards (Sutton, R. S. and Barto A. G., 2018):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad - \; eq.\ (1)$$

This can be explained in terms of rewards at successive time steps:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots G_t \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots)\, G_t \\ &= R_{t+1} + \gamma G_{t+1} \quad - \; eq.\ (2) \end{aligned}$$

Where, $\gamma < 1$.

1.4.1.2. Policies and Value Functions

In brief, policies address the problem of action selection in any given state whereas value functions tell us how good a given action or state is for an agent. Policies are simple functions that govern how the agent picks an action. The value functions are then defined according to the policies (Sutton, R. S. and Barto A. G., 2018).

1.4.1.3. State-Value Function

For a policy, π the state value function v_π tells us how good the state s is for an agent following the policy π.

$$v_\pi(s) = E_\pi[G_t | S_t = s]\; v_\pi(s) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\,] \quad - \; eq.\ (3)$$

1.4.1.4. Action-Value Function

Similar to v_π the action-value function q_π tells us for a given policy, how good it is for the agent to take any given action (**a**) in a given state(**s**) at time(**t**).

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]\ q_\pi(s,a) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \quad -eq.\ (4)$$

This is also called the Q-function and provides a Q-value and is an important component of Q-Learning (Sutton, R. S. and Barto A. G., 2018).

1.4.1.5. Optimal Policy and Value Functions

Briefly, a policy π is considered to be optimal than another policy π' if,

$$\pi \geq \pi' \ if\ and\ only\ if\ v_\pi(s) \geq v_{\pi'}(s)\ for\ all\ s \in S.$$

Here $v_\pi(s)$ stands for the expected return for starting in a state s and then following the policy π.

The optimal value functions are defined as:

Optimal State-Value function: $v_*(s) = \max_\pi v_\pi(s) \quad -eq.\ (5)$

Optimal Action-Value function: $q_*(s,a) = \max_\pi q_\pi(s,a) \quad -eq.\ (6)$

For all $s \in S$ and $a \in A(s)$**.**

One fundamental property q_* is that it must follow the *Bellman optimality equation* (Sutton, R. S. and Barto A. G., 2018) as follows:

$$q_*(s,a) = E[R_{t+1} + \gamma \max_{a'} q_*(s', a')] \quad -eq.\ (7)$$

1.4.2. Basic Q-Learning

The basic Q-learning algorithm to find the optimal policy is by finding the optimal Q-values for each state-action pair. This is achieved by utilizing the Bellman optimality equation to find the optimal q_*. We iterate through all the states and fill the Q-table which consists of the Q-value for the state-action pairs.

The Q-value is updated according to the learning rate(α), i.e. as the simulation goes on, the agent learns from the state-action pairs it has observed. This is calculated according to the given formula (Sutton, R. S. and Barto A. G., 2018):

$$q^{new}(s,a) = (1-\alpha)q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right) \quad \text{– eq. (8)}$$

Here we can observe that in this Q-value update equation, the learned value is the same as the Bellman optimality equation.

1.4.2.1. *Exploration methods*

One important part of any reinforcement algorithm is how to define the exploitation and exploitation are defined. Exploration is simply the agent trying to explore the environment. On the other hand, exploitation is taking action based on the policy, thus taking the best or one of the best actions in a given state. In terms of Q-learning, exploration is taking a random selection of actions whereas exploitation is utilizing the learned Q-values to determine the best action for a given state, i.e. exploitation of the Q-table. There are different exploration-exploitation methods. Let us discuss two of them in brief.

1.4.2.2. Epsilon greedy Strategy

Here, we define an exploration rate ϵ which is initially set to 1(100%). Then in each episode, or intervals of episodes, we reduce this epsilon according to a decay rate. As the epsilon decreases, the agent slowly shifts from exploration of the environment to the exploitation of the Q-table (Sutton, R. S. and Barto A. G., 2018).

We do this by picking a random number in each instance and then comparing it with the exploration rate. If the random number is greater than epsilon then we go for exploitation, else we go for exploration.

The problem with the epsilon greedy method is that it cannot distinguish between two good actions/Q-values. It always picks the highest Q-value during exploitation. To overcome this there are other exploration methods, such as Soft-max strategy.

1.4.2.3. Soft-max (or Boltzmann) Strategy

To overcome the shortcomings of the epsilon greedy method during exploitation, we can convert the Q-values into a Boltzmann distribution and then pick actions according to it. Here a hyperparameter temperature(τ) is introduced.

$$P_t(a) = \frac{exp(Q_t(a)/\tau)}{\sum_{i=1}^{n} exp(Q_t(i)/\tau)} \quad \text{– eq. (9)}$$

We start with a high temperature which corresponds to exploration and as we reduce the temperature the agent starts to exploit the Q-table (Sutton, R. S. and Barto A. G., 2018).

1.4.2.4. *Limitations of Q-learning*

Basic Q-learning is good when dealing with small state space of up to a few hundred or thousands, but as we increase the number of possible states the agent can

observe over time, the algorithm starts to demand more exploration and thus takes much more time to converge to an optimal final state.

Most real-world problems have a very large state space, thus basic Q-learning is not effective for problems. To find solutions to such problems, we need to consider many other reinforcement learning approaches such as policy gradient methods, actor-critic, model-based methods, and Deep Q Networks

1.5 Deep Reinforcement Learning (DRL)

When we talk about deep learning we talk about utilizations of different Artificial Neural Networks (ANN). There are many types of deep reinforcement techniques but here only Deep Q-Learning (DQL), which was implemented in the work, is described.

1.5.1. Artificial Neural Networks (ANN)

ANN mainly consists of three types of neural layers, a layer of "input", then the "hidden" layers, and lastly the output layer. There are multiple types of ANN, some of the most commonly used ones are feedforward NN (FFNN) and convolutional neural network (CNN). For our model, we are working with FFNN. For Deep Q-Learning, the most commonly used type is CNN, which was introduced to learn Atari games (Mnih V et al., 2013).

FFNN are the simplest types of ANN, the steps included in training a FFNN are:

1. Create a FFNN with randomized weights
2. Do a forward pass through the network and get the predicted value
3. Calculate the loss of predicted and expected value using functions such as mean square error (MSE)

4. Do backpropagation through the network to get the slope of loss vs weights; there can be an optimization algorithm which takes care of this such as Adam (Kingma D, 2017)
5. Do a gradient descent to find the nearest minima and adjust the weight accordingly

1.5.2. Deep Q Learning (DQL)

For DQL we are replacing our policy equation[ref] with an ANN, we call this a Deep Q-Network (DQN), which is now used to approximate optimal Q-values. This algorithm was explained by Mnih V et al., 2013, to create a DQL model to play Atari games, and has been used in many other complex games such as DOTA2 and GO. CNN has been used for deep learning algorithms for complex datasets such as images and protein structures to be analyzed. The detailed algorithm of the DQL is explained in the following sections.

DQL models have been made before for SHM and affinity maturation. Faris J et al., 2022, described a DQL model for vaccine development which consists of an affinity maturation (AM) model to simulate the effects of vaccines on AM.

1.6 Objectives

1.6.1. Major Objective

The main objective of this project is to create a reinforcement learning model for somatic hypermutations using Pembro and PD-1.

1.6.2. Minor Objective

We also aim to find states/antibodies that have better binding affinity than Pembro and validate the good states using AlphaFold2 and inter-residue distance plots of the predicted structures.

2. MATERIALS AND METHODS

2.1 Pembrolizumab

The crystal structure of Pembro (Pembro) was acquired from RCSB Protein Data Bank with ID: 5B8C. The structure contains multiple complexes of Pembro-PD-1, the complex consisting of chains A (light chain), B (heavy chain), and C (PD-1) was taken. It was then relaxed using GROMACS (version 2020.1-Ubuntu-2020.1-1).

Pembro was made by Keytruda, the experimentally derived binding affinity against hPD-1 is 29 pM (Patnaik A,2015) i.e. 14.4 kcal/mole. But the binding affinity for the relaxed 5B8C using PRODIGY was found to be 12.6 kcal/mole, we are taking this as a reference moving forward.

A

```
integrator      = steep ; cg,md
nsteps          = 2000
nstlist         = 10
rlist           = 1.0
coulombtype     = pme
rcoulomb        = 1.0
vdw-type        = cut-off
rvdw            = 1.0
nstenergy       = 10
emtol = 10
emstep = 0.01
```

B

```
integrator      = steep
nsteps          = 2000
nstlist         = 10
rlist           = 0.5
coulombtype     = cut-off
rcoulomb        = 0.5
vdw-type        = cut-off
rvdw            = 0.5
nstenergy       = 10
emtol           = 0.1
emstep          = 0.01
```

C

```
integrator      = cg
nsteps          = 2000
nstlist         = 10
rlist           = 0.5
coulombtype     = cut-off
rcoulomb        = 0.5
vdw-type        = cut-off
rvdw            = 0.5
nstenergy       = 10
emtol           = 0.1
emstep          = 0.01
```

Figure 3. Parameters of **A** ion.mdp, **B** em_steep.mdp and **C** em_cg.mdp

```
#!/bin/bash

state="$1"

gmx pdb2gmx -f "${state}.pdb" -o "${state}_processed.gro" -water spce -ignh

gmx editconf -f "${state}_processed.gro" -o "${state}_box.gro" -c -d 1.4 -bt cubic

gmx solvate -cp "${state}_box.gro" -cs spc216.gro -o "${state}_solvated.gro" -p topol.top

gmx grompp -f "base_files/ions.mdp" -c "${state}_solvated.gro" -p topol.top -o ions.tpr -maxwarn 3

echo "13" |gmx genion -s ions.tpr -o "${state}_solvated_ions.gro" -p topol.top -pname NA -nname CL -neutral

gmx grompp -f "base_files/em_steep.mdp" -c "${state}_solvated_ions.gro" -p topol.top -o "${state}_steep.tpr"

gmx mdrun -v -deffnm "${state}_steep"

gmx grompp -f "base_files/em_cg.mdp" -c "${state}_steep.gro" -p topol.top -o "${state}_cg.tpr"

gmx mdrun -v -deffnm "${state}_cg"

gmx trjconv -f "${state}_cg.gro" -o "${state}_final.pdb" -pbc mol -s "${state}_cg.tpr"
```

Figure 4. Bash script for running GROMACS for energy minimization

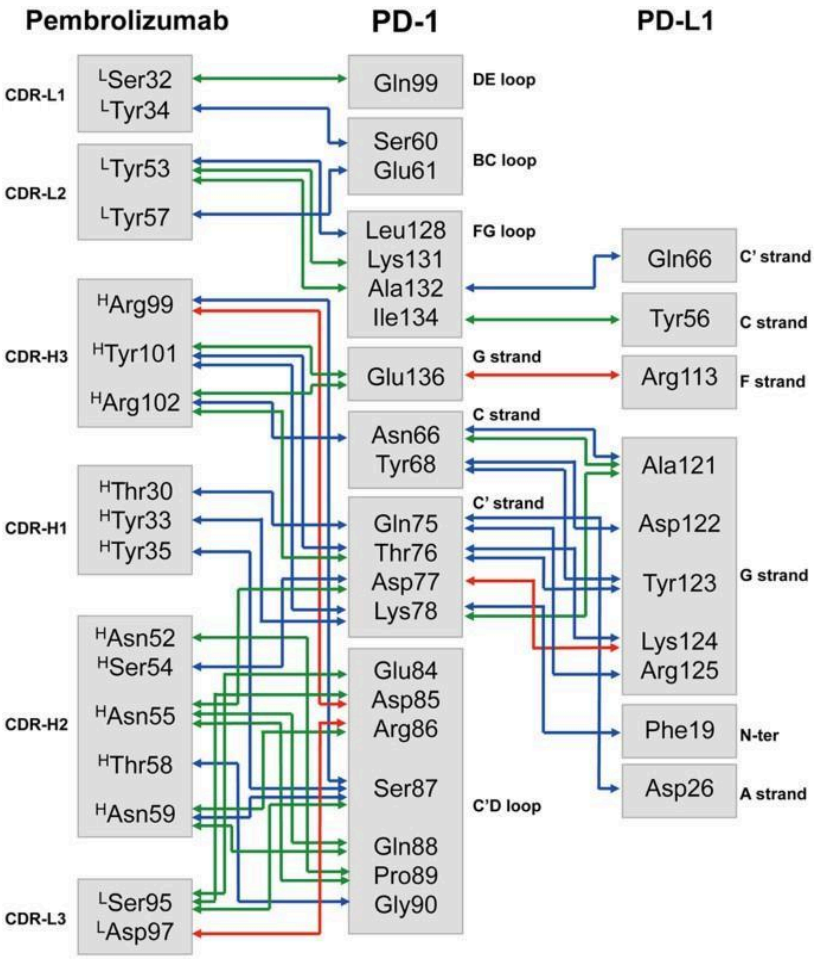


Figure 5. Interaction map of 17 residues in Pembro-PD-1 complex (5b8c). Direct protein/protein hydrogen bonds are in blue, water-mediated hydrogen bonds are in green, and salt bridges are in red (Horita S et al., 2016)

2.2 Binding Affinity Tools

Binding affinity (BA) can be determined experimentally, but it isn't feasible to do so, thus we use computational methods to estimate binding affinities. We consider the following factors in determining the binding affinity estimator To choose which method is best for our use depends on the time taken to estimate binding affinity, accessibility of the method (if it is available via web server or can be installed locally on local machines), accuracy and relevance of the method, and the metadata provided by the method. We compared multiple methods including PRODIGY (Xue L et al., 2016), LISA (Raucci R et al.,2018), CSM-AB (Myung Y et al., 2022), AREA-AFFINITY (Yang Y et al, 2023) and

DG-Affinity (Yuan Y et al,2023) by literature review and comparing them based on the parameters mentioned above before.

