Journal Club

Self-play reinforcement learning guides protein engineering

Received: 30 November 2022

Accepted: 20 June 2023

Yi Wang[1], Hui Tang[1], Lichao Huang[1], Lulu Pan[2], Lixiang Yang[1], Huanming Yang[3,4], Feng Mu[1] ✉ & Meng Yang[1] ✉

Yifan Qin

25-05-09

Introduction

- Under protein engineering, direct evolution employs random mutation to get candidates
- Functional assay are applied for scoring candidates
- Vulnerable to local minima and discards enormous mutations

Introduction

- Machine learning models are trained on labelled variants and sequences
- Candidates are greedily proposed and searched for functional properties predicted by the model
- Reduces wet-lab burden, but efficient sampling on vast sequence space is still challenging

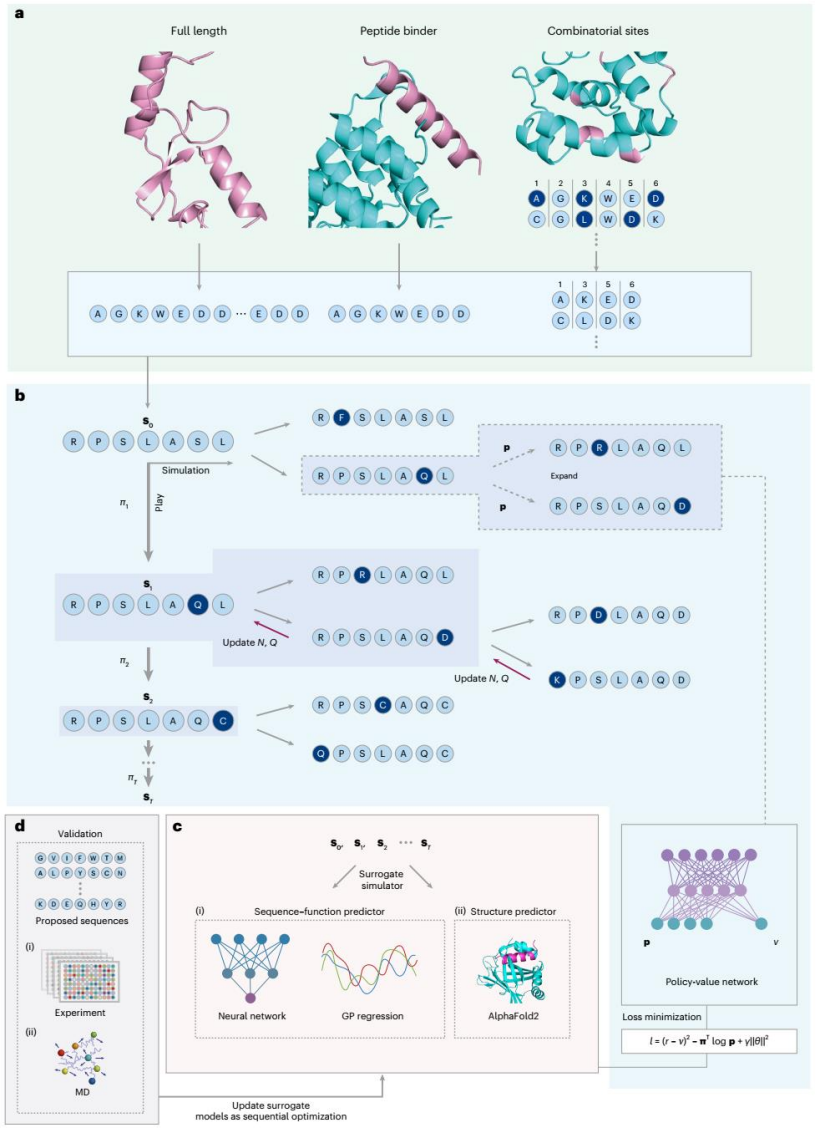
Introduction

- Reinforcement learning (RL) allows agent to learn how to perform actions by interacting with environment
- Agent also learns how to maximize a reward function using actions
- Some protein design methods utilize such paradigm but faced with sparse reward problem

