Extended Abstract

Motivation. Protein engineering in automated cloud labs is a sequential decision problem: under a fixed dollar budget, an agent must repeatedly decide what sequences to propose and how to measure them, choosing between cheap, noisy proxy assays and expensive, accurate "gold" assays that occasionally fail to synthesize. This is naturally a reinforcement learning (RL) problem, and crucially its action factorizes: what to propose (library and batch size) is a per round decision, while which assay to use is a per candidate decision. A single flat policy cannot express the latter because it must commit one assay fraction across an entire batch. I ask whether a hierarchical policy that routes each candidate individually buys a real advantage, and under what conditions that advantage holds.

Method. I build CloudLabDP, a POMDP over a two tier assay protein design campaign on GFP, scored in $[0, 1]$ by an ESM-2 oracle over ≤ 12 rounds. My agent factors the policy into two heads trained jointly inside a single environment step: a high level PPO policy π^{hi} that selects the library (single site / combinatorial / ML designed) and batch size, and a low level REINFORCE policy π^{lo} that outputs a per candidate probability of routing to the gold assay, $\pi^{lo}(s)_i = \sigma(\mathrm{MLP}(\mu_i, v_i, b_t, t/T))$, from belief ensemble mean/variance and budget/round fractions. Credit is assigned per candidate: $r_i = \frac{\Delta\text{best}}{\#\text{winners}}\, i\in\text{winners} - \lambda\,\mathrm{cost}(\text{tier}_i)$. I compare against three baselines that each ablate one capability: Random, Bayesian optimization (SAMPLE style, gold only), and a flat DyNA-PPO policy with a fleet wide gold fraction.

Implementation. The environment, belief ensemble (1D CNN committee), and all four agents are implemented in PyTorch with Stable Baselines3 for PPO; π^{lo} is a custom online REINFORCE update executed inside `env.step`. I evaluate on a 16 cell grid with budget $B \in \{60, 120, 240, 480\}$ crossed with gold failure rate $\rho \in \{0.2, 0.4, 0.6, 0.8\}$. There's 4 evaluation seeds per cell, plus router and library ablations and a belief augmented variant, all trained on Modal A10G GPUs.

Results. The hierarchical advantage exists but is narrow. At the tightest budget (B=60, ρ=0.2) HRL attains the best mean fitness (0.611) of all four agents and it is ahead of flat PPO (0.561). However, as the budget grows, the gap closes and flat PPO overtakes (B=240: 0.899 vs. 0.809). The agents converge on qualitatively distinct spending policies: Bayesian Optimization is locked to 100% gold, flat PPO ramps gold from 33% to 65% as budget grows, while HRL stays proxy heavy ($\sim$1/3 gold) regardless of budget. A router ablation confirms the learned per candidate routing carries the method ($0.809 \rightarrow 0.731$ when randomized), whereas a fixed library ablation does not hurt (0.884). The high level library choice adds little. At tight budget HRL's failure blind allocation makes it the worst agent at high failure (ρ=0.6: 0.451), and exposing the observed failure rate as an input did not rescue it.

Discussion. The advantage of per candidate routing concentrates exactly where intuition predicts: when assays are scarce relative to the task. It disappears when budget is abundant. The fragility result is the more useful contribution. It localizes the weakness to a structural, not informational, gap. HRL learned a static allocation that ignores how often gold actually fails, and giving it the failure signal was insufficient because nothing in training forced it to adapt. Spending gold "per protein" is the right abstraction, but only if the routing head is also made failure aware during training.

Conclusion. In tight budget, low failure settings of cloud lab protein design, hierarchical, per candidate assay routing yields an advantage. I recommend a concrete fix based on my results: randomize the gold failure rate ρ across episodes during training so the router is forced to learn that gold becomes a worse buy as it fails more often and to cut its gold spending accordingly.