PRODIGY uses a linear model consisting of the constituent parameters including types of intermolecular non-covalent contacts and also non-interacting surfaces (NIS) to predict the binding affinity. LISA utilizes a non-linear model to estimate the protein-protein interactions. Both tools are considered good for protein-protein interactions (PPI) (Yang Y et al, 2023). They are locally non-covalent and fast. Since LISA has not been maintained and is not dependable for the future. PRODIGY is maintained (https://github.com/haddocking/prodigy) and also provides metadata on the type of contacts the proteins are making; we chose this tool for BA calculation.

CSM-AB, AREA-AFFINITY, and DG-Affinity are web server-based tools and are specifically made for Antibody and antigen interaction. But AREA-AFFINITY and DG-Affinity do not provide an API for easy accessibility. CSM-AB is slow and the web server goes off-service occasionally, which is problematic when running long simulations.

PRODIGY (version 2.1.3) was considered the best for our use case after the assessment of tools. The command used for PRODIGY:

$prodigy {pdb_file} --selection A,B C --temperature 25

```
[+] Parsed structure file ERIHRPAFLNHNAATVW_mutated (3 chains, 347 residues)
[+] No. of intermolecular contacts: 103
[+] No. of charged-charged contacts: 12
[+] No. of charged-polar contacts: 16
[+] No. of charged-apolar contacts: 30
[+] No. of polar-polar contacts: 7
[+] No. of apolar-polar contacts: 18
[+] No. of apolar-apolar contacts: 20
[+] Percentage of apolar NIS residues: 35.34
[+] Percentage of charged NIS residues: 24.10
[++] Predicted binding affinity (kcal.mol-1):    -12.9
[++] Predicted dissociation constant (M) at 25.0˚C:  3.7e-10
```

Figure 6. Example output from PRODIGY for a mutated state

The residues are categorized as follows in PRODIGY:

- *Polar*: C, H, N, Q, S, T, W
- *Apolar*: A, F, G, I, L, V, M, P, Y
- *Charged*: E, D, K, R

2.3 Basic Q-Learning in RL model

2.3.1. *Basic Workflow*

The basic pipeline for Q-Learning is as follows:

1. Initiate the Agent class with the action list
    a. Action list consists of all the possible actions
    b. Create an empty dictionary of Q-table with the action list
    c. Assign the hyperparameters such as learning rate, and exploration rate to recommended values
2. Initiate the simulation with the starting state of Pembro
3. Predict action through Epsilon greedy or Soft-max strategy
4. Perform the mutation in PyMOL (version 2.6.0a0 Open-Source, 2024-01-19) and save the state in a local PDB bank for simulations
5. Use PRODIGY (version 2.1.3) to get the interactions and binding affinity of the mutated state
6. Calculate the Score for the mutated state, and convert the score into discrete rewards. The score is generated by doing a dot product of input from PRODIGY and scoring vector.
    - Scoring Vector: [0.1062428, -0.1065163, 0.396131, -0.3537632, 0.6397215, -0.1923142, -0.3012518, -0.1059661, -0.1]; in some simulations, the -0.1 is changed to -0.5 or -2 (mentioned where it is done).
    - The scoring vector was extracted from Vangone A and Bonvin A, 2015 and altered to add van-der Waals as a negative reward, and charged-polar and apolar-apolar contacts were added according to their Pearson correlation with binding affinity.

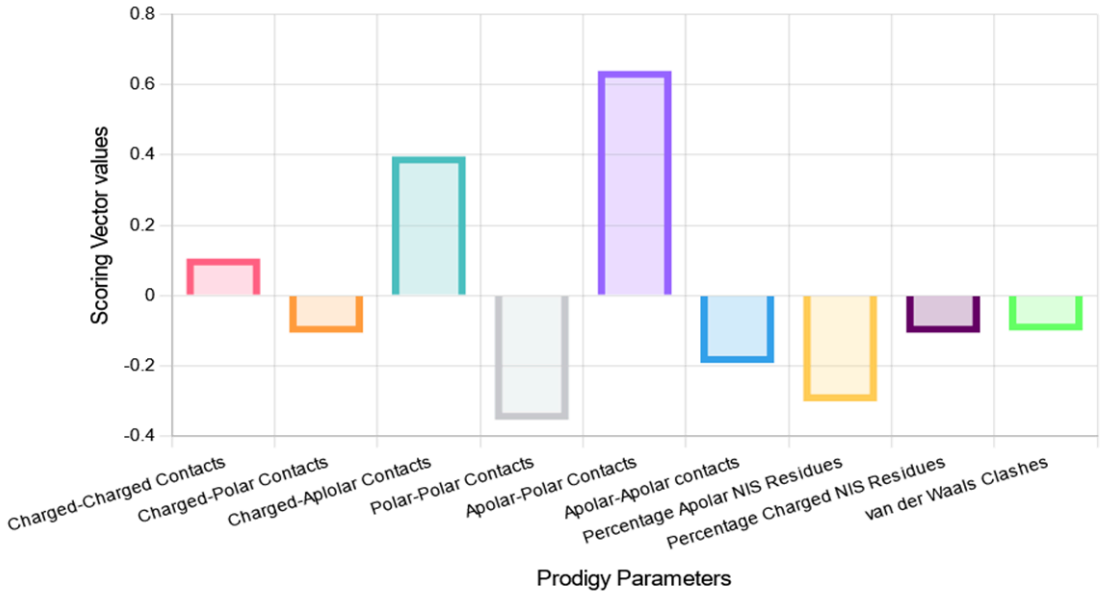


Figure 7. Scoring vector values of reward function

Table 1. Converting Score into reward

Score	Binding Affinity (kcal/mole)	Reward
>=3	< -12.6	10
>=3	>= -12.6	5
3>score>-1	<= -12.6	1
3 >score > -1	> -12.6	-1
-1>= score > -10	-	-1
< -10	-	-10

7. Update the Q-value according to the Q-value update equation (eq 8).
8. Iterate through steps 3 to 7 till the simulation reaches the maximum episode limit.

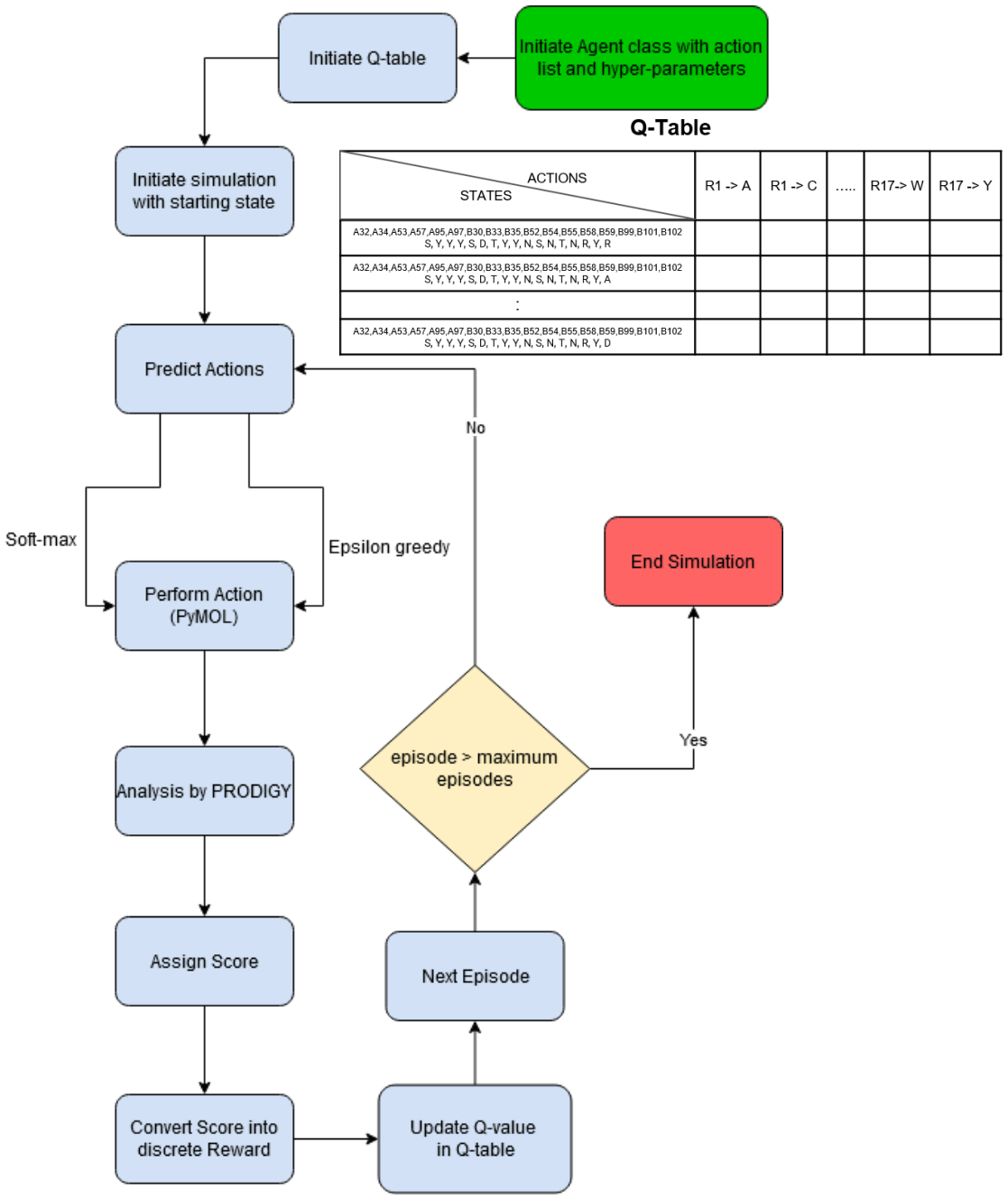


Figure 8. Flowchart of Basic Q-learning

2.3.2. State Reduction

The state space of 17 residues mutated to 20 aa is 20^{17} which is very large for Q-learning. To reduce the state space we can restrict the residues we are working with. To perform this, we used a mutation matrix made from MSA of Fab region of IgG made by Prof. Subhashini Srinivasan's lab member Mr. Yash.

Table 2. Mutation Bias Matrix for the 17 positions in Pembro-PD-1 (5B8C)

CDR	Position	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
CDR-L1	Ser-32	1	0	4	0	3	1	0	0	0	1	0	6	0	0	0	24	0	0	0	13
CDR-L1	Tyr-34	1	0	7	0	1	1	2	2	5	0	0	8	0	2	0	27	1	1	1	24
CDR-L2	Tyr-53	0	0	0	0	9	0	9	0	11	0	0	1	0	0	1	13	0	0	1	425
CDR-L2	Tyr-57	1	0	3	2	11	3	1	6	4	1	0	106	0	2	15	199	98	2	0	16
CDR-H3	Arg-99	19	1	61	22	17	83	12	7	19	18	7	6	31	8	26	33	25	18	12	35
CDR-H3	Tyr-101	11	4	33	16	17	32	6	19	5	13	6	6	4	0	31	21	22	8	5	65
CDR-H3	Arg-102	7	4	20	7	11	41	1	2	0	14	5	8	11	1	11	26	23	10	1	70
CDR-H1	Thr-30	2	0	0	1	0	4	0	17	6	0	1	23	2	0	9	88	342	0	0	0
CDR-H1	Tyr-33	81	0	13	5	8	40	6	5	1	6	0	21	5	1	1	17	31	31	85	138
CDR-H1	Tyr-35	1	0	5	7	1	2	236	1	4	0	0	83	0	19	1	99	14	3	0	19
CDR-H2	Asn-52	6	0	64	3	8	0	13	69	13	5	3	194	0	0	1	37	4	15	3	57
CDR-H2	Ser-54	6	0	10	19	10	106	9	60	9	9	10	98	0	3	12	63	9	11	0	49
CDR-H2	Asn-55	5	0	43	3	49	28	3	5	1	12	0	141	1	2	7	149	27	9	2	8
CDR-H2	Thr-58	70	0	0	1	0	4	0	7	3	0	1	5	17	0	0	13	366	7	0	0
CDR-H2	Asn-59	12	0	17	21	3	19	13	12	68	0	7	211	1	6	21	51	19	4	0	10
CDR-L3	Ser-95	12	1	3	3	12	13	10	3	1	15	3	11	1	1	54	88	1	0	10	222
CDR-L3	Asp-97	4	0	28	11	2	13	2	6	5	3	4	83	1	5	10	179	30	4	4	6

The mutation bias matrix was used to reduce the state space. In total, there were different sets of reduced states which were tested in the simulations:

1. *Light Chain*: Using all 6 positions of the light chain but mutating them only into:
    a. 3 amino acids:
        i. S, Y and N
    b. 6 amino acids: S, Y, N, R, T and D
2. *Heavy Chain*: Only Mutating selected positions for the mutations and mutating them into other residues
    a. Mutating 3 residues (CDR-H1: Tyr-33, CDR-H2: Ser-54, CDR-H3: Arg-99) into 17 aa ('A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y').
    b. Mutating 4 residues (CDR-H2: Ser-54, CDR-H3: Arg-99, Tyr-101 and Arg-102) into 16 aa ('A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'R', 'S', 'T', 'V', 'W', 'Y')

2.4. Deep Q-Learning

The basic pipeline for Deep Q-learning is adapted from Mnih V et al., 2013. The pipeline was made using PyTorch (v.2.2.1+cu121):

1. Initialize the Brain class with hyper-parameters like exploration rate, learning rate, network sync rate, replay memory size, memory mini batch size etc. and initialize the replay memory.
2. Initialize the Policy network with random weights:
    a. Using FFNN with following layers:

i. Input layer: input was created by doing one hot coding of 17 positions and 20 aa creating a binary tensor of size 340.

1. *Simulation 9*: 340→340 2. *Simulation 10*: 340 →170

ii. Hidden layers: 1. *Simulation 9*: 3 hidden layers of:340→170, 170→85, and 85→42. 2. *Simulation 10*: 2 hidden layers of 170→85 and 85→85

iii. Output layer: size of action list : 1. *Simulation 9***:** 42→51 there were 51 actions for the simulation we ran (3 positions into 17 aa).