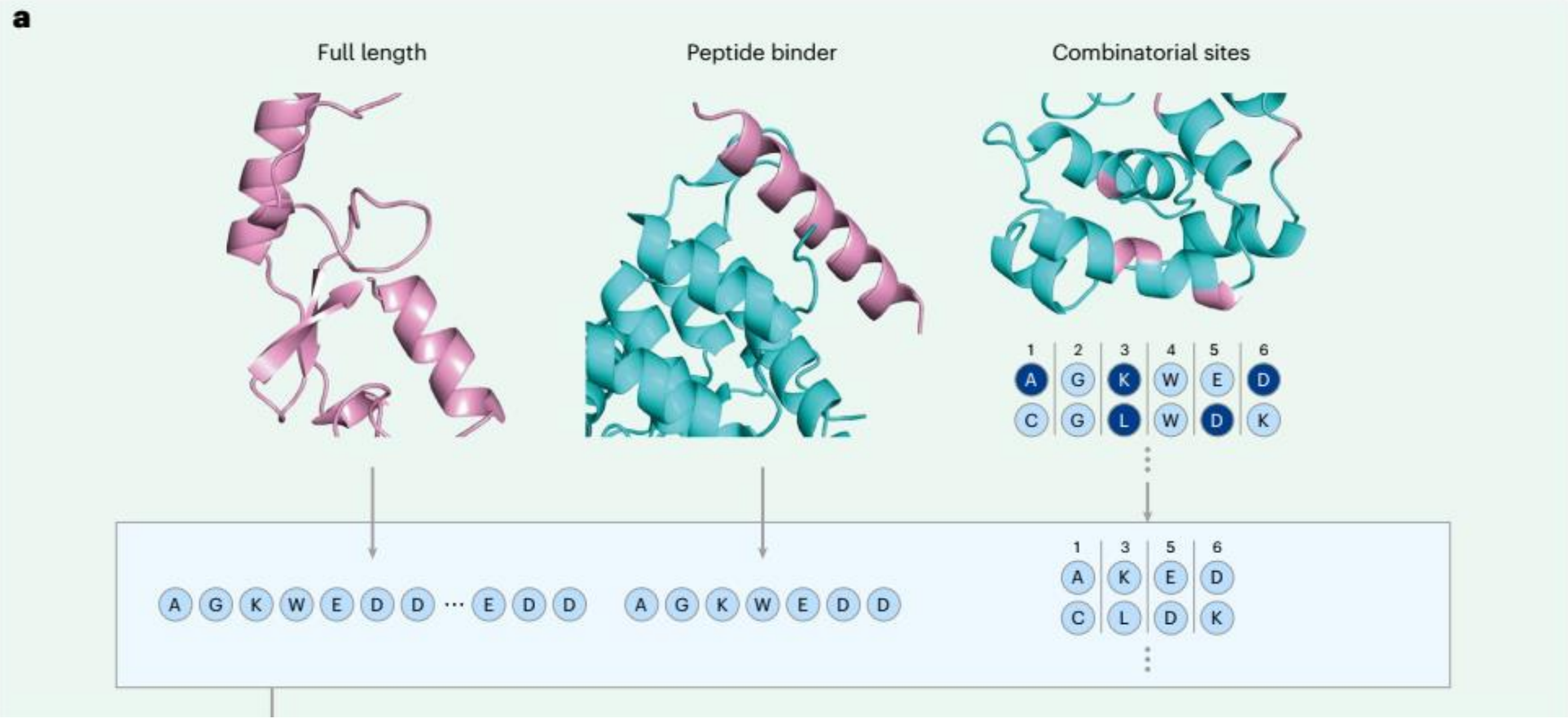
Introduction

- AlphaZero's self-play RL has mastered a suite of chess games and shown promise in solving combinatorial optimization problems
- **EvoPlay** adapts self-play RL to single-player optimization problem
- goal-directed protein design or directed evolution