Hierarchical RL for Cost Aware Protein Engineering Campaigns in Cloud Labs

Emilin Mathew
Department of Computer Science and Biology
Stanford University
emilinm@stanford.edu

Abstract

Cloud labs have automated protein engineering campaigns which pose a sequential resource allocation problem. Under a fixed budget, an agent needs to choose which sequence library to propose and which of two assay tiers (a cheap noisy proxy or an expensive, failure prone gold assay) to spend on each candidate. The action factorizes into a per round proposal decision and a per candidate routing decision which a flat policy structurally can not express. I present CloudLabDP, a POMDP benchmark on GFP design scored by an ESM-2 oracle, and a hierarchical reinforcement learning (HRL) agent that pairs a high level PPO library policy with a low level REINFORCE router that assigns each candidate its own probability of gold measurement from belief features. Across a 16 cell budget$\times$failure grid against Random, Bayesian optimization, and flat DyNA-PPO baselines, HRL attains the best fitness in the tight budget and low failure setting (0.611 at B=60). It performs worse as budget grows, specifically at high failure rates because it learns a failure blind allocation. Ablations show the learned router, not the library head, drives the gain. A belief augmented variant confirms the high failure collapse is structural rather than a matter of observability. Per candidate routing is the right abstraction but it must be made failure aware in training.

1 Introduction

Autonomous protein engineering campaigns in cloud labs are operational planning problems. There are many sequential decisions such as deciding which mutational library strategy to use, which assay type to route candidates through, and how to recover when failures occur. Each decision comes with a different cost, latency, and yield. The effects also compound over time as one round's data shapes what gets proposed next.

Existing research in cloud labs primarily focus on sequence design methods (Bayesian optimization in SAMPLE (4), model based PPO in DyNA-PPO (1), and GFlowNets (2)), but they collapse the sequential decision making structure by assuming a single uniform cost. This fails to provide the most utility to cloud labs because an expression only screen might cost orders of magnitude less than a kinetic characterization run, or a substantial fraction of submitted sequences might fail upstream of any fitness measurement at all. These all come with real costs.

Furthermore, the per round action factorizes. What to propose (which mutation library and how large a batch) is a single decision shared by the whole round. But how to measure whether a given candidate is worth a gold assay is intrinsically a per candidate decision; a promising sequence deserves an accurate readout, while a low predicted fitness sequence one does not. A flat policy that emits a single gold fraction for the batch cannot express this because it commits the same measurement mix to every candidate it proposes. This mismatch is exactly the kind of structure hierarchical reinforcement learning (HRL) is meant to exploit.

Stanford CS224R 2025 Final Report

I investigate whether a hierarchical policy that routes each candidate individually offers an advantage in cloud lab protein design or where that advantage breaks down. I contribute: (i) CloudLabDP, a configurable POMDP benchmark for budget aware, two tier protein design campaigns on GFP with an ESM-2 fitness oracle, a CNN belief ensemble, and stochastic synthesis failures; (ii) a hierarchical agent that pairs a high level PPO library policy with a low level REINFORCE router trained jointly inside one environment step, with per candidate credit assignment; and (iii) an empirical study across a 16 cell budget×failure grid. I found that HRL has an advantage under tight budgets and low simulated failures.

2 Related Work

RL for biological sequence design. DyNA-PPO (1) modeled sequence design as an MDP over iterative mutation steps and trained PPO against a learned proxy fitness model. This inspired work like GFlowNets with Bayesian active learning (2) and δ-Conservative Search (3), which have improved sample efficiency and diversity. However, all of these position the wet lab evaluator to be a black box that returns a single scalar fitness value per query. They have no notion of heterogeneous assay costs or upstream failures.

Bayesian autonomous protein engineering. The landmark SAMPLE (4) work used four Bayesian optimization agents to model the fitness landscape and select the most promising candidates on a thermostability engineering task. They found high performing variants while testing fewer than 2% of possible sequences. The iBioFAB biofoundry extended this to campaigns with protein language model designed libraries, achieving 16–26$\times$ activity improvements in four rounds (5). Both are end-to-end demonstrations of autonomous lab campaigns, but their decision making remains single level. The library strategy is fixed by a human before the campaign starts, the assay tier is fixed by the platform, and failure handling is procedural rather than learned. My SAMPLE style Bayesian Optimization (BO) baseline captures this approach.