2. *Simulation 10*: 85→64, 4 positions mutated into 16 aa.

3. Clone the Policy network into target network
4. Loop for the Episodes:
    a. Select an action using policy network with Epsilon greedy strategy
    b. Execute action in PyMOL and save state PDB in the local PDB bank
    c. Observe State, Reward and next state
    d. Store experience tuple (state, action, reward, next state) in replay memory
    e. Sample random mini batch from replay memory → Pass replay mini batch to policy network
    f. Calculate loss (MSE) b/w output Q values (from policy network) and target Q value
        i. Requires passing states through target network (clone of policy network, updated after certain episodes)
        ii. Target Q value is calculated using the Q-value update equation but using target network output as replacement for Q-values.
    g. Gradient Descent (Optimizer: Adam) updates weight in policy network to minimize loss
    h. After certain episodes (network sync rate), weights in the target network are updated with the policy network.

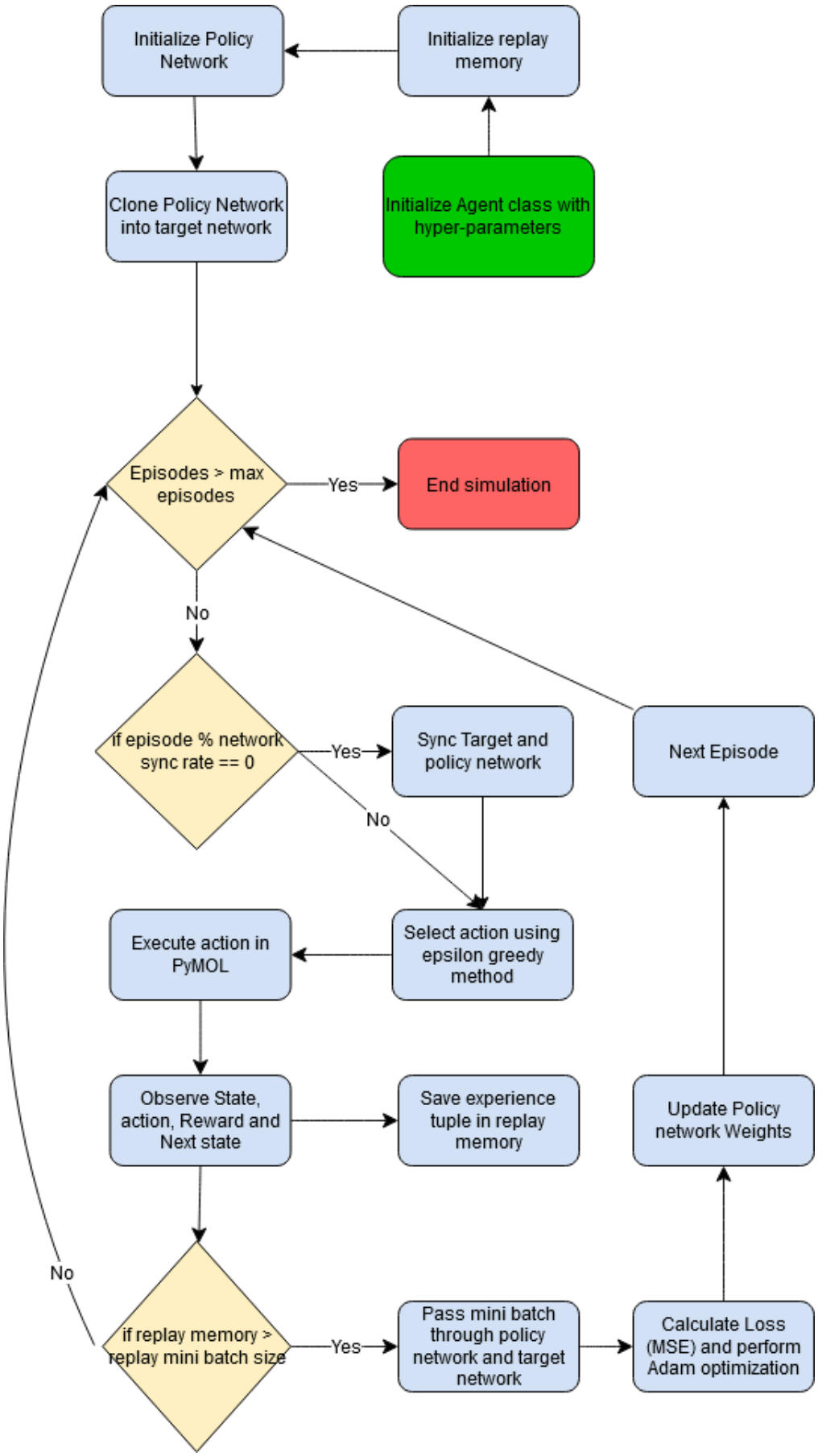


Figure 9. Flowchart of DQL pipeline

2.5. Structure Validation

2.5.1. AlphaFold2 and RMSD

We have used LocalColabFold 1.5.0 which uses AlphaFold 2.3.1 for predicting the states. This allows us to perform validation on if the state is easily folding following the assumptions mentioned in § 1.1.

The pipeline:

1. The last 10000 episodes of the simulation is selected and sorted in ascending order of BA
2. The PDB of states are extracted from local PDB bank
3. The PDB are converted into FASTA format using pdb2fasta (https://zhanggroup.org/pdb2fasta/pdb2fasta) and into LocalColabFold fasta format for multimer.
4. LocalColabFold is used to predict the structures. The command used:

$ nohup colabfold_batch --amber --use-gpu-relax --num-recycle 20 --num-models 5 --num-relax 1 --recycle-early-stop-tolerance 0.01 {file_name}.fasta {file_name} > {file_name}.log &

5. The relaxed structures are separated and PRODIGY is run to get BA
6. RMSD is calculated of the predicted structure against the Pembro structure used as initial state in simulations using PyMOL. Commands used:

load {Pembro control}.pdb

load {AF predicted structure}.pdb

align {Pembro control}, {AF predicted structure}

rms_cur {Pembro control}, {AF predicted structure}, matchmaker=1

2.5.2. Cα – Cα distance plot

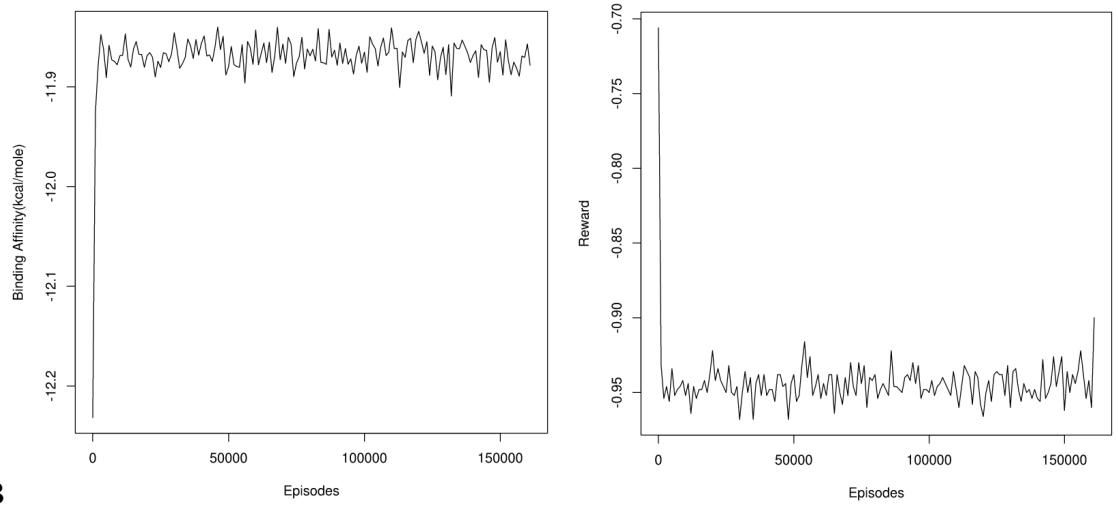
The Cα to Cα distance of the PDB were calculated using PDB class from BioPython (v. 1.80) for chain A vs chain A, chain B vs chain B, chain A vs chain B, chain A vs chain C, chain B vs chain C and heatmap was created for the distance matrices.

3. RESULTS AND DISCUSSIONS

3.1. Mutating all 17 residues

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Random Seed :: 17
Chain :: both
Chain::Residues:: ['1,A,32', '2,A,34', '3,A,53', '4,A,57', '5,A,95', '6,A,97',
                   '7,B,30', '8,B,33', '9,B,35', '10,B,52', '11,B,54', '12,B,55', '13,B,58',
                   '14,B,59', '15,B,99', '16,B,101', '17,B,102']
Num. Amino acids mut. :: 20
Amino acids mut. list :: ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K',
                          'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
=========================================================
Max Episodes :: 500000
Learning Rate :: 0.01
Discounted Return :: 0.9
==========Epsilon Greedy parameters==============
Exploration Rate(Epsilon decay) :: 1.0
Exploration Decay(Epsilon decay) :: 0.01
Exploration Decay/Episode(Softmax) :: 1000
Minimum Exploration(Epsilon decay) :: 0.05
```

A



B

C

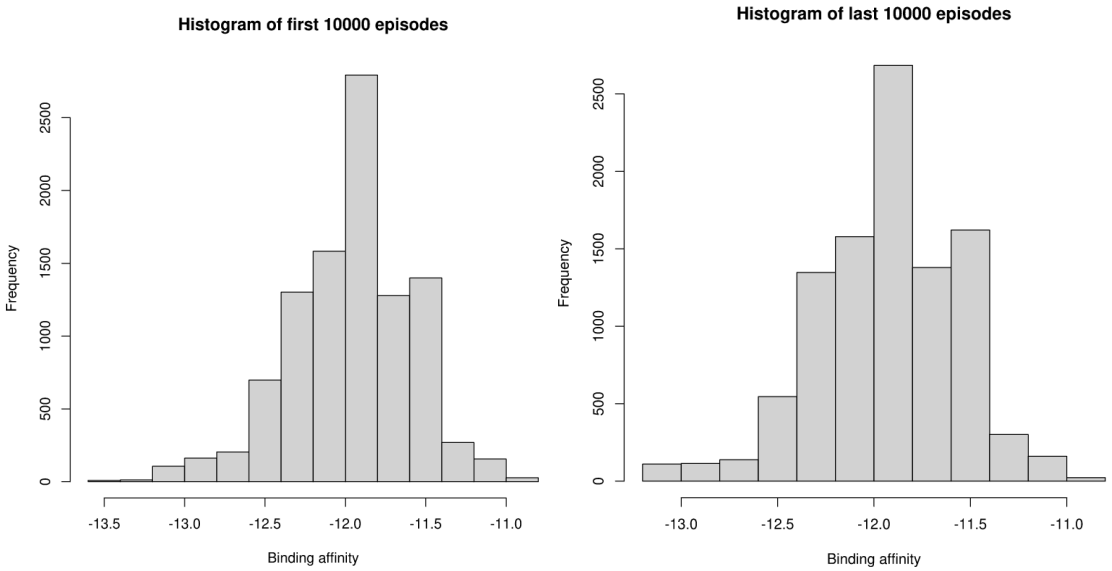


Figure 10. Simulation 1. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

We observed that due to the state space being so large, it is not possible to explore and thus not possible to fill the Q-table to exploit and learn. Thus the state space was reduced in the consequent simulations.

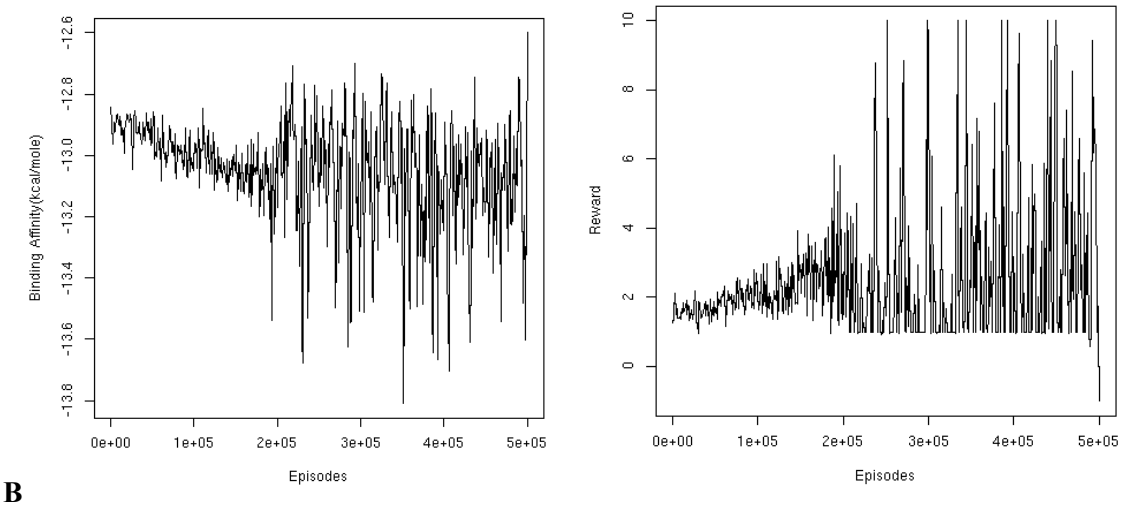
3.2. Mutating residues of Light Chain

3.2.1. Mutating residues of light chain into 3 residues

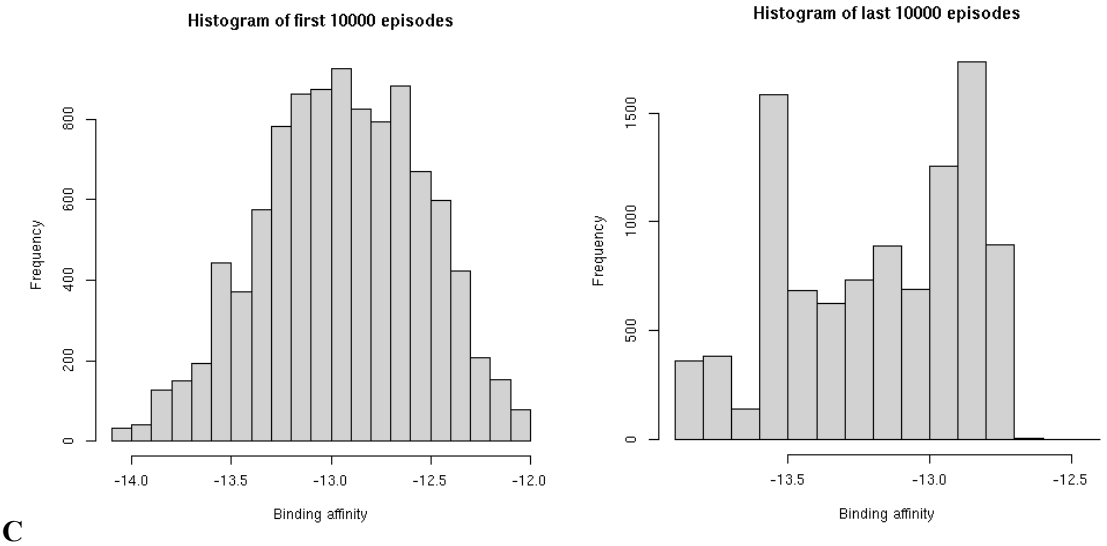
3.3.1.1. Simulation 2

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: light
Residue list :: ['1,A,32', '2,A,34', '3,A,53', '4,A,57', '5,A,95', '6,A,97']
Num. Amino acids mut. :: 3
Amino acids mut. list:: ['S', 'Y', 'N']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay interval :: 1000 episodes
Minimum Exploration :: 0.01
Max Episodes :: 500000
Random Seed :: 37
```