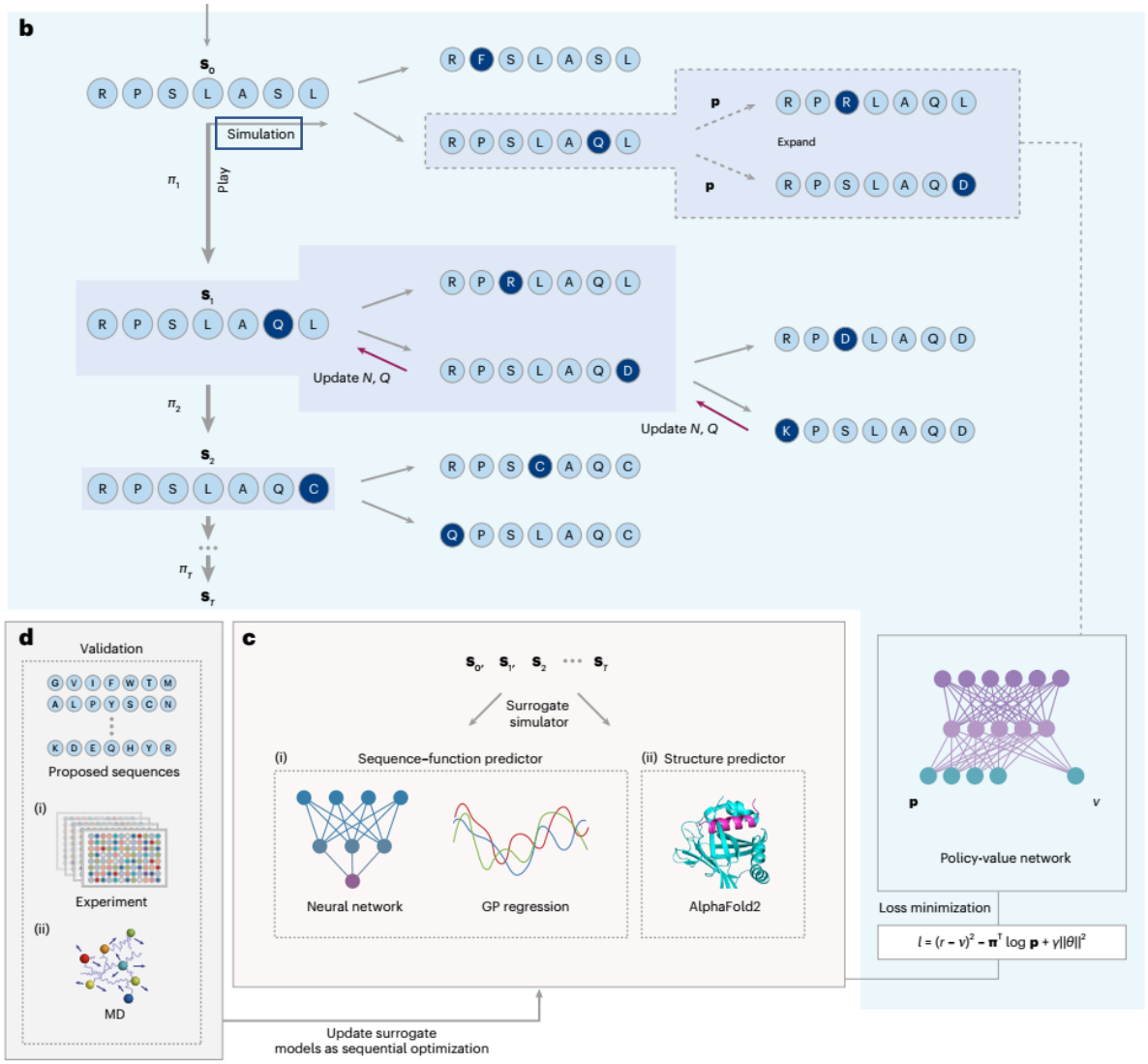
EvoPlay: overview



EvoPlay: Input



EvoPlay: Simulation

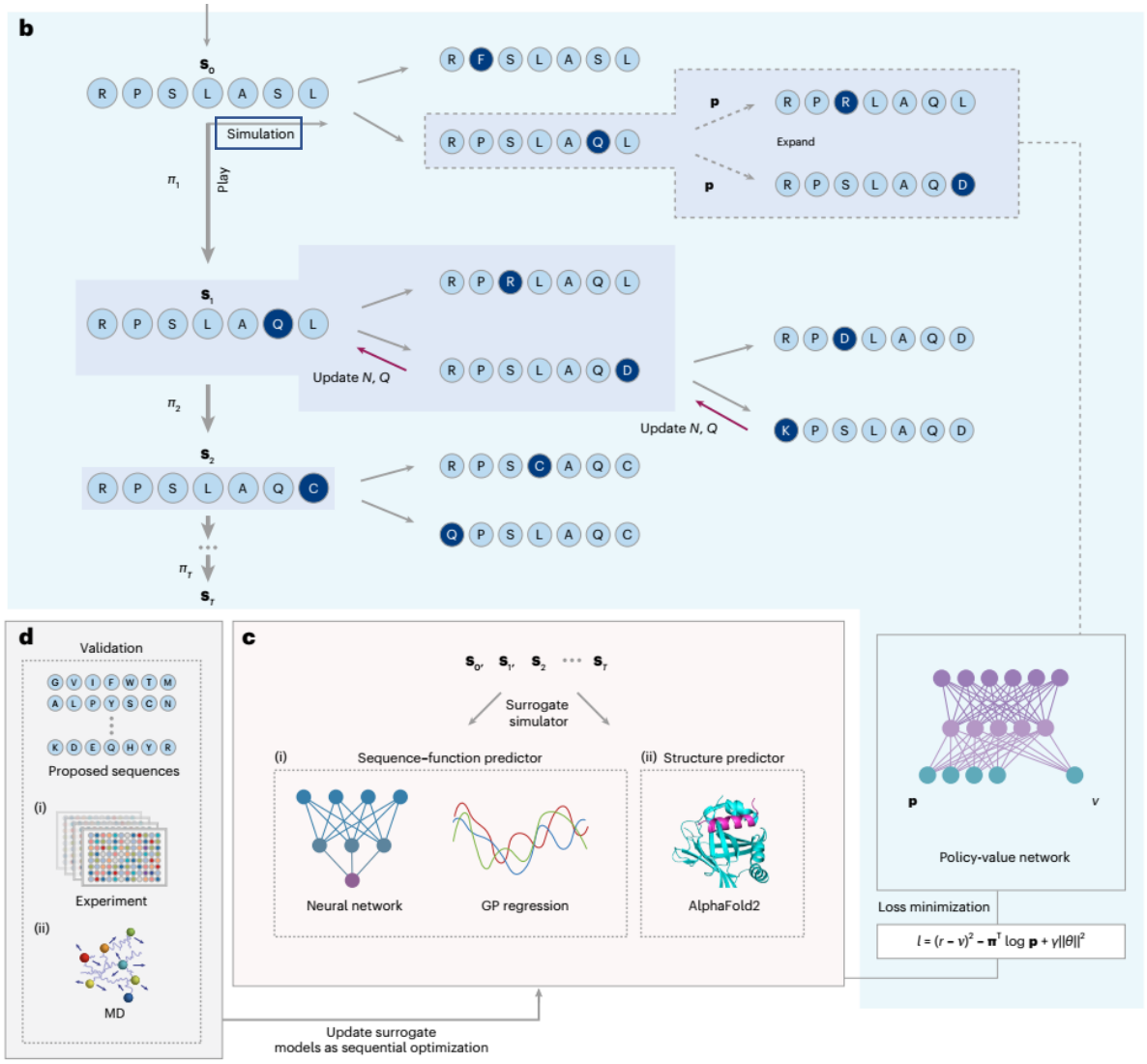


- Given sequence s

$$P_s, v = f_\theta(s)$$

- $P_s \in \mathbb{R}^{20\times L}, v \in \mathbb{R}$
- a is one of action in $20 \times L$ actions
- New node is selected based on $U_{s,a} + Q_{s,a}$