Benchmark substrates. The Fitness Landscape EXploration Sandbox (FLEXS) is an open source simulator for model based sequence design (6). Each landscape has a ground truth oracle $g(x)$ that is expensive to query, so algorithms interact with a noisy proxy $f(x) \approx g(x)$. Oftentimes, you will run cheap model queries before committing to ground truth measurements that incur real cost. My environment follows a similar two level structure: first a proxy screen, then a gold assay, but I add the operational cost and failure layer FLEXS lacks. I use an ESM-2 (7) oracle over GFP as the fitness signal (see Section 4). While I considered ProteinGym and FLIP, they lack the interactive simulator interface needed for RL training.

Hierarchical RL. The options framework (8) decomposes control into high and low level policies. I borrow the decomposition but invert the usual time abstraction motivation. My hierarchy is not over temporally different options but over the within step factorization of a proposal decision (π^{hi}, PPO (9)) and a per item routing decision (π^{lo}, REINFORCE (10)).

The gap. To the best of my knowledge, no prior work treats the assay layer as a structured, heterogeneous cost decision, makes stochastic operational failures (i.e. synthesis failure, contamination, drift) part of the agent's environment, or compares learned policies against BO baselines under a fixed dollar budget rather than a query count budget. Framing two tier assay allocation as a per candidate routing head trained jointly with a library policy is novel and the failure mode analysis is my project's main empirical contribution.

3 Method

3.1 The CloudLabDP Environment

My simulated protein design campaigns run for up to T=12 rounds under a fixed budget B. The state tracks the best validated gold fitness so far, the fraction of budget and rounds remaining, the last library used, and recent per tier success rates. Each round the agent (1) chooses a library from {single site, combinatorial (k=2), ML designed} and a batch size; (2) the library operator generates candidate sequences; and (3) each candidate is routed to one of two assay tiers. I calibrated the cost

ratio from literature and partial Strateos/Twist rate cards, which justify a $1 \times / 10\times$ ratio. The proxy tier costs \$1 with observation noise $\sigma_{\text{proxy}} \approx 0.30$ and never fails, while the gold tier costs \$10 with $\sigma_{\text{gold}} \approx 0.02$ but fails to synthesize with probability ρ (15–30% at baseline). In the failure case, the spend is lost and no measurement is returned. The agent learns to anticipate these failures through its belief state. I measure fitness using the ESM-2 pseudo likelihood of a GFP variant mapped to $[0, 1]$; the episode ends when the budget is exhausted or T rounds elapse.

Formally, the state at round t is $s_t = (\mathcal{H}_t, b_t, B_t)$, where $\mathcal{H}_t$ is the campaign history (sequences submitted, tier used, noisy observation, success/failure flag), b_t is a learned belief over the fitness landscape, and B_t is the remaining budget. The agent only sees a noisy, tier conditional measurement $\tilde{f}(x) = f(x) + \varepsilon_{\text{tier}}$ with $\varepsilon_{\text{tier}} \sim \mathcal{N}(0, \sigma^2_{\text{tier}})$. The reward is $R = \max_{\{x:\text{tier}=\text{gold}\}} \tilde{f}(x)$, the best validated fitness observed, delivered as the per round improvement in that quantity. Therefore, only successful gold measurements can advance the objective, making the routing decision economically consequential.

3.2 Belief model

A group of 1D convolutional networks over one hot sequences predicts fitness mean μ_i and epistemic variance v_i for any protein candidate. Each round, the ensemble is refit on all successful observations, weighted by inverse tier variance so gold readouts dominate ($1/0.02^2$ vs. $1/0.30^2$). The belief drives the ML designed library and supplies the features the router uses to decide which candidates deserve a gold assay.

3.3 Hierarchical policy

I factor the policy into two heads trained jointly inside one environment step.

High level π^{hi} (PPO). A PPO policy maps the observation to a 4D action which selects what to propose this round in terms of choosing a library and batch size.

Low level π^{lo} (REINFORCE). Given the candidates proposed by π^{hi}, the router emits an independent gold probability for each:

$$\pi^{lo}(s)_i = \sigma\big(\text{MLP}(\mu_i,\, v_i,\, b_t,\, t/T)\big), \tag{1}$$

from the belief mean/variance of candidate i and the current budget fraction b_t and round fraction t/T. π^{lo} conditions on each candidate's predicted fitness and uncertainty.