B



C

Figure 11. Simulation 2. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

For this simulation relatively larger aa (S, Y and N) were substituted as compared to before, these were conserved aa observed in the mutation bias matrix. We observed that till 2e+5 episodes the BA decreased and Score increased, then it started to fluctuate highly as more states with +10 rewards are observed.

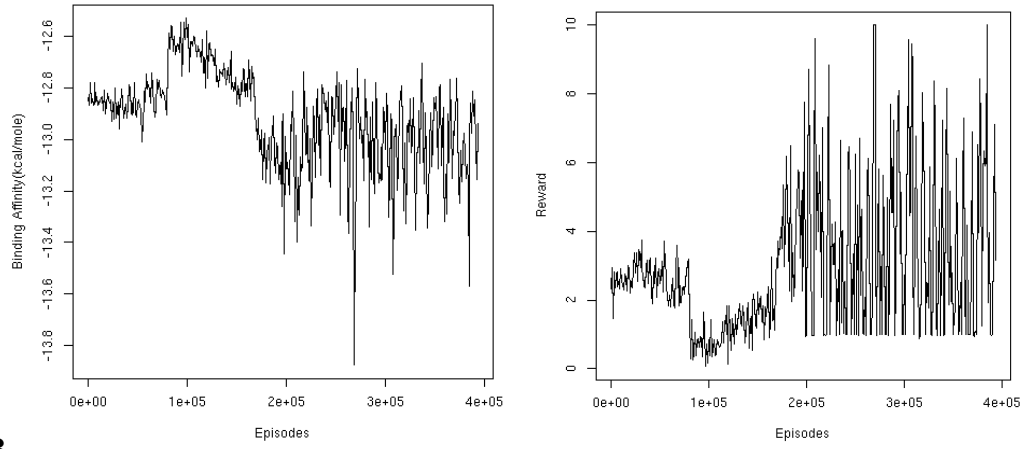
3.2.2. Mutating residues of light chain into 6 residues

3.2.2.1. Simulation 3

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: light
Residue list :: ['1,A,32', '2,A,34', '3,A,53', '4,A,57', '5,A,95', '6,A,97']
Num. Amino acids mut. :: 6
Amino acids mut. list:: ['S','Y','N','R','T','D']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay interval :: 1000 episode
Minimum Exploration :: 0.01
Max Episodes :: 500000
Random Seed :: 643
```

B



C

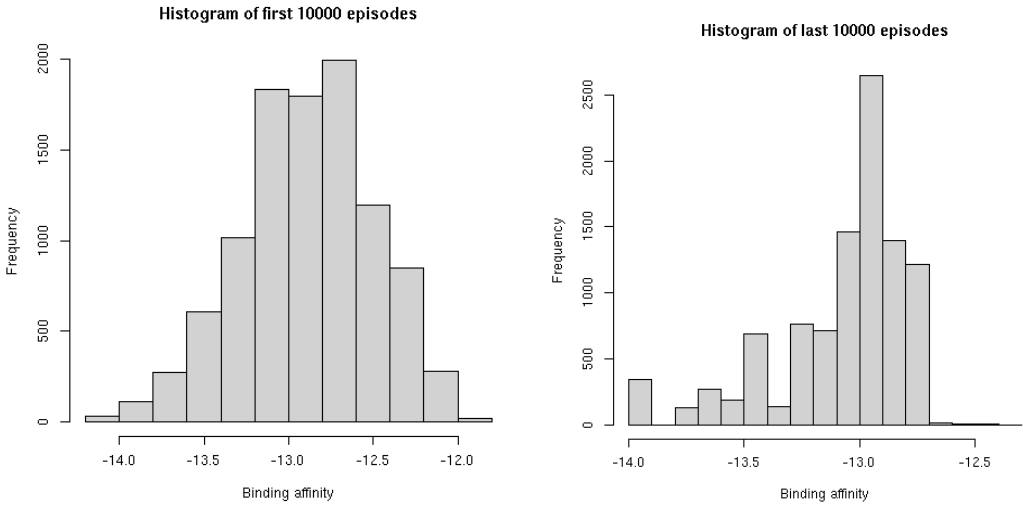


Figure 12. Simulation 3. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

Here we mutate light chain residues into 6 aa, these were also conserved aa observed in the light chain. We observed states with better BA being preferred at the end of simulation comparing the histogram of first and last 10,000 episodes.

We selected conserved amino acids for substitution in the light chain, this does give us better binding states, but to create a more biological model for SHM, it is important to explore positions which can tolerate more types of variations. So we did simulations based on less conserved positions on the heavy chain next.

3.3. Mutations on Heavy Chain

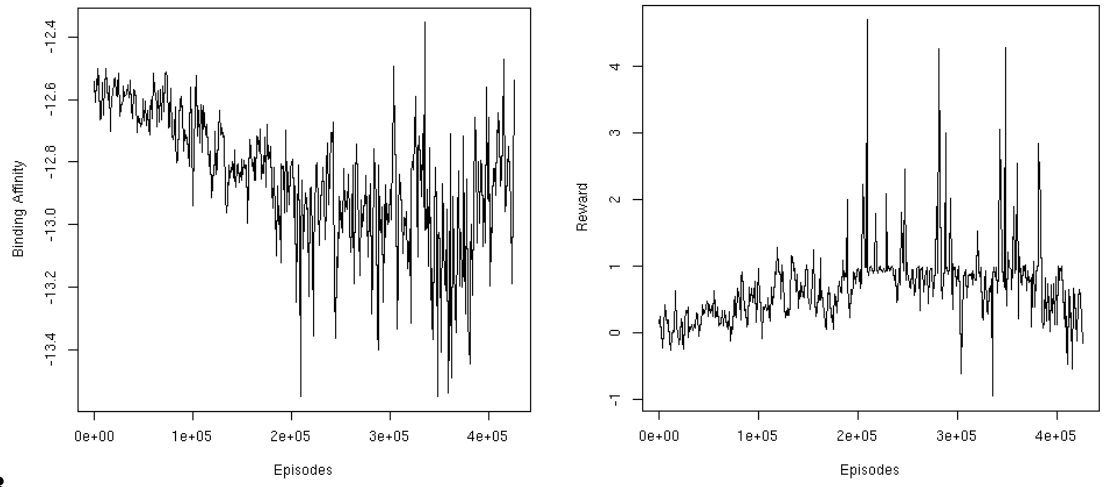
3.3.1. Mutating 3 residues into 17 residues

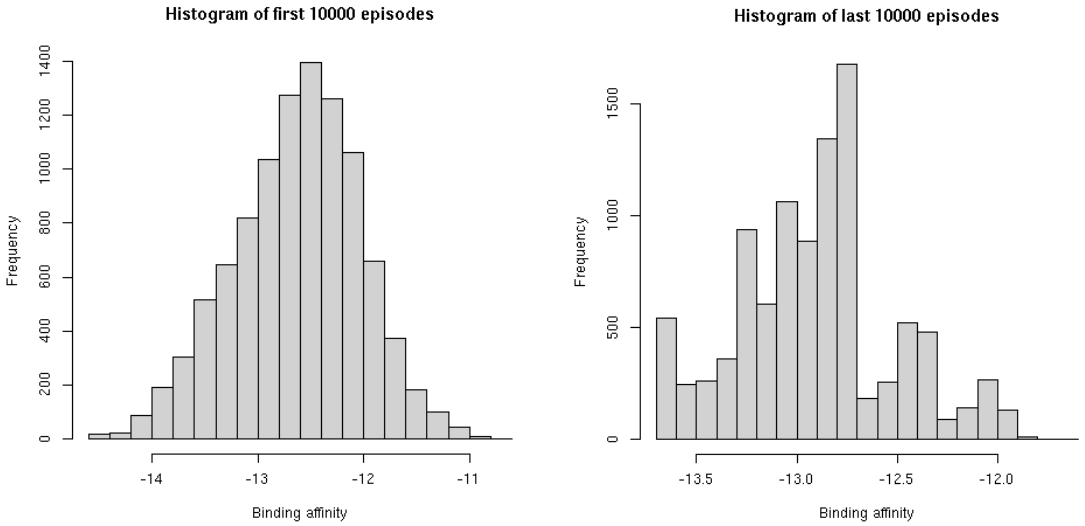
3.3.1.1. Simulation 4

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: heavy_bias
Residue list :: ['8,B,33', '11,B,54', '15,B,99']
Num. Amino acids mut. :: 17
Amino acids mut. list:: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay/Episode :: 1000
Minimum Exploration :: 0.01
Max Episodes :: 800000
Random Seed :: 71
```

B





C

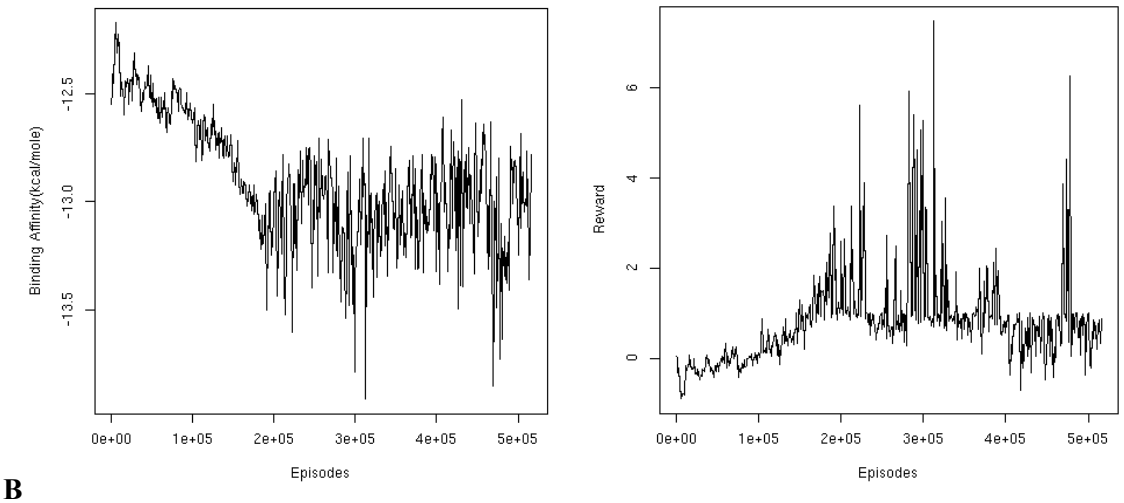
Figure 13. Simulation 4. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

We can observe that the overall BA decreased and score increased by episodes. But here the mean of the first 10,000 episodes is around -12.6 kcal/mole which is the same as our initial state. In previous simulations on light chains, the mean was either higher or less than this, this simulation thus seems more natural for our stated state. Also, the variation in the BA and Reward has reduced significantly as compared to simulations on light chains. The overall frequency of better BA states increased significantly compared to the first 10 000 episodes. To make sure this is not a fluke, let us use a different random seed for the same parameters.