EvoPlay: Simulation



- New node is selected based on $U_{s,a} + Q_{s,a}$

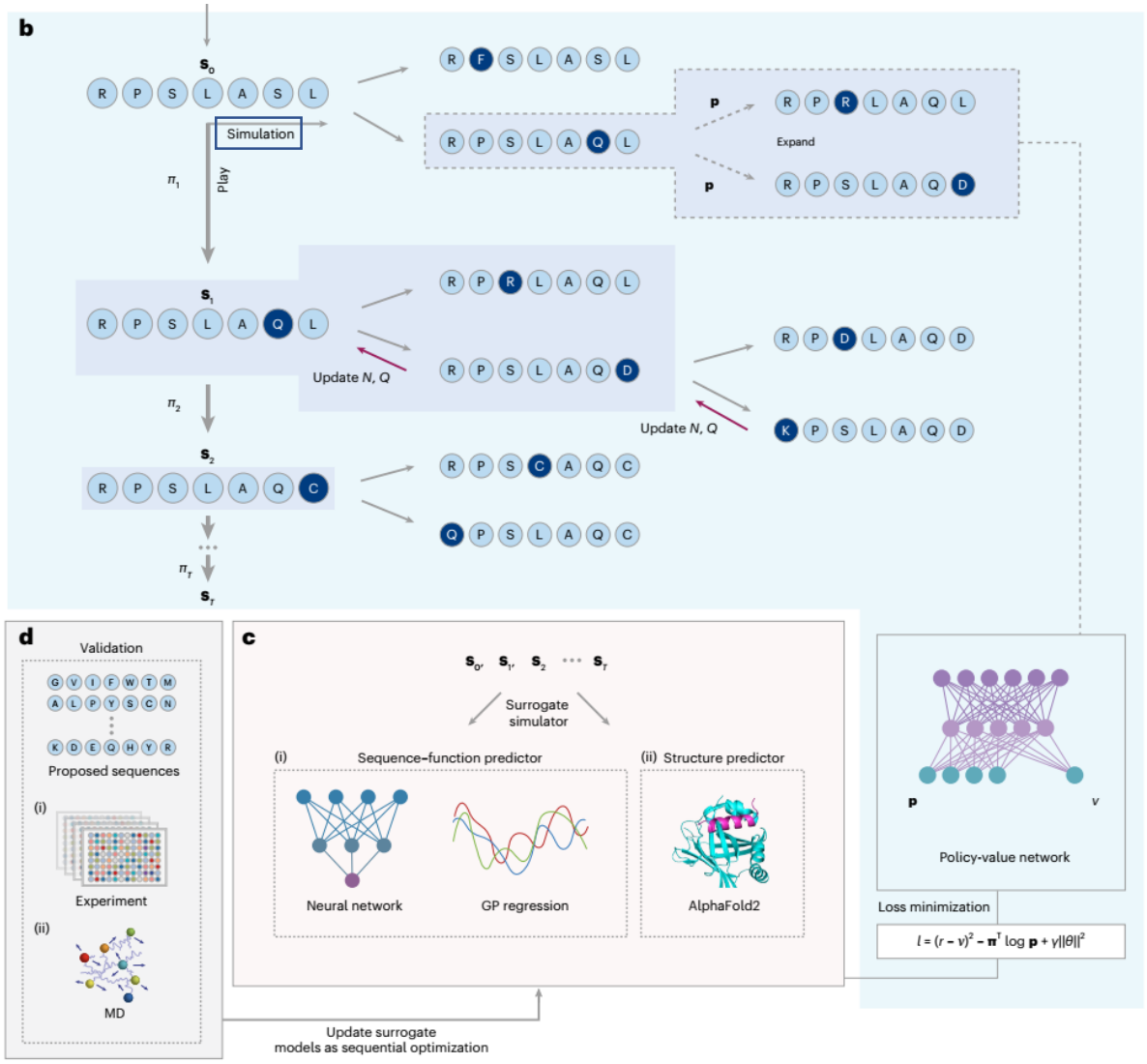
$$U_{s,a} = CP_{s,a}\frac{\sqrt{N_{s,a}^{parent}}}{1 + N_{s,a}}$$