The router is trained online with REINFORCE using a per candidate reward

$$r_i = \frac{\Delta\text{best}}{\#\text{winners}}\, _{i \in \text{winners}} - \lambda \cdot \text{cost}(\text{tier}_i), \tag{2}$$

where "winners" are candidates whose successful gold measurement ties the new best fitness. Under this model, improvement credit is shared only among the candidates that actually produced it and every candidate pays a cost penalty for the tier it consumed. The penalty λ is important: at the default λ=0.05 the router collapsed to all proxy (gold costs $10\times$ but typical per step improvement is < 0.1), so I set λ=0.005 to keep gold economically viable. The high level policy is trained with Stable Baselines3 PPO and the low level update is a custom REINFORCE step executed inside `env.step`, so both heads learn from the same trajectory.

3.4 Baselines

Each baseline ablates one capability my full method needs (Table 1). Random samples actions uniformly (no belief and no learned routing). BO (SAMPLE style) uses the same belief ensemble with UCB acquisition (β=2.0) but is locked to the combinatorial library and the gold tier. Flat PPO (DyNA-PPO style) learns the library and batch size and a single fleet wide gold fraction, but cannot route per candidate. Only HRL covers the full decision space.

Table 1: Baselines as capability ablations.

Method	What it does	Missing
Random	Pick candidates at random	belief + routing
BO (SAMPLE)	Belief, single library, gold only	library + routing
Flat PPO	Library + fleet wide gold fraction	per candidate routing
HRL	Belief + library + per-candidate routing	—

4 Experimental Setup

I evaluate on a 16 cell grid crossing budget $B \in \{60, 120, 240, 480\}$ with gold failure rate $\rho \in \{0.2, 0.4, 0.6, 0.8\}$, running all four agents with 4 evaluation seeds per cell. The primary metric is the best validated gold fitness at episode end. I also report the realized gold fraction $n_{\text{gold}}/(n_{\text{gold}} + n_{\text{proxy}})$ to characterize learned spending behavior and fitness per cost as an efficiency check. The budget sweep fixes ρ=0.2, the robustness sweep fixes B=240, and a joint sweep crosses $B \in \{60, 120, 240\}$ with $\rho \in \{0.2, 0.4, 0.6\}$ to probe the tight budget×high failure corner directly. Flat PPO is trained for 200k environment steps and HRL for up to 300k each with 4 parallel environments on Modal A10G GPUs. The belief ensemble uses 5 members refit every round in the full configuration. I additionally run two ablations at the B=240, ρ=0.2 cell. I use a random router (fixed $P(\text{gold})$=0.5), a fixed library (combinatorial only), and a belief augmented HRL that adds the observed gold failure rate to the observation, trained directly in the high failure regime.

5 Results

5.1 Quantitative Evaluation

Table 2 reports best gold fitness across the budget and robustness sweeps. I have three main findings.

Finding 1: HRL has an advantage at tight budgets. At the tightest budget (B=60, ρ=0.2), HRL's mean fitness (0.611) is the highest of all four agents, ahead of flat PPO (0.561), BO (0.553), and Random (0.585) (Figure 1). As budget grows the advantage not only closes but reverses: by B=240 flat PPO leads at 0.899 versus HRL's 0.809, and at B=480 every method approaches the fitness ceiling (0.92–1.00). The interpretation is intuitive: when gold assays are scarce relative to the task, deciding which candidate earns one matters. However, when the budget is large, any reasonable policy buys enough gold to catch the same peaks.

Finding 2: agents learn qualitatively distinct allocation strategies. The four agents have different spending policies (Figure 2). BO is structurally locked to 100% gold. Flat PPO ramps its gold fraction up with budget, from 0.33 at B=60 to 0.65 at B=480, learning to buy more accurate assays when it can afford them. HRL instead stays proxy heavy throughout, holding near ∼1/3 gold ($0.33 \rightarrow 0.44$) almost independently of budget and nearly identically across seeds. HRL discovers a conservative strategy: lean on cheap proxy signal for belief and spend gold selectively. This is a sensible policy at tight budget, but, as Finding 3 shows, its budget invariance causes its failure.