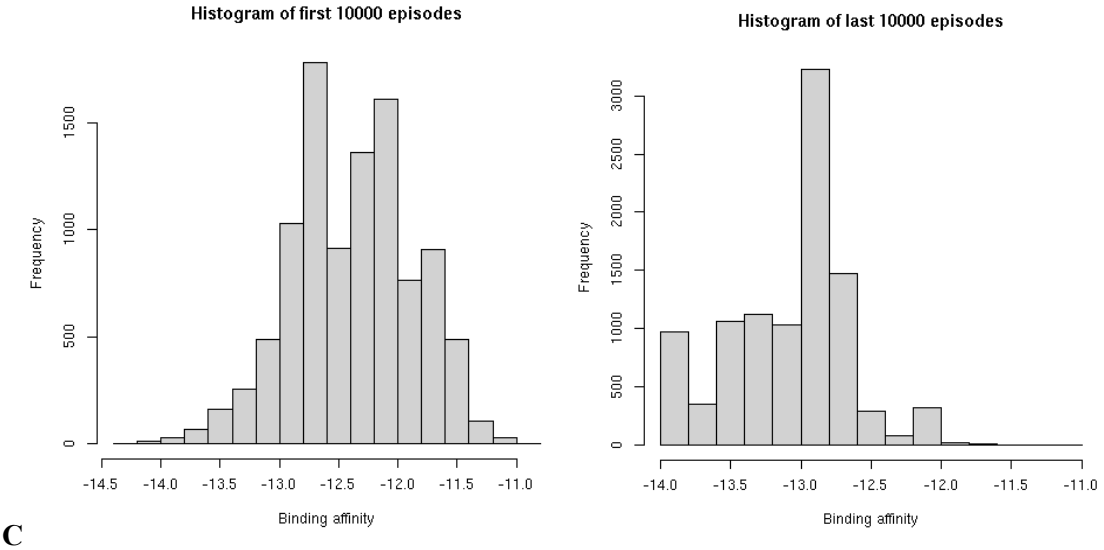
3.3.1.2. Simulation 5

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: heavy_bias
Chain::Residues:: ['8,B,33', '11,B,54', '15,B,99']
Num. Amino acids mut. :: 17
Amino acids mut. list :: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay interval :: 1000 episodes
Minimum Exploration :: 0.01
Max Episodes :: 800000
Random Seed :: 17
```



B



C

Figure 14. Simulation 5. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

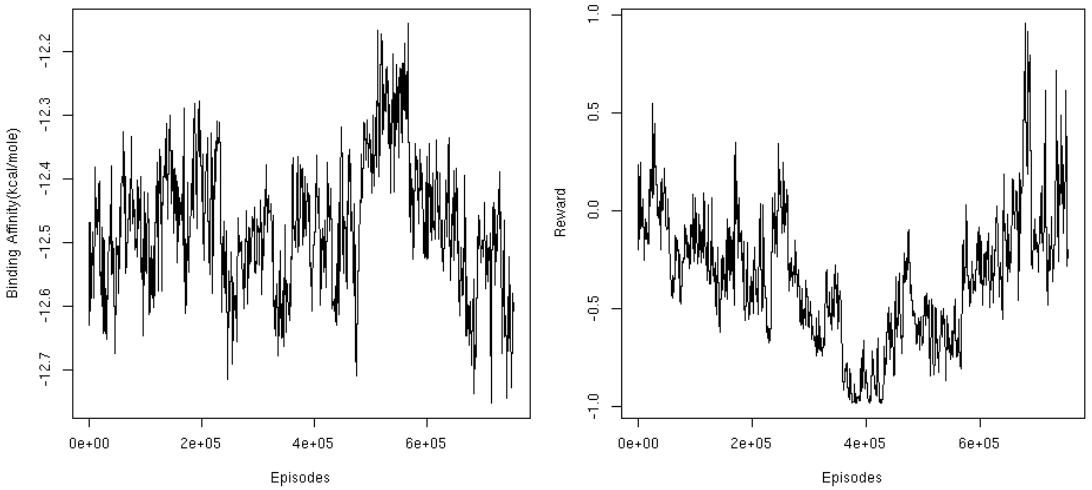
We can observe that even after changing the random seed the simulation showed similar results to the earlier simulation, thus the learning was not dependent on the random seed. Let us try a different exploration method i.e. soft-max for these positions.

3.3.1.3. Simulation 6

A

```
Brain Type :: Q-Table
Exploration Type :: softmax
Chain :: heavy_bias
Residue list :: ['8,B,33', '11,B,54', '15,B,99']
Num. Amino acids mut. :: 17
Amino acids mut. list:: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Temperature :: 100
Temperature Decay :: 0.5
Temperature Decay interval :: 1000 episodes
Minimum Temperature  :: 1.0
Learning Rate :: 0.01
Learning Decay :: 0.0
Minimum Learning Rate :: 0.1
Discounted Return :: 0.9
Max Episodes :: 800000
Random Seed :: 717
```

B



C

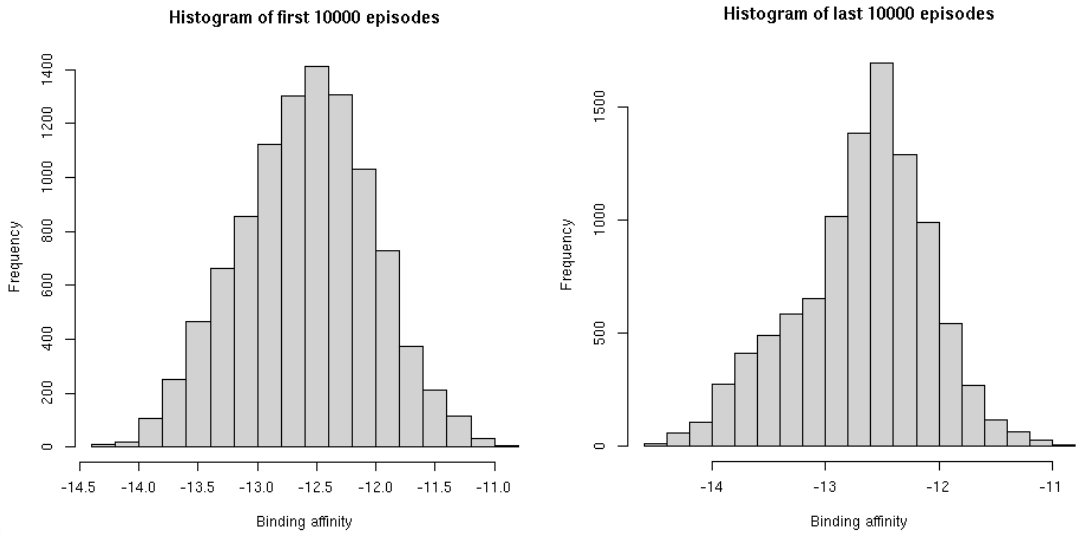


Figure 15. Simulation 6. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

We can observe that the BA and score remained very random throughout the simulation. The histogram of the first and last 10,000 states are also very similar. This can

be because the parameters for soft-max are not optimized, when comparing with epsilon greedy. Or it could be because of improper implementation of the softmax.

For the following simulations, epsilon greedy was used. The soft-max implementation should be cross-checked and tested for a small RL environment such as available from OpenAI gym.

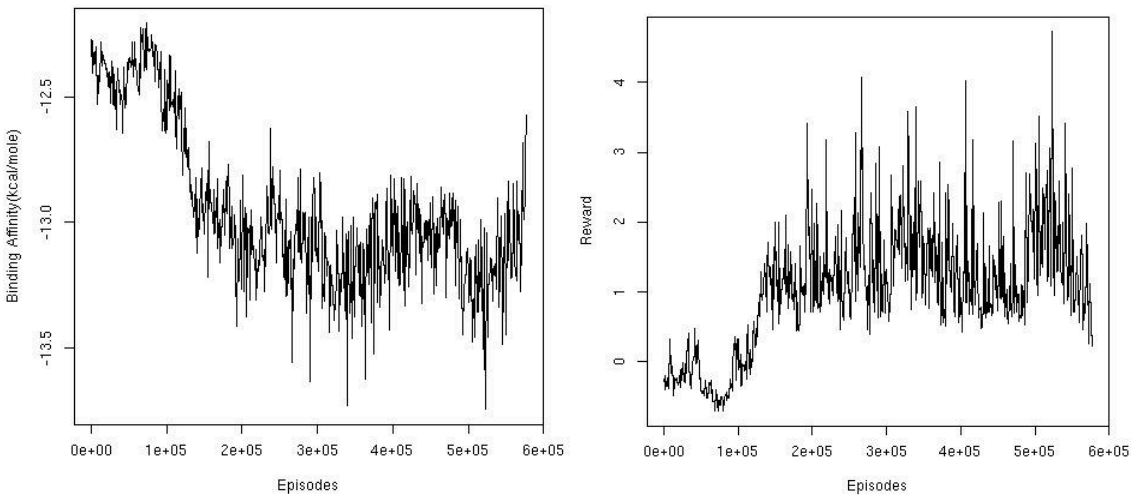
3.3.2. Mutating 4 residues into 16 residues

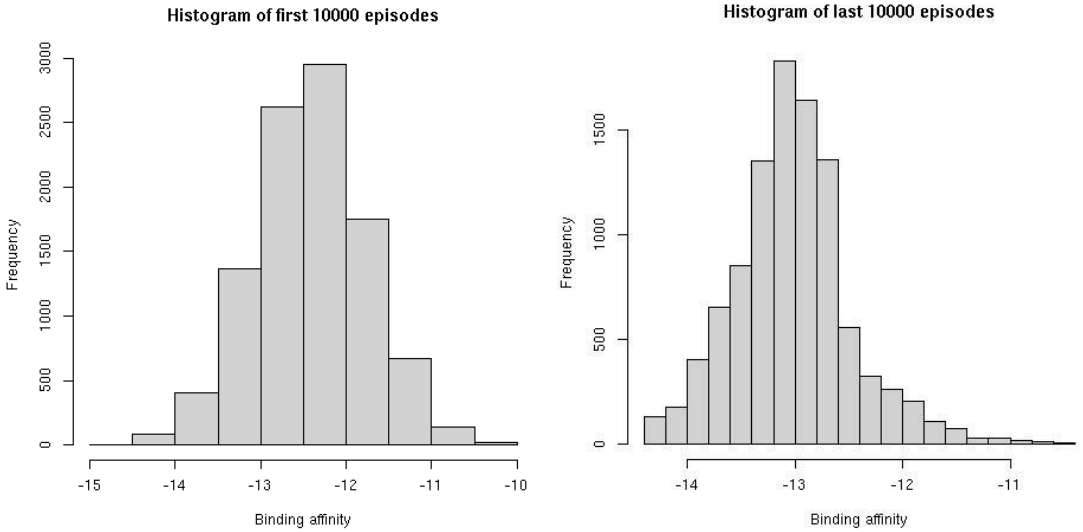
3.3.2.1. Simulation 7

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: heavy_bias2
Chain::Residues:: ['11,B,54', '15,B,99', '16,B,101', '17,B,102']
Num. Amino acids mut. :: 16
Amino acids mut. list :: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay interval :: 1000 episodes
Minimum Exploration :: 0.05
Random Seed :: 71
Max Episodes :: 800000
```

B





C

Figure 16. Simulation 7. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

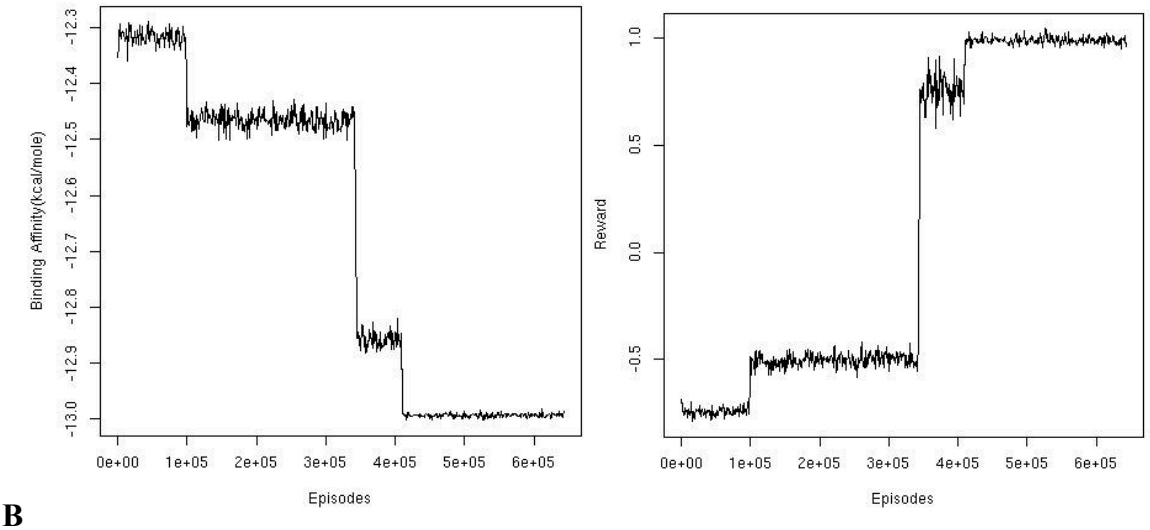
We observed that the BA decreased and Reward increased drastically around the 1.5e+5 episodes. The variation in the BA and Reward also seems to be reduced when compared to simulations before. This simulation also provided higher binding affinity states.

We tried a different algorithm for the next simulation to try to replicate the biological system in which the states where the B cells with high BA are killed by the system. Thus, if a state reaches -10 reward, the simulation resets to the initial Pembro state.

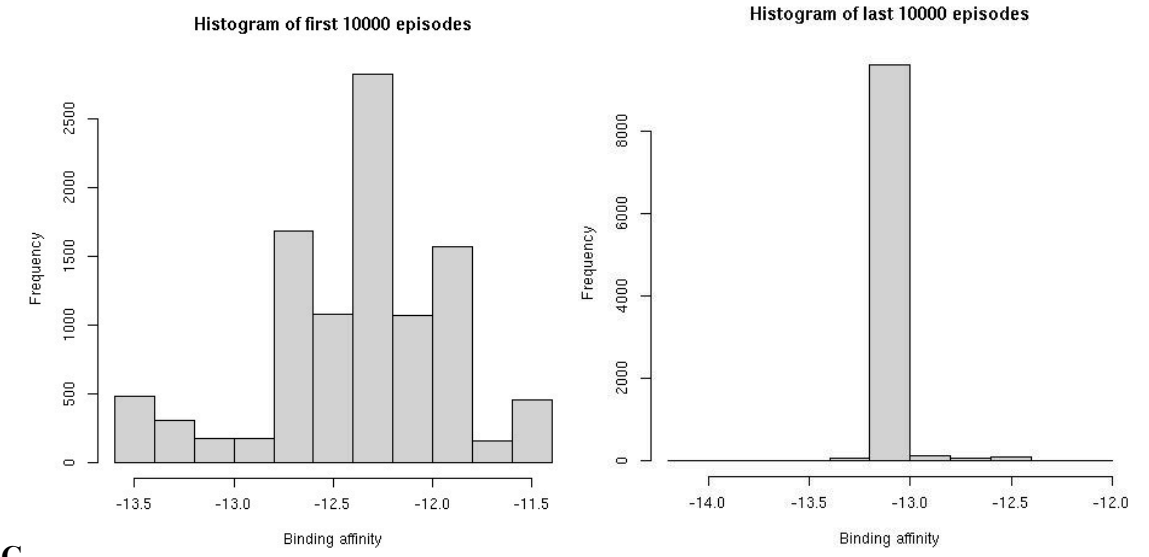
3.3.2.2. Simulation 8

A

```
Brain Type :: Q-Table
Exploration Type :: epsilon_decay
Chain :: heavy_bias2
Chain::Residues:: ['11,B,54', '15,B,99', '16,B,101', '17,B,102']
Num. Amino acids mut. :: 16
Amino acids mut. list :: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
Exploration Rate :: 1.0
Exploration Decay :: 0.005
Exploration Decay interval :: 1000 episodes
Minimum Exploration :: 0.05
Max Episodes :: 800000
Random Seed :: 17
```



B



C

Figure 17. Simulation 8. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

What an interesting change in the flow of the simulation. As compared to before, here the reduction in the BA and increase in Reward is much more discrete and in steps. And the last 10 000 are dominated by the -13 kcal/mol states, although we haven't performed the structure validation on these states yet. It seems that the new condition of survival of states restricts the exploration also, as now the agent cannot explore states which might need to jump though few -10 reward states.