- If action a selected

$$N_{s,a} = 1 + N_{s,a}$$

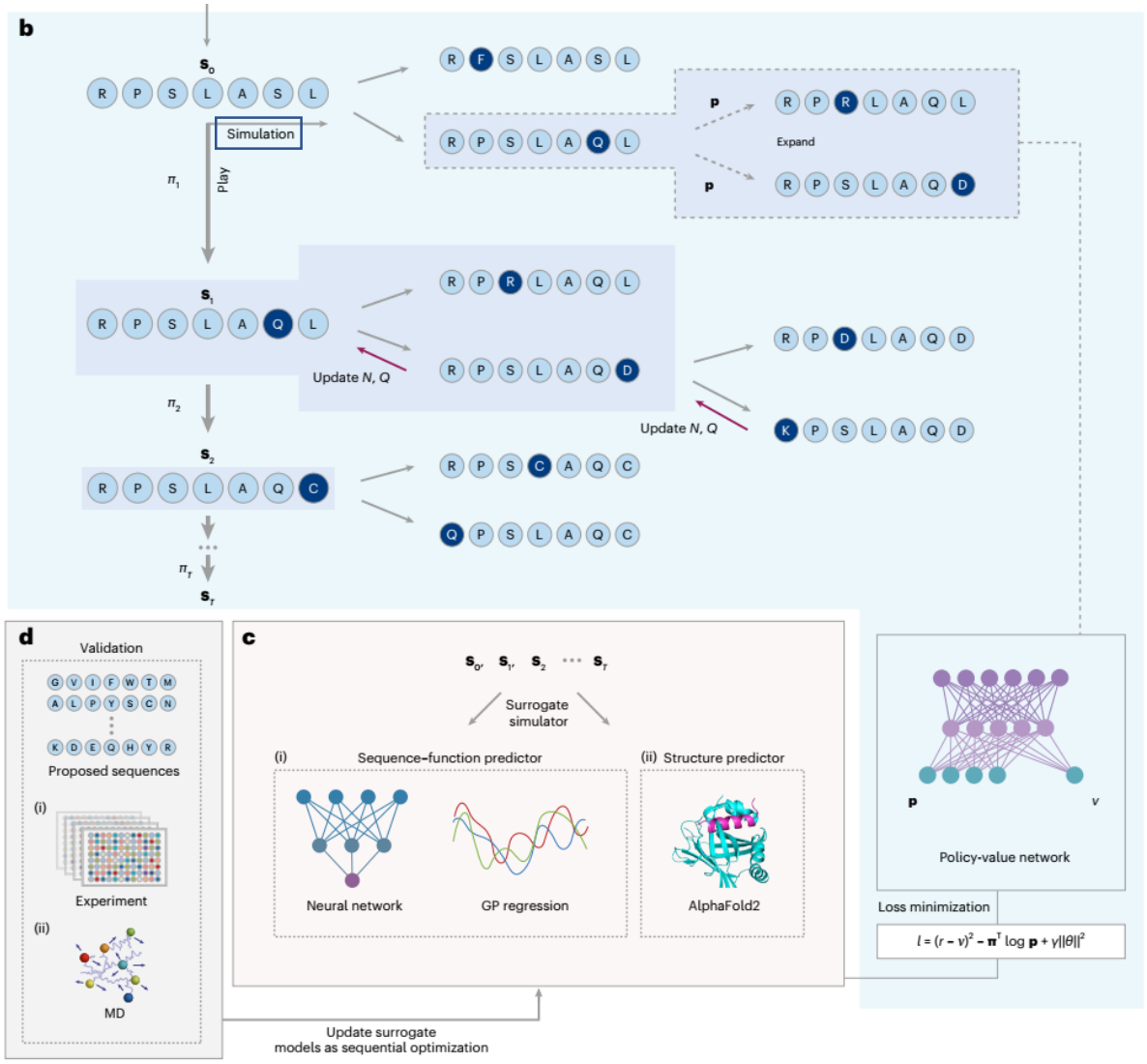
$$Q_{s,a} = \frac{Q_{s,a}(N_{s,a} - 1) + v}{N_{s,a}}$$

EvoPlay: Simulation



- Stop a trajectory when new node expanded or end condition reached
  - Selected action lead to explored sequence or less reward sequence
- Simulation lasts for ~400 trajectories