Finding 3: The edge is fragile to assay failure. The conservative allocation is failure blind. HRL holds the same ∼1/3 gold fraction regardless of how often gold fails. In the joint sweep at tight budget (B=60), HRL falls from best at low failure (ρ=0.2: 0.611) to worst at high failure (ρ=0.6: 0.451, versus flat PPO 0.544, Random 0.527, BO 0.517)(Figure 3). Because it keeps spending the same share on gold even as gold stops paying off, it wastes budget on failed syntheses even when budget is scarcest. The hierarchical advantage thus holds only in the tight budget, low failure corner of the grid.

Table 2: Best gold fitness (mean over 4 seeds). Left: budget sweep at ρ=0.2. Right: robustness sweep at B=240.

	Budget sweep (ρ=0.2)				Robustness sweep (B=240)			
Agent	B=60	120	240	480	ρ=.2	.4	.6	.8
Random	0.585	0.659	0.836	1.000	0.987	0.894	0.807	0.686
BO (SAMPLE)	0.553	0.603	0.778	0.973	0.927	0.856	0.748	0.651
Flat PPO	0.561	0.690	0.899	0.989	0.949	0.899	0.778	0.706
HRL	0.611	0.616	0.809	0.923	0.860	0.947	0.740	0.708

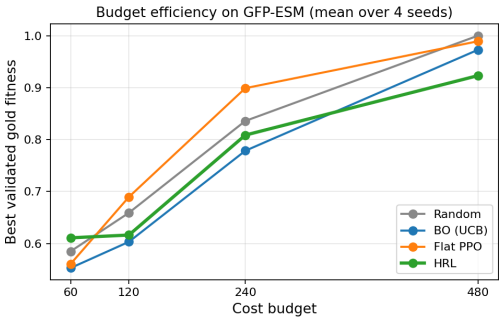

Figure 1: Finding 1. Best gold fitness vs. budget (ρ=0.2). HRL leads at B=60; methods converge and flat PPO overtakes as budget grows.

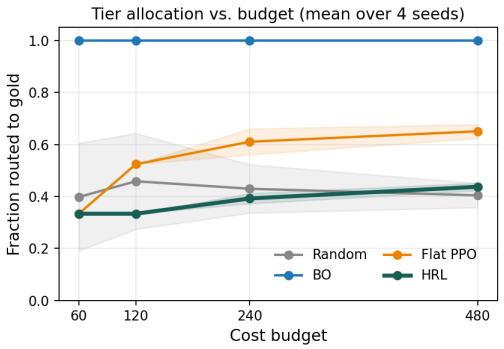

Figure 2: Finding 2. Realized gold fraction vs. budget. BO locked at 100%; flat PPO ramps up; HRL stays proxy heavy.

Ablation Findings: The router, not the library, carries the method. At the B=240, ρ=0.2 cell (Table 3), replacing the learned router with a fixed $P(\text{gold})$=0.5 drops fitness from 0.809 to 0.731, confirming that per candidate routing is the component doing the work. Surprisingly, fixing the library to combinatorial does not hurt (0.884, slightly above full HRL). The high level library policy adds little value over a sensible fixed choice on this landscape, and may even introduce optimization variance. The hierarchy's value lives almost entirely in π^{lo}, not π^{hi} which directly motivates simplifying the high level in future work.

Table 3: Ablations at B=240, ρ=0.2 (mean best gold fitness over 4 seeds). Removing the learned router hurts most; fixing the library does not hurt.