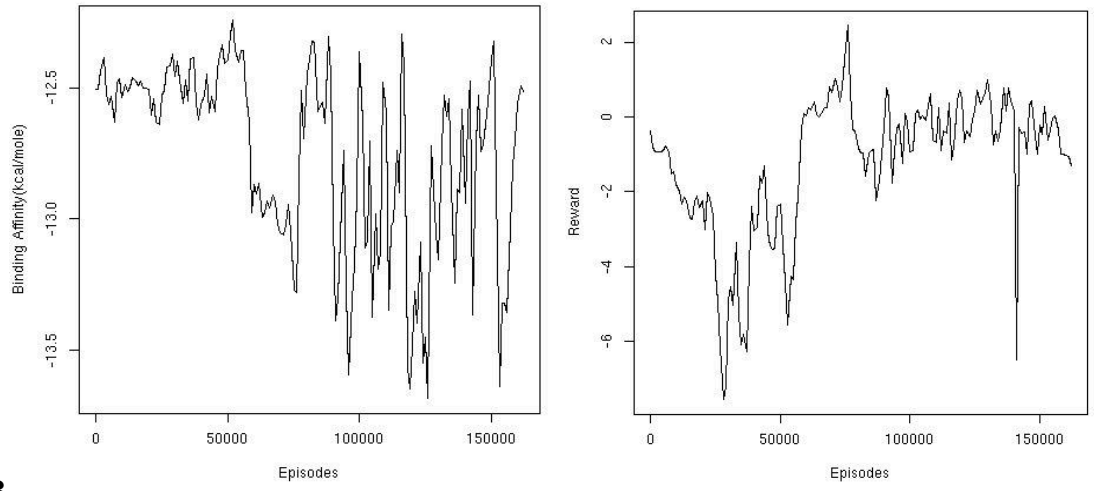
3.4. Deep Q Learning

3.4.1 *Simulation 9*

A

```
Brain Type :: Deep-Q
Exploration Type :: epsilon_decay
Random Seed :: 7
Max Episodes :: 500000
Chain :: heavy_bias
Chain::Residues:: ['8,B,33', '11,B,54', '15,B,99']
Num. Amino acids mut. :: 17
Amino acids mut. list :: ['A', 'D', 'E', 'F', 'G', 'H', 'I', 'K',
                          'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
Learning Rate :: 0.01
Discounted Return :: 0.9
===========Epsilon Greedy parameters==============
=====This  applies to both Q-learning and Deep Q=======
Exploration Rate(Epsilon decay) :: 1.0
Exploration Decay(Epsilon decay) :: 0.01
Exploration Decay/Episode(Softmax) :: 1000
Minimum Exploration(Epsilon decay) :: 0.01
=====Deep Q Learning parameters=======
Network Sync Rate :: 5000
Replay Memory Size :: 5000
Replay Memory::Mini Batch Size :: 200
```

B



C

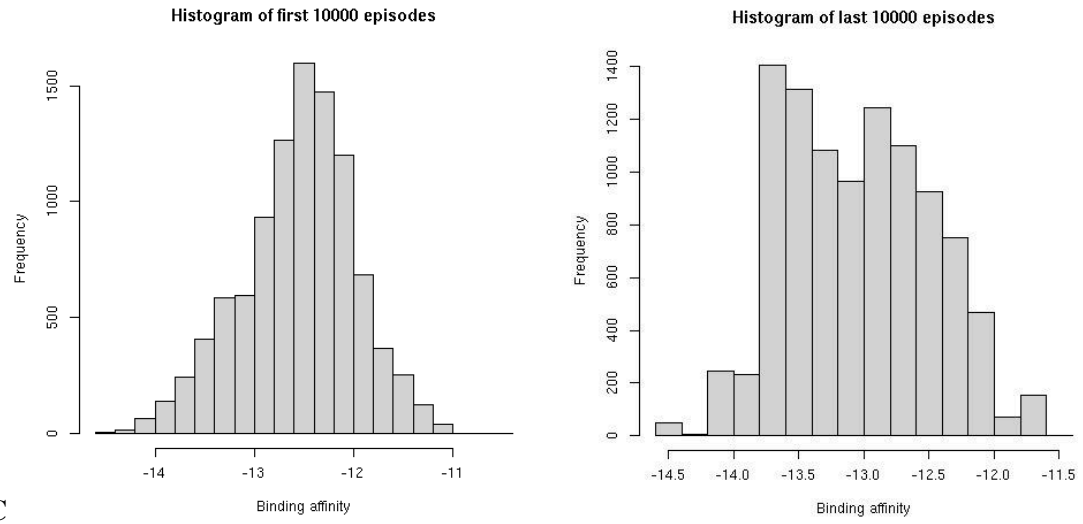


Figure 18. Simulation 9. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

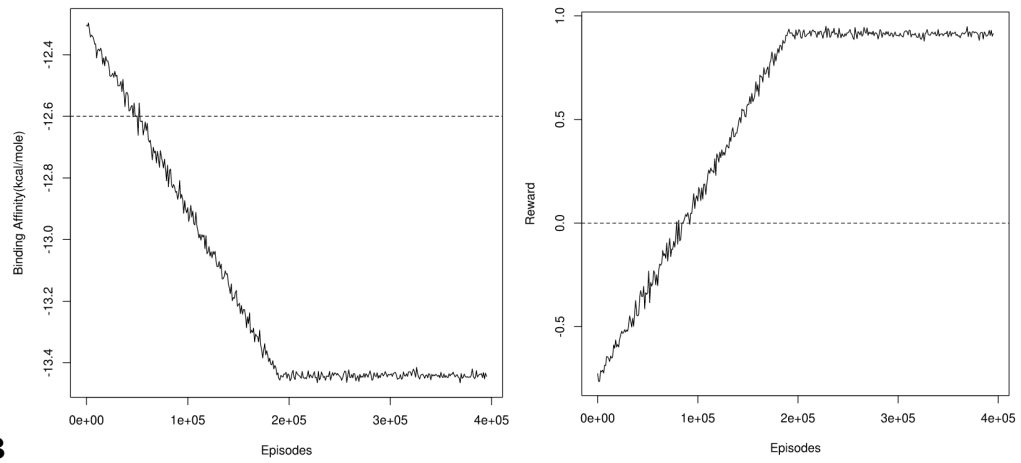
The DQL simulation is much slower and more compute intensive as compared to Q-Learning, given the time constraint, only one simulation was possible. We can observe that the simulation has a lot of variation, but for BA, this variation is occurring in our favorable region. We got some good states from the simulation, but the simulation itself requires optimization of parameters. The input tensor also can be changed from a simple one hot coding of the state to include factors such as type of aa in the positions i.e. hydrophobic, charged, polar, apolar, etc. and the types of interactions each position have with PD-1. Since the DQL system is slow, it requires much more time to design effectively, which will be continued in the future.

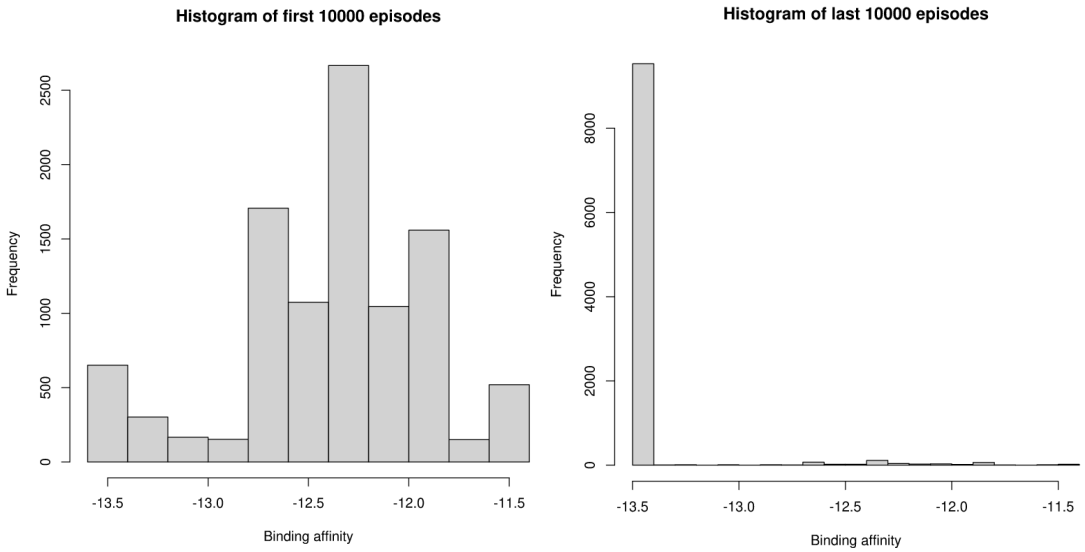
3.4.2 *Simulation 10*

A

```
Brain Type :: Deep-Q
Exploration Type :: epsilon_decay
Random Seed :: 37
Chain :: bias2
Chain::Residues:: ['11,B,54', '15,B,99', '16,B,101', '17,B,102']
Num. Amino acids mut. :: 16
Amino acids mut. list :: ['A', 'D', 'E', 'F', 'G', 'H', 'I',
                          'K', 'L', 'N', 'R', 'S', 'T', 'V', 'W', 'Y']
BA Tool :: prodigy
======================================================================
Max Episodes :: 500000
Learning Rate :: 0.01
Learning Decay :: 0.0
Minimum Learning Rate :: 0.1
Discounted Return :: 0.9
==================Epsilon Greedy parameters============================
Exploration Rate(Epsilon decay) :: 1.0
Exploration Decay(Epsilon decay) :: 0.005
Exploration Decay/Episode(Softmax) :: 1000
Minimum Exploration(Epsilon decay) :: 0.05
==================Deep Q Learning parameters==========================
Network Sync Rate :: 2000
Replay Memory Size :: 5000
Replay Memory::Mini Batch Size :: 200
**********************************************************************
```

B





C

Figure 19. Simulation 10. **A.** Hyperparameters, **B.** Average BA (left) and Reward (Right) per 1000 episodes, **C.** Histogram of first (left) and last (right) 10,000 episodes

We observe that the binding affinity is gradually reducing till it reaches -13.4 kcal/mole near ~2e+05th episode, and remains there for ~2e+05 episodes similarly the reward slowly increases and stabilizes at +1.

We observe that the simulation ends with episodes with around -13.5 kcal/mole binding affinity.

It can be said by these observations that the simulation converges and the agent has learnt, but the policy network was not tested, thus it is inconclusive.

3.5. Validation of Higher Affinity States

3.5.1. Validation by AlphaFold2 and RMSD

3.5.1.1. States of mutations on light chain

Table 3. Simulation 3: Mutating 6 residues of light chain into 6 aa

State	Freq	Reward	Crystal BA	AF BA	RMSD
DRRTSYTYYNSNTNRYR	65	10	-13.4	-23.3	19.48
STRNYYTYYNSNTNRYR	45	10	-13.6	-13.5	2.061
YRNNSRTYYNSNTNRYR	141	10	-13.4	-13.4	2.058
DSRNYYTYYNSNTNRYR	41	10	-13.7	-13.3	2.072
DTRNYYTYYNSNTNRYR	89	10	-13.7	-13.3	1.987
YRNNNRTYYNSNTNRYR	101	10	-13.3	-13.3	2.072

We have only shown the top 5 states with respect to Reward and binding affinity after AlphaFold2. We can observe that AlphaFold2 can incorrectly fold the antibodies for certain mutations. This is explained in detail later.