EvoPlay: Simulation

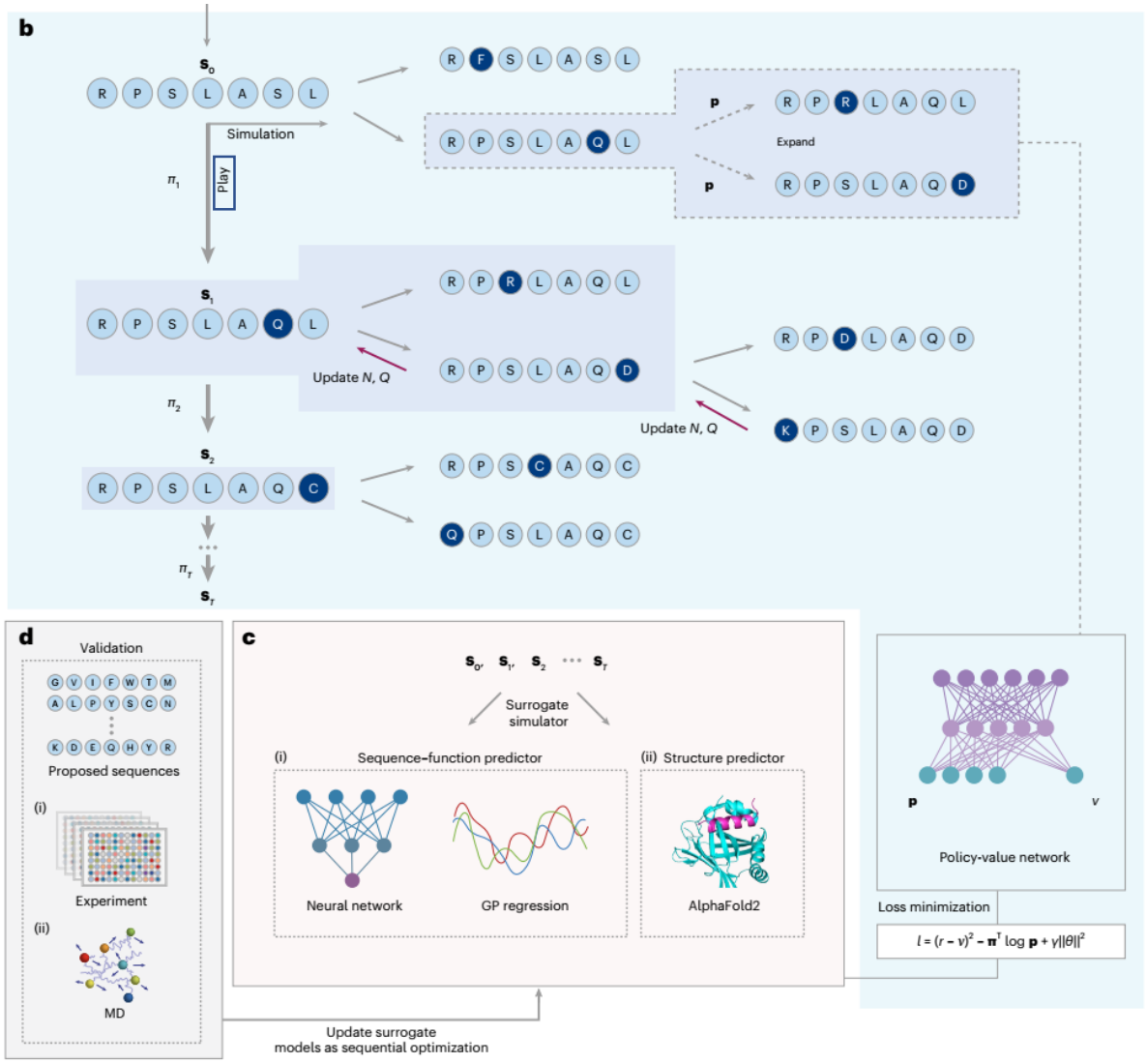


- Loss

  $\mathcal{L} = (r - v)^2 - \boldsymbol{\pi}^T \log \boldsymbol{P} + \gamma \left|\left|\theta\right|\right|^2$

- Trajectories (s, a, r) are used to train the network θ
  - r is from surrogate simulator (predictor/AF2)

EvoPlay: Play

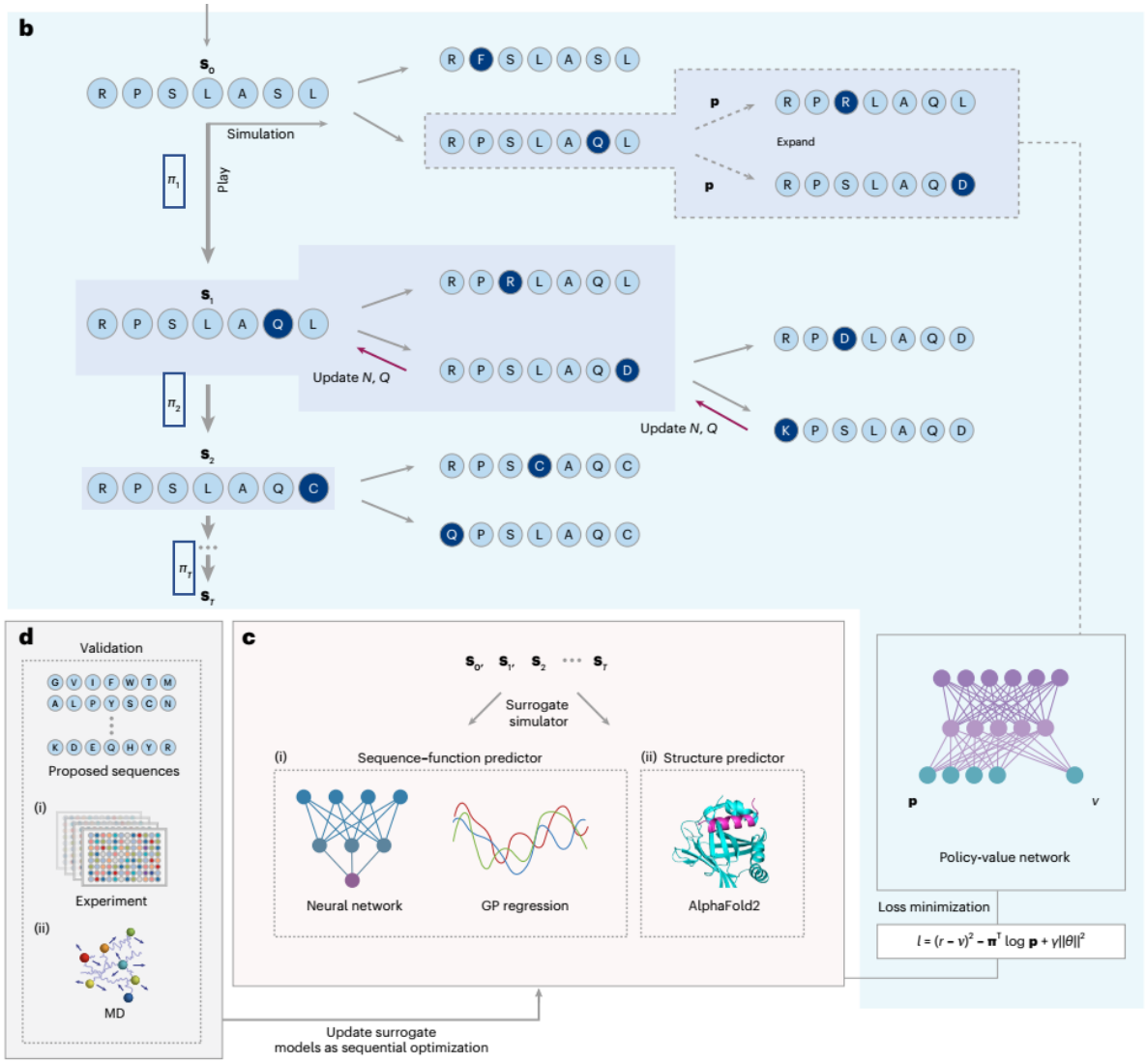


- Given simulation result
- A real action π_t to s_t is sampled according to visit count