Variant	Best gold fitness
HRL (full)	0.809
HRL, fixed library (combinatorial)	0.884
HRL, random router (P=0.5)	0.731
Flat PPO (reference)	0.899

5.2 Qualitative Analysis

Why does HRL perform worse than the baselines at high failure rates, and is the cause informational or structural? I tested the most natural fix: a belief augmented HRL that observes the running gold failure rate (two extra observation dimensions) and is trained directly in the high failure regime. If the problem were that the agent simply could not see how often gold fails, this should rescue it. It does not (Figure 4): the belief augmented variant still degrades sharply as ρ rises (ρ=0.2: 0.755 $\rightarrow$ ρ=0.8: 0.475), tracking the same collapse as the original. I believe the bottleneck is therefore structural, not informational. HRL converged to a static allocation rule and nothing in the training objective forced that rule to be a function of the failure rate. Giving it the signal without a reason to use it changes

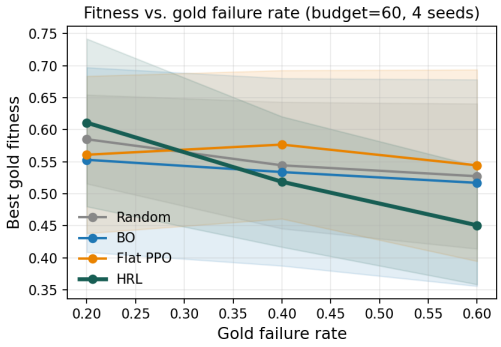

Figure 3: Finding 3. At tight budget (B=60), HRL drops from best (ρ=0.2) to worst (ρ=0.6).

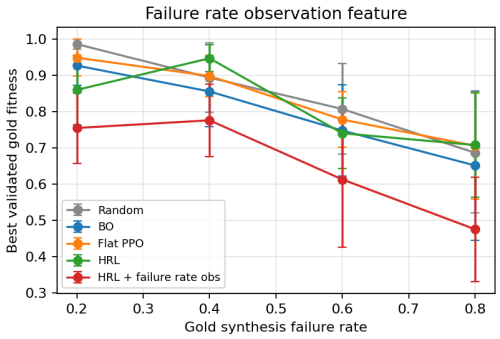

Figure 4: Belief augmented HRL. Observing the gold failure rate does not prevent the high failure collapse.

nothing. This negative result rules out the obvious fix and points squarely at failure aware training as the real next step.

6 Discussion

The HRL per candidate routing advantage is real and dominates when there's a scarce budget and reliable gold assay performance. When budget is abundant, the decision stops mattering and a simpler flat policy that buys gold freely does at least as well. When gold is unreliable, a policy that does not adapt its spending to failure actively hurts. My ablations sharpen this: the gain is attributable to the low level router.

Further analysis showed that exposing the failure rate did not help. That localizes the defect to the training objective. HRL learned a constant allocation because, trained at a single operating point, a constant was optimal in distribution. The static rule is then applied unchanged to operating points where it is badly miscalibrated. My takeaway is that we should spend gold per protein, not per round. It is the right abstraction, but realizing it robustly requires the router to be trained against a distribution of failure rates so that adapting its gold probability to ρ is something the objective actually rewards.

Limitations. (1) Single training distribution: each agent was trained at one (B, ρ) operating point and evaluated across the grid, so much of the degradation could be due to out-of-distribution generalization rather than a fundamental ceiling. (2) Observability is not the bottleneck (shown above), but I did not test the complementary fix of domain randomized failure training, which my analysis predicts should help. (3) Training variance is uncharacterized: I used a single training seed per method, so the 4 seed error bars reflect environment stochasticity, not optimization variance. The overlapping error bars mean the rank orders, while consistent seed by seed, are suggestive rather than conclusive. Lastly, I also use ESM-2 as a stand in oracle rather than wet lab GFP measurements, so absolute fitness values are surrogate.

7 Conclusion

I built CloudLabDP, a budget aware two tier POMDP for protein design campaigns, and a hierarchical agent that routes each candidate to an assay tier individually. Per candidate routing yields a measurable advantage in the tight budget, low failure setting. However, the advantage is brittle: the agent learns a failure blind allocation that makes it the worst performer at high failure, and exposing the failure rate does not fix it. The next step is failure aware training over the full budget$\times$failure grid (domain randomization over ρ), so that the router learns to condition its gold spending on how often gold actually pays off. The right abstraction is to spend gold per protein, not per round. The remaining work is to make that spending adaptive.