3.5.1.2. States of mutations on heavy chain

Table 4. Simulation 4: Mutating 3 residues of heavy chain into 17 aa

State	Freq	Reward	Crystal BA	AF BA	RMSD
SYYYSDTWYNYNTNGYR	21	1	-13.0	-25.3	20.214
SYYYSDTRYNLNTNYYR	246	1	-13.1	-13.9	2.205
SYYYSDTRYNLNTNDYR	72	1	-13.3	-13.6	2.205
SYYYSDTDYNLNTNDYR	90	1	-12.8	-13.5	2.161
SYYYSDTLYNLNTNEYR	61	1	-13.5	-13.5	2.121
SYYYSDTRYNFNTNWYR	81	10	-14.0	-13.3	2.176

Table 5. Simulation 5: Mutating 3 residues of heavy chain into 17 aa

State	Freq	Reward	Crystal BA	AF BA	RMSD
SYYYSDTDYNFNTNYYR	19	1	-13.4	-13.4	2.140
SYYYSDTHYNYNTNYYR	215	1	-13.0	-13.4	2.196
SYYYSDTIYNLNTNVYR	132	1	-13.3	-13.3	2.135
SYYYSDTEYNYNTNFYR	93	1	-13.2	-13.2	2.244
SYYYSDTNYNWNTNIYR	41	1	-12.9	-13.2	2.197

Table 6. Simulation 7: Mutating 4 residues of heavy chain into 16 aa

State	Freq	Reward	Crystal BA	AF BA	RMSD
SYYYSDTYYNYNTNEWE	34	1	-13.7	-13.5	2.195
SYYYSDTYYNYNTNNWE	34	1	-12.9	-13.5	2.172
SYYYSDTYYNLNTNAWW	77	1	-12.7	-13.4	2.101
SYYYSDTYYNFNTNAWY	45	10	-14.0	-13.3	2.217
SYYYSDTYYNFNTNIYY	39	10	-14.1	-13.0	2.185

Table 7. Simulation 8: Mutating 4 residues of heavy chain into 16 aa in loop

State	Freq	Reward	Crystal BA	AF BA	RMSD
SYYYSDTYYNFNTNRYR	5	1	-13.4	-13.1	2.179
SYYYSDTYYNLNTNRYR	6	1	-13.5	-13.0	2.153
SYYYSDTYYNWNTNRYR	6	1	-13.4	-13.0	2.286
SYYYSDTYYNYNTNRYR	9	1	-13.6	-12.8	2.193

As compared to the light chain mutations, less states with +10 rewards are observed towards the end of simulation. This shows that even though there are less positions on the light chain we are working with the set of 3 aa and 6 aa affecting the BA of antibodies to a great extent.

3.5.1.3 States from DQL

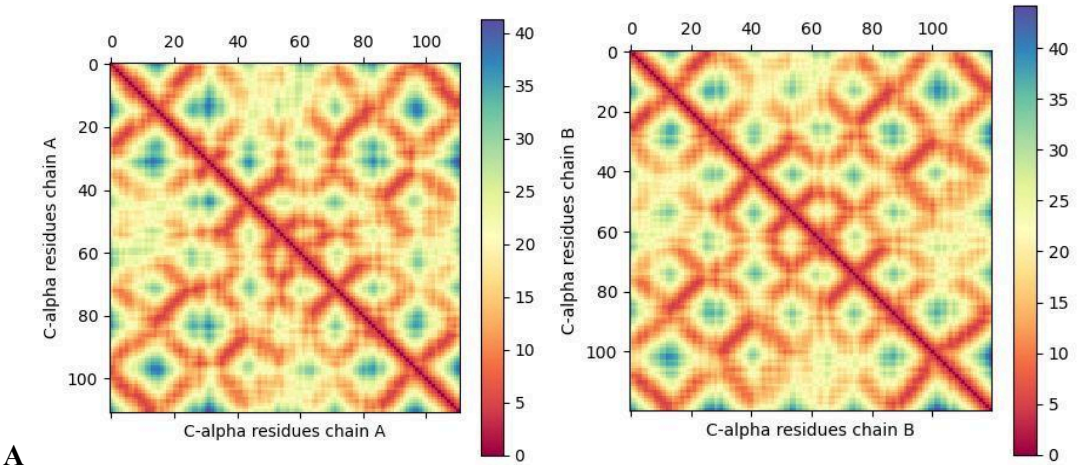
Table 8. Simulation 10: DQL without -10 reward apoptosis loop

State	Freq	Reward	Crystal BA	AF BA	RMSD
SYYYSDTGYNLNTNDYR	165	1	-13.0	-13.9	2.169
SYYYSDTSYNLNTNDYR	117	1	-12.8	-13.6	2.177
SYYYSDTSYNLNTNLYR	15	1	-13.9	-13.2	2.161
SYYYSDTVYNLNTNDYR	94	1	-13.5	-13.1	2.147
SYYYSDTYYNVNTNEYR	52	1	-13.2	-13.1	2.141

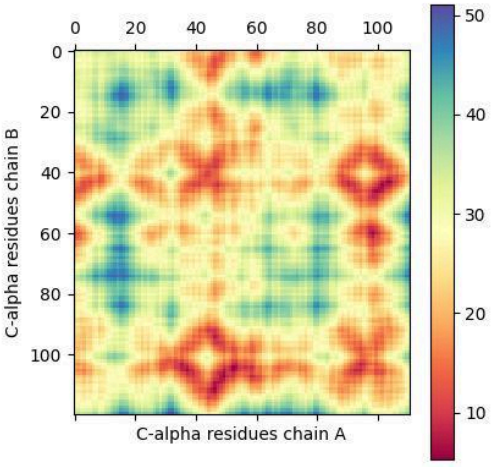
Although we got states which had binding affinity better than Pembro, these states have only been validated using static structures. Without rigorous molecular dynamics study followed by removal of states which may recognize self antigens, these are not biologically relevant, especially in clinical cases like that of Pembro for cancer treatment.

3.5.2. Validation by Cα distance plot

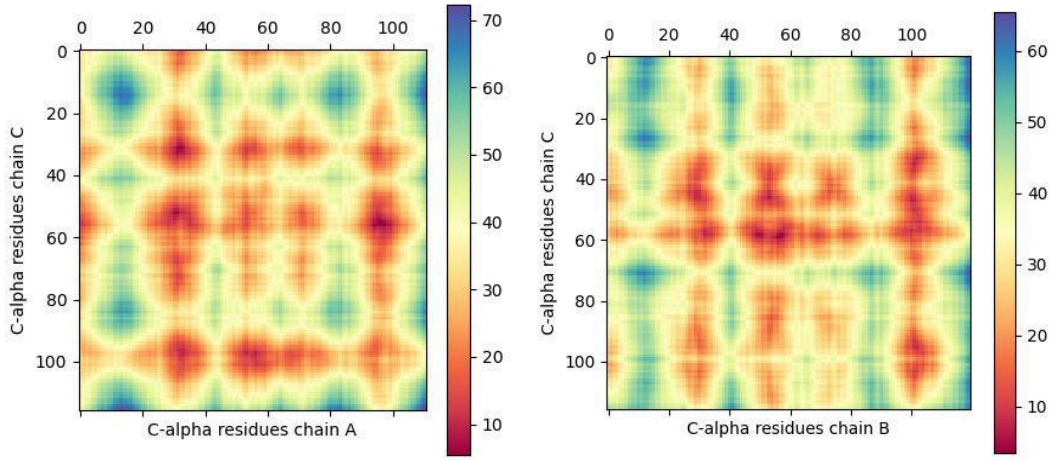
3.5.2.1. Pembro (control)



A



B

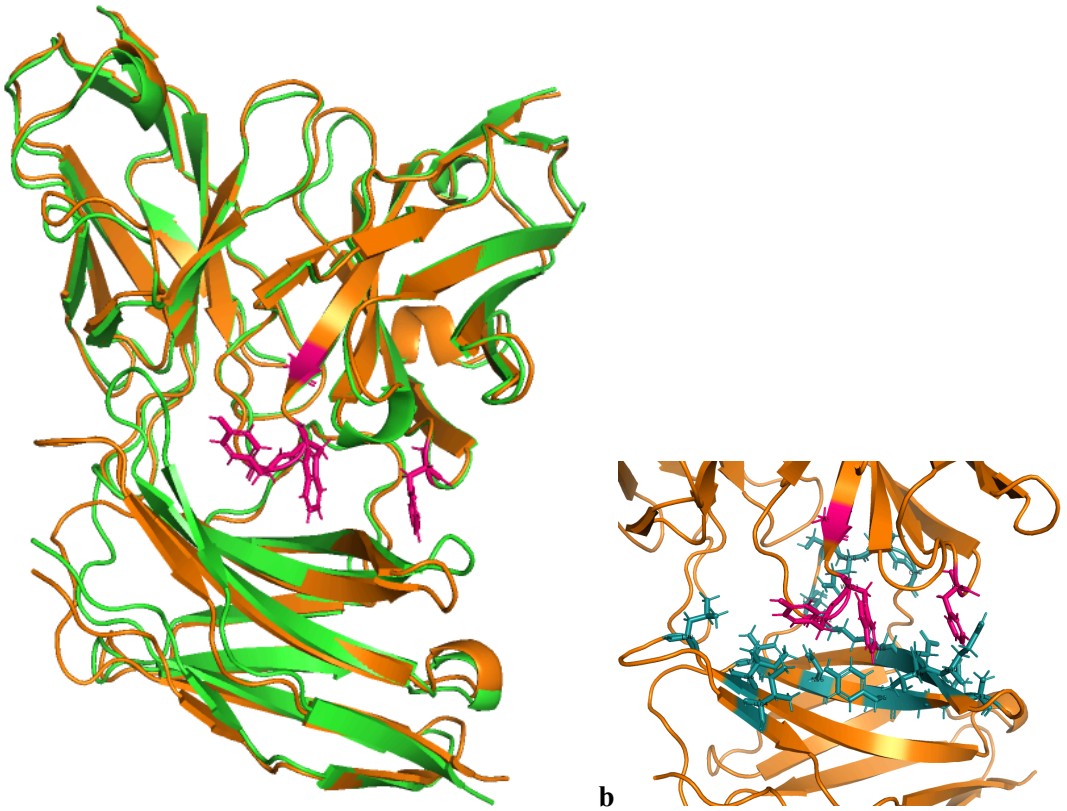


C

Figure 20. Pembro: A. Cα distance plot of chain A vs chain A and chain B vs chain B, B. Cα distance plot of chain A vs chain B, C. Cα distance plot of chain A vs chain C and chain B vs chain C

These plots are used to validate if our assumption about SHM model in § 1.1 holds for the states produced by the model after AlphaFold2 prediction.

3.5.2.2. Properly folded state after AlphaFold2 prediction



A a b

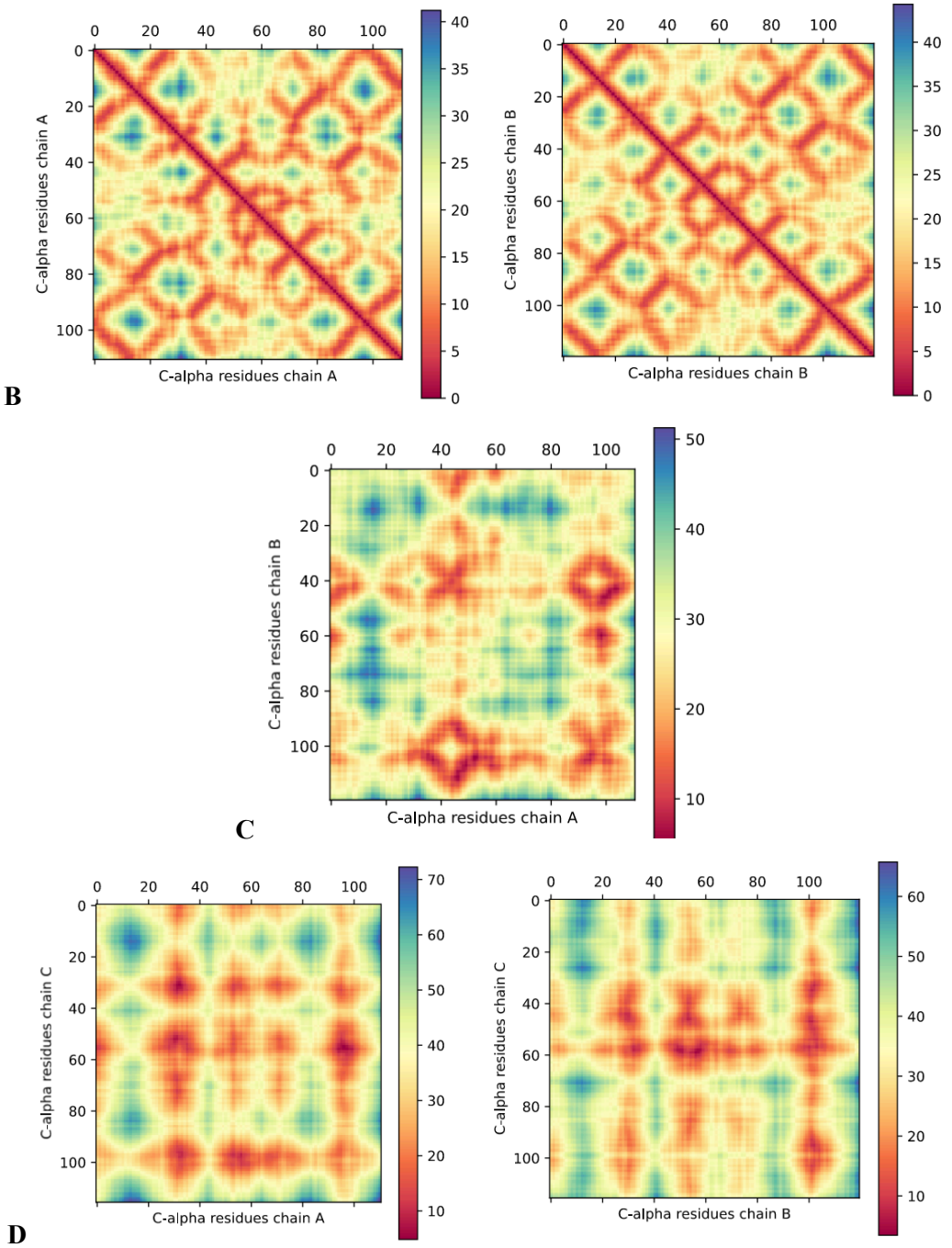
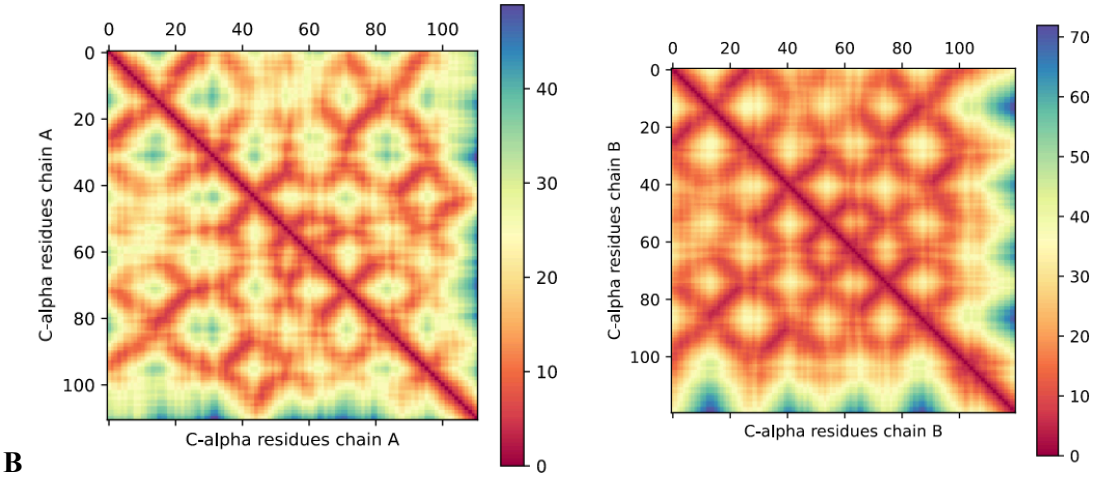