$$\pi_t \propto N(s_t, a)^{1/\tau}$$

- τ is a tunable temperature parameter

EvoPlay: Final



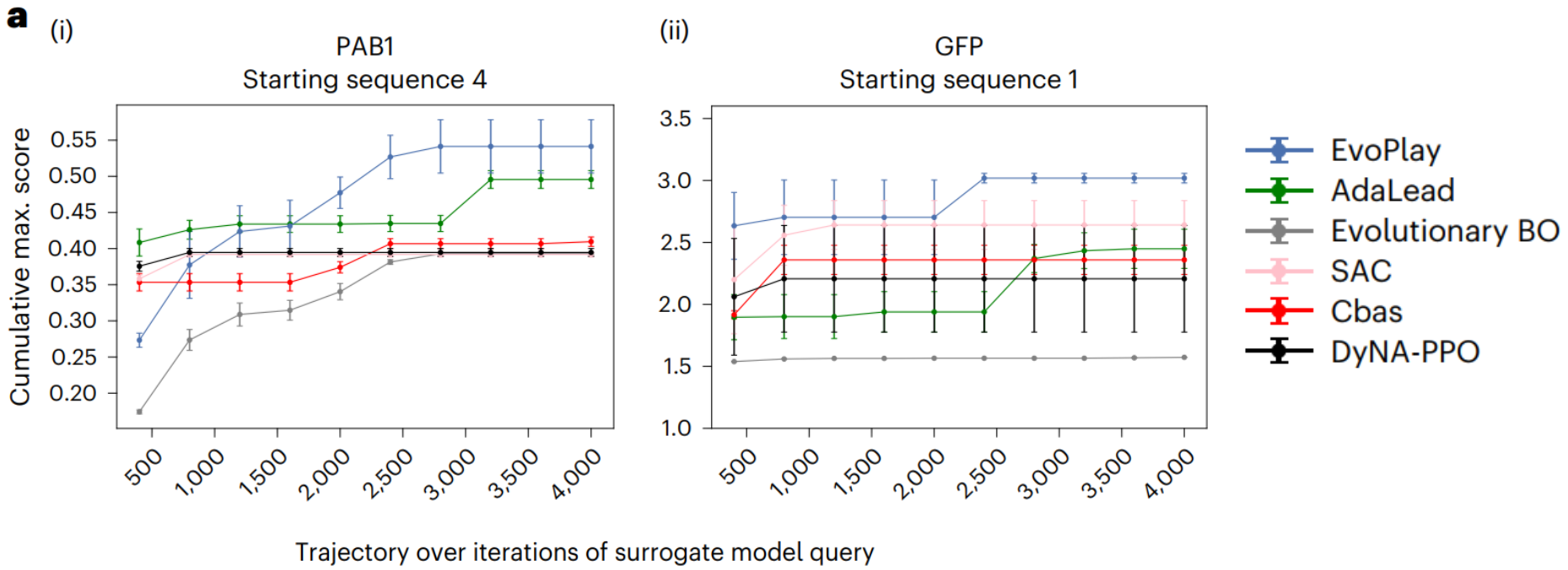
- We get the final sequence after
  1. Doing T mutations(real play)
  2. Early stop if
     1. The score of new sequence is lower
     2. We do not change the sequence or is explored during real play

Using EvoPlay for design

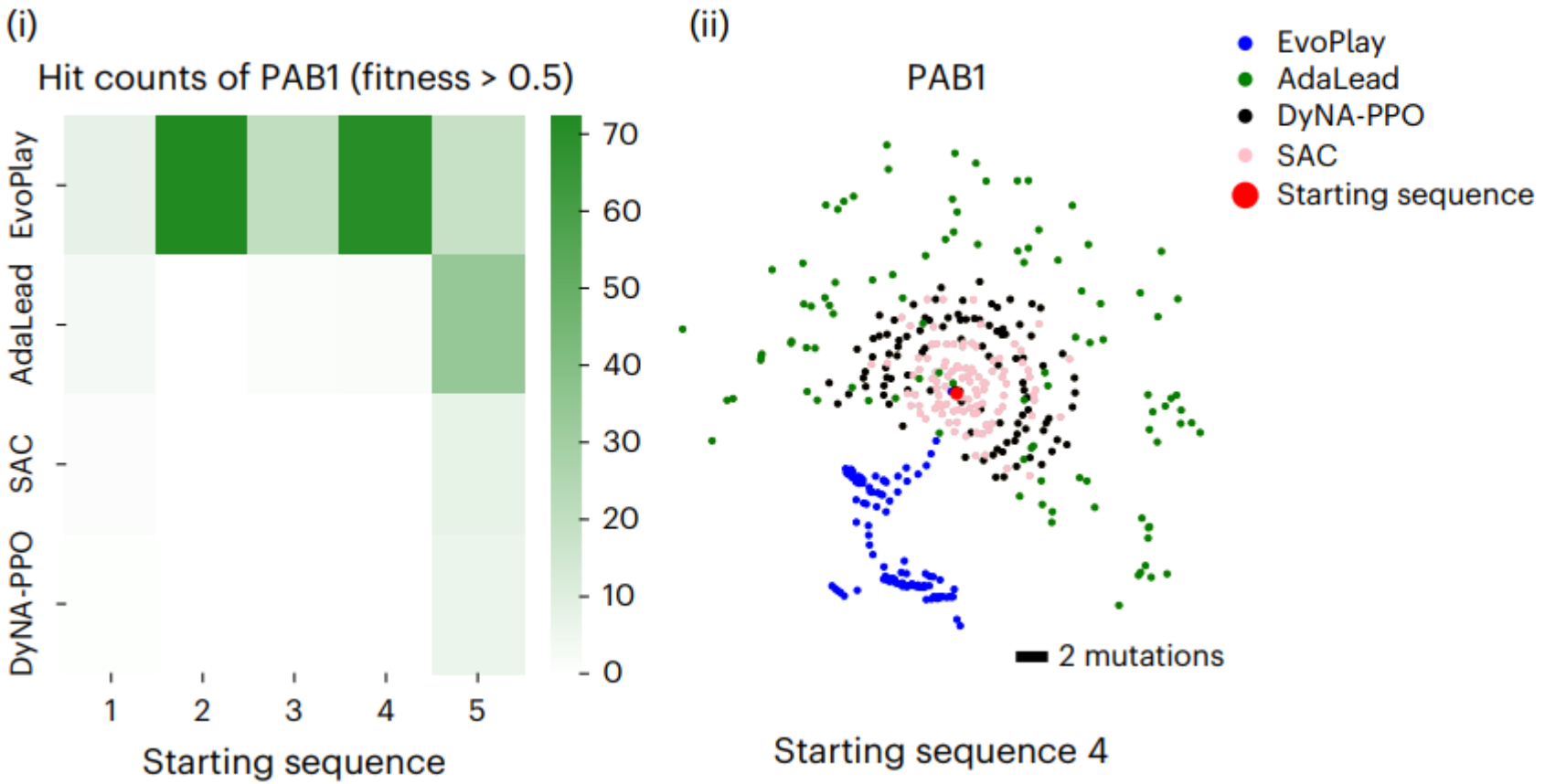
- Use GFP and poly(A)-binding protein (PAB1) for evaluation
- Action space: 20^{237} and 20^{44}
- Fitness:
  - GFP: fluorescence intensity
  - PAB1: XY Enrichment score
- 5 starting sequences for each type of protein