8 Team Contributions

This was a solo project. I was responsible for all components: designing and implementing the CloudLabDP POMDP environment, implementing all four agents (Random, BO, flat PPO, and the hierarchical PPO + REINFORCE policy), building the training and evaluation pipeline, and analyzing the results.

Changes from Proposal. My proposal originally scoped a single headline comparison (HRL vs. flat PPO on the FLEXS GFP-TAPE oracle) with the expectation that HRL would dominate. I made two changes. First, I substituted an ESM-2 oracle for the heavier FLEXS GFP-TAPE backend to make the 16 cell sweep x 4 times computationally feasible on Modal GPUs. Second, early results showed HRL did not uniformly dominate, which redirected the project from a does it win toward where and why analysis. This is why I ended with the 16 cell budget$\times$failure grid, an allocation strategy study, router and library ablations, and the belief augmented failure rate experiment. These were all added after the proposal to diagnose the fragility I observed. This pivot from advocating a method to characterizing its regime of validity and its failure mode is the most valuable outcome of the project.

References

[1] Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., & Colwell, L. (2020). Model-based reinforcement learning for biological sequence design. *International Conference on Learning Representations (ICLR)*.

[2] Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F. P., Ekbote, C., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., & Bengio, Y. (2022). Biological sequence design with GFlowNets. *International Conference on Machine Learning (ICML)*.

[3] Kim, H., Kim, M., Yun, T., Choi, S., Bengio, E., Hernández-García, A., & Park, J. (2025). Improved off-policy reinforcement learning in biological sequence design. *International Conference on Machine Learning (ICML)*. arXiv:2410.04461.

[4] Rapp, J. T., Bremer, B. J., & Romero, P. A. (2024). Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nature Chemical Engineering*, 1, 97–107.

[5] Singh, N., Lane, S., Yu, T., Lu, J., Ramos, A., Cui, H., & Zhao, H. (2025). A generalized platform for artificial intelligence-powered autonomous enzyme engineering. *Nature Communications*. `https://doi.org/10.1038/s41467-025-61209-y`

[6] Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., & Kelsic, E. D. (2020). AdaLead: A simple and robust adaptive greedy search algorithm for sequence design. arXiv:2010.02141.

[7] Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., et al. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637), 1123–1130.

[8] Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2), 181–211.

[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

[10] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3), 229–256.

9 Appendix

A Environment and Training Details

Costs and noise. Proxy tier: cost \$1, observation noise σ=0.30, failure rate 0. Gold tier: cost \$10, σ=0.02, failure rate ρ (default 0.2). Up to T=12 rounds, maximum batch size 16, default

budget B=400 (swept to $\{60, 120, 240, 480\}$). Libraries: single site (one mutation per candidate), combinatorial (k=2 random mutations), and ML designed (UCB over a radius 3 neighborhood of the belief model).

Hyperparameters. Flat PPO: 200k steps, n_{steps}=1024, batch size 128, lr 3×10^{-4}, γ=0.995, λ_{GAE}=0.95, 4 parallel envs. HRL high level: 300k steps, n_{steps}=512, batch size 64, lr 3×10^{-4}, γ=0.995. HRL low level: REINFORCE, lr 1×10^{-3}, cost penalty λ=0.005. Belief ensemble: 5 CNN members (1D conv, kernel 5), refit each round for 15 epochs with inverse variance weighting, buffer capped at 4096. BO: 4 member committee, UCB β=2.0, fixed combinatorial library, gold tier only, batch size 8. The compute capped overnight HRL configuration uses 50k steps, 2 belief members, and refit every 16 steps to fit A10G GPU time budgets.

B Additional Results

The joint budget×failure sweep (Figure 3) crosses $B \in \{60, 120, 240\}$ with $\rho \in \{0.2, 0.4, 0.6\}$. The fragility is sharpest at B=60 (HRL $0.611 \to 0.519 \to 0.451$ as ρ rises) and softens with budget (B=240: $0.809 \to 0.728 \to 0.665$), consistent with the interpretation that failed gold syntheses hurt most when budget is scarce. Fitness per cost is dominated by budget level rather than agent identity (all agents ≈ 0.010 at B=60 decaying to ≈ 0.002 at B=480), reflecting diminishing returns as the fitness ceiling is approached.