Figure 21. **A. a.** AF predicted structure of state SYYYSDTYYNFNTNAWY aligned with Pembro (control) state, **b.** Interactions of the mutated atoms with PD-1, **B.** Cα distance plot of intra chain A and chain B, **C.** Cα distance plot of inter chain A vs chain B, D. Cα distance plot of inter chain A vs chain C and chain B vs chain C

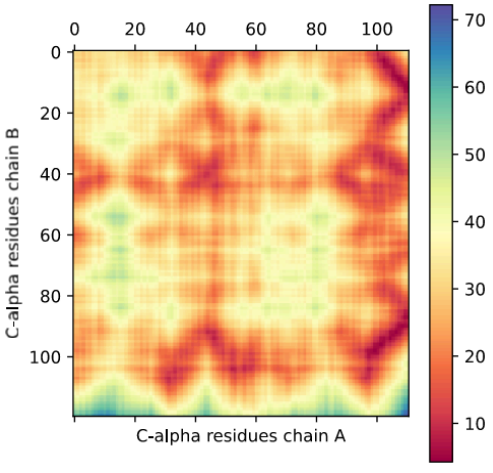
Comparing the plots from control to the predicted structure, we can observe that overall the plots are almost the same, signifying that the assumptions made by us holds that the light and heavy chain folds properly after the SHM and that the relative configuration of the Pembro-PD-1 complex remains the same.

3.5.2.3. Improperly folded state after AlphaFold2 prediction

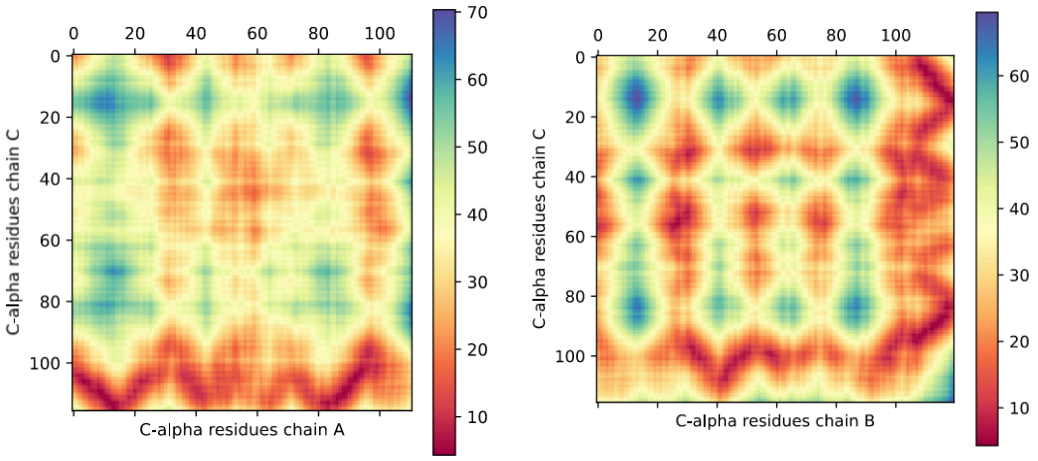
A



B



C



D

Figure 22. A. AF predicted structure of state DRRTSYTYYYNSNTNRYR aligned with Pembro (control) state, **B.** Cα distance plot of intra chain A and chain B, **C.** Cα distance plot of inter chain A vs chain B, **D.** Cα distance plot of inter chain A vs chain C and chain B vs chain C

Our assumption doesn't hold for all predicted structures, although for the majority of structures it does. We can observe that the light and heavy chains are improperly folded and the relative configuration of the mutated Pembro-PD-1 complex has changed. Perhaps this can be avoided if we perform AF predictions with multiple random seeds for our states.

The validation of the states require more than just AlphaFold2 and Cα distance plots for it to be biologically relevant, it requires more dynamic study states with solvation and without solvation. It is also important to note that self antigen recognizing Abs should be removed through bioinformatics methods for this study to be clinically relevant.

4. CONCLUSION

We can observe that using Q-learning we were able to simulate the SHM, for a very restricted state space. The DQL model showed convergence for a restricted state space, but it is not biologically relevant as it doesn't take actions based on understanding of contacts. The agent requires more parameters in the input layer which could describe the biological properties of amino acids, perhaps then the agent will take actions by understanding their biologically relevance.

The main achievement was the creation of a proper reward function which is biologically relevant and can be used to train the model. We observed that using the scoring vector and the reward calculation we were able to train the model, and the training simulations showed convergence for both Q-Learning and Deep Q-Learning for a very restricted state space. It must be pointed out that the agents were trained, but have not been tested, which needs to be done.

We were able to find states with better binding affinity than Pembro according to assessment with PRODIGY, but these states need to be further validated using methods such as molecular dynamics study to confidently provide with an antibody which is biologically relevant and can be engineered in a wet lab.

The current model has a limitation that it requires a PDB with crystal structure of an antibody-antigen and the set of important interactions it makes with the antigen to implement it properly. This can be overcome if we utilize AlphaFold3, which is much better than AlphaFold2 for protein-ligand binding.

It is also necessary to create a screening process which can recognize the antibodies which can bind to self antigens and remove them. Without this, the model would not make any biological sense, especially in clinical scenarios.

5. REFERENCES

1. Dunbar, J., Krawczyk, K., Leem, J., Marks, C., Nowak, J., Regep, C., Georges, G., Kelm, S., Popovic, B., & Deane, C. M. (2016). SAbPred: a structure-based antibody prediction server. *Nucleic Acids Research*, *44*(W1), W474–W478. https://doi.org/10.1093/NAR/GKW361
2. Evans, R., O'Neill, M., Pritzel, A., Antropova, N., Senior, A., Green, T., Žídek, A., Bates, R., Blackwell, S., Yim, J., Ronneberger, O., Bodenstein, S., Zielinski, M., Bridgland, A., Potapenko, A., Cowie, A., Tunyasuvunakool, K., Jain, R., Clancy, E., … Hassabis, D. (2022). Protein complex prediction with AlphaFold-Multimer. *BioRxiv*, 2021.10.04.463034. https://doi.org/10.1101/2021.10.04.463034
3. Faris, J. G., Orbidan, D., Wells, C., Petersen, B. K., & Sprenger, K. G. (2022). Moving the needle: Employing deep reinforcement learning to push the boundaries of coarse-grained vaccine models. *Frontiers in Immunology*, *13*. https://doi.org/10.3389/FIMMU.2022.1029167
4. Horita, S., Nomura, Y., Sato, Y., Shimamura, T., Iwata, S., & Nomura, N. (2016). High-resolution crystal structure of the therapeutic antibody pembrolizumab bound to the human PD-1. *Scientific Reports 2016 6:1*, *6*(1), 1–8. https://doi.org/10.1038/srep35297
5. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature 2021 596:7873*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2
6. Kastritis, P. L., Moal, I. H., Hwang, H., Weng, Z., Bates, P. A., Bonvin, A. M. J. J., & Janin, J. (2011a). A structure-based benchmark for protein–protein binding affinity. *Protein Science*, *20*(3), 482–491. https://doi.org/10.1002/PRO.580
7. Kastritis, P. L., Moal, I. H., Hwang, H., Weng, Z., Bates, P. A., Bonvin, A. M. J. J., & Janin, J. (2011b). A structure-based benchmark for protein-protein binding affinity. *Protein Science*, *20*(3), 482–491. https://doi.org/10.1002/PRO.580
8. Kastritis, P. L., Rodrigues, J. P. G. L. M., Folkers, G. E., Boelens, R., & Bonvin, A. M. J. J. (2014a). Proteins Feel More Than They See: Fine-Tuning of Binding Affinity by Properties of the Non-Interacting Surface. *Journal of Molecular Biology*, *426*(14), 2632–2652. https://doi.org/10.1016/J.JMB.2014.04.017
9. Kastritis, P. L., Rodrigues, J. P. G. L. M., Folkers, G. E., Boelens, R., & Bonvin, A. M. J. J. (2014b). Proteins feel more than they see: Fine-tuning of binding affinity by properties of the non-interacting surface. *Journal of Molecular Biology*, *426*(14), 2632–2652. https://doi.org/10.1016/J.JMB.2014.04.017
10. Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization*.
11. Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., & Steinegger, M. (2022). ColabFold: making protein folding accessible to all. *Nature Methods 2022 19:6*, *19*(6), 679–682. https://doi.org/10.1038/s41592-022-01488-1
12. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.
13. Myung, Y., Pires, D. E. v, & Ascher, D. B. (2022). CSM-AB: graph-based antibody-antigen binding affinity prediction and docking scoring function. *Bioinformatics (Oxford, England)*, *38*(4), 1141–1143. https://doi.org/10.1093/BIOINFORMATICS/BTAB762

14. Odegard, V. H., & Schatz, D. G. (2006). Targeting of somatic hypermutation. *Nature Reviews Immunology 2006 6:8*, *6*(8), 573–583. https://doi.org/10.1038/nri1896
15. Papavasiliou, F. N., & Schatz, D. G. (2002). Somatic hypermutation of immunoglobulin genes: Merging mechanisms for genetic diversity. *Cell*, *109*(2 SUPPL. 1), S35–S44. https://doi.org/10.1016/S0092-8674(02)00706-7
16. Patnaik, A., Kang, S. P., Rasco, D., Papadopoulos, K. P., Elassaiss-Schaap, J., Beeram, M., Drengler, R., Chen, C., Smith, L., Espino, G., Gergich, K., Delgado, L., Daud, A., Lindia, J. A., Nicole Li, X., Pierce, R. H., Yearley, J. H., Wu, D., Laterza, O., … Tolcher, A. W. (2015). Phase i study of pembrolizumab (MK-3475; Anti-PD-1 monoclonal antibody) in patients with advanced solid tumors. *Clinical Cancer Research*, *21*(19), 4286–4293. https://doi.org/10.1158/1078-0432.CCR-14-2607/115837/AM/PHASE-I-STUDY-OF-PEMBROLIZUMAB-MK-3475-ANTI-PD-1
17. Peterson, C., Denlinger, N., & Yang, Y. (2022). Recent Advances and Challenges in Cancer Immunotherapy. *Cancers*, *14*(16). https://doi.org/10.3390/CANCERS14163972
18. Raoufi, E., Hemmati, M., Eftekhari, S., Khaksaran, K., Mahmodi, Z., Farajollahi, M. M., & Mohsenzadegan, M. (2020). Epitope Prediction by Novel Immunoinformatics Approach: A State-of-the-art Review. *International Journal of Peptide Research and Therapeutics*, *26*(2), 1155. https://doi.org/10.1007/S10989-019-09918-Z
19. Raucci, R., Laine, E., & Carbone, A. (2018). Local Interaction Signal Analysis Predicts Protein-Protein Binding Affinity. *Structure (London, England : 1993)*, *26*(6), 905-915.e4. https://doi.org/10.1016/J.STR.2018.04.006
20. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, 2nd ed. In *Reinforcement learning: An introduction, 2nd ed.* The MIT Press.
21. Vangone, A., & Bonvin, A. M. J. J. (2015). Contacts-based prediction of binding affinity in protein–protein complexes. *ELife*, *4*(JULY2015). https://doi.org/10.7554/ELIFE.07454
22. Vangone, A., & Bonvin, A. M. J. J. (2017). PRODIGY: A Contact-based Predictor of Binding Affinity in Protein-protein Complexes. *Bio-Protocol*, *7*(3). https://doi.org/10.21769/BIOPROTOC.2124
23. Xue, L. C., Rodrigues, J. P., Kastritis, P. L., Bonvin, A. M., & Vangone, A. (2016). PRODIGY: a web server for predicting the binding affinity of protein–protein complexes. *Bioinformatics*, *32*(23), 3676–3678. https://doi.org/10.1093/BIOINFORMATICS/BTW514
24. Yang, Y. X., Huang, J. Y., Wang, P., & Zhu, B. T. (2023). AREA-AFFINITY: A Web Server for Machine Learning-Based Prediction of Protein-Protein and Antibody-Protein Antigen Binding Affinities. *Journal of Chemical Information and Modeling*, *63*(11), 3230–3237. https://doi.org/10.1021/ACS.JCIM.2C01499/ASSET/IMAGES/LARGE/CI2C01499_0005.JPEG
25. Yuan, Y., Chen, Q., Mao, J., Li, G., & Pan, X. (2023). DG-Affinity: predicting antigen–antibody affinity with language models from sequences. *BMC Bioinformatics*, *24*(1), 1–12. https://doi.org/10.1186/S12859-023-05562-Z/FIGURES/6
26. Zhou, J., Le, C. Q., Zhang, Y., & Wells, J. A. (2024). A general approach for selection of epitope-directed binders to proteins. *Proceedings of the National Academy of Sciences*, *121*(19). https://doi.org/10.1073/PNAS.2317307121