Using EvoPlay for design

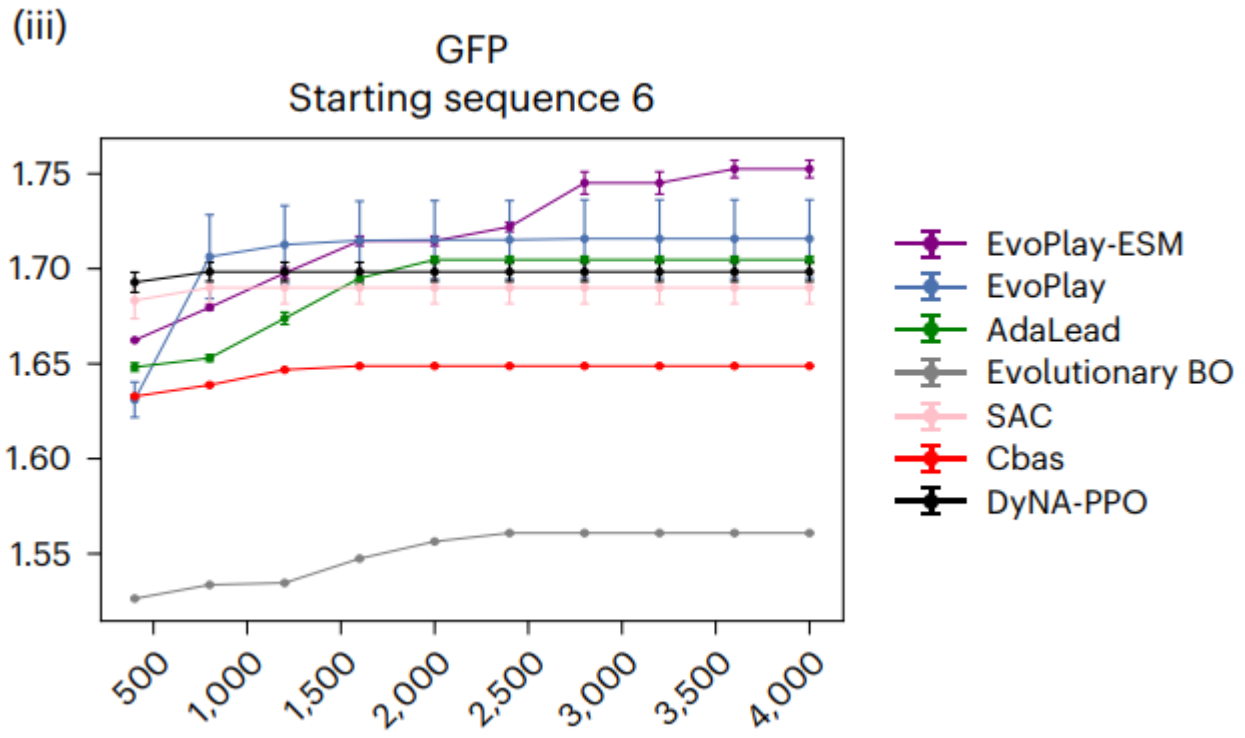
- CNN as surrogate model (sequence–function model)
  - Given sequence, gives fitness (reward r)



Using EvoPlay for design



Using EvoPlay for design



Summary

- Use Monte Carlo Tree Search for designing protein sequence with higher fitness score
- Validate the method in designing protein in full length for GFP and PAB1
- (not covered)Peptide design with AF2 as surrogate model, combinatorial site design and GLuc engineering shows promising results