Protein Engineering as Stochastic, Adversarial Bandit Problem with Domain Model Likelihoods

Richard Michael
Department of Computer Science
University of Copenhagen
richard.michael@di.ku.dk

Abstract

The biochemical discovery and engineering process is an online learning problem and can be viewed as a multi-armed bandit problem. We show the initial discrepancy between the naive treatment of protein optimization in this framework and how to constrain it to obtain a feasible action-space. This brings learning-theoretical results to the optimization in biochemical engineering and allows us to apply bandit algorithms in this domain. Lastly, we propose a modification of a state of the art best-of-both-worlds algorithm to be used to optimize a protein engineering task and provide empirical results accounting for delays of observations.

1 Introduction

Protein engineering is a noisy optimization problem starting from one or multiple reference sequences with a particular *fitness* trait and iteratively modifying the sequence of amino acids to improve that trait. The design decisions come with little guarantees that the modifications at each step ultimately yield the improvement we are looking for. The potential design space is vast and the number of functional variants of a protein is vanishingly small (1). Furthermore, the measurements taken in the wet-lab come with experimental noise from the type of assay used to measure the properties of protein sequences (2; 3; 4). In contrast to in-silico measurements through computational oracles, which are proxies for downstream tasks of interest, ie. Rosetta for stability (5) or AlphaFold 3 for structure and interaction prediction (6).

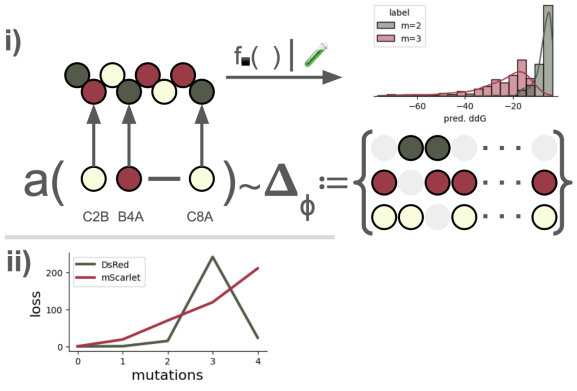


Figure 1: Overview for protein sequence optimization. i) Given a reference sequence of amino acids we determine the mutation to make (actions a), which we can sample from a prior model (ϕ). We measure the function values either computationally ($f(\cdot)$) or in-vitro. ii) Losses (unconstrained) can vary based on the system measured (ie. two different RFP proteins), the noise level of the experiments, and the relative distance to the starting point ie. number of mutations added.

Recent advances in *domain models* for proteins such as protein language models (PLMs) (7; 8; 9; 10) and datasets for benchmarking (11; 12) promise applicable models to optimize protein sequences and selection criteria for models given some tasks of interest. The moment we decide to run a campaign

Preprint. Under review.

where decisions are based on previous iterations and the results are used to fit a model, we have left the *offline* learning domain and have to solve an *online* learning problem. Current active learning optimization approaches (13; 1) then propose a design space of k possible mutants per iteration, at which point we have to consider the trade-off between potentially collecting more data or exploiting already observed effects of the mutants. Thus, the experimentalist may be inclined to ask: "How many experiments should we run (and how many replicates) until we know that the campaign has been successful?"
In contrast to applied machine learning stand recent advances in learning-theoretic online learning algorithms yielding optimality in stochastic, adversarial, and corrupted settings with or without delay of observations (14; 15; 16). The applied researcher concerned with optimizing biochemical sequences may be inclined to ask if those algorithms warrant application and if they are suitable for empirical assessment. Simply applying learning-theoretical algorithms to optimization problems can give us worst case guarantees and bounds to estimate performances, but naively applying them to tasks like biochemical engineering campaigns can give problems such as: vacuous bounds, too large search spaces, ill-defined budgets. Furthermore if the returns of actions are delayed (ie. running experiments) bandit algorithms may be difficult to apply. To address some of these issues we present the following results:

1. we examine which learning-theoretical framework to select given the considerations of a protein engineering campaign,
2. we show how to effectively constrain the problem to be able to treat it as a bandit problem using prior domain models,
3. we construct an in-silico multi-fidelity delayed feedback experiment mirroring a protein engineering campaign from a set of established computational oracles,
4. we modify and apply an existing state of the art learning-theoretic algorithm, to select design actions based on computed relative likelihoods respective a starting sequence.

1.1 Related Work

Using machine learning models in a protein engineering campaigns has been studied by (3; 17; 18; 19; 20; 13) amongst others. (21; 1) present protein engineering as an online learning problem, the latter testing empirically on a combinatorically complete enzyme optimization task with multiple objectives. While sampling protein sequences for design has been investigated by (22) and de novo modeling by (23) with (24) investigating protein language models.
To the best of our knowledge none of these previous investigations account for learning-theoretic optimal algorithms (as presented in (25)) or explore the divergence between theoretical guarantees and empirical observations.

2 Results

We present how protein engineering relates to a multi-armed bandit problem (Section 2.1), why initial bounds can become vacuous (Section 2.2) and how to resolve this with prior information to constrain our action space, structure the evaluation budget and delayed returns such that we can apply this algorithm class (Section 2.3); leading us to a practical FTRL algorithm (Section 2.4).

2.1 Proteins are multi-armed bandits in stochastic, adversarial environments

When modifying a protein sequence we can only observe a limited number of candidates. During an engineering campaign let a mutation at one or multiple sites of the sequence be the action (later I_t), followed by a measurement either in-vitro (eg. a screening assay) or in-silico (eg. a biophysical simulator) - see Fig. 1. As *we do not have perfect information* available and only observe a limited number from the set of available actions, this can be viewed a multi-armed bandit setting; given that some prior assumptions are met (25). The arms or bandits in this case are the protein designs to make, the observations of which can be stochastic, adversarial or both. The moment we have selected a design for measurement a loss comes with it, which may take some time to actually observe.[1]

[1]For simplicity we consider the loss the squared residual to an optimal design, however other losses can be considered.

Stochastic environments Laboratory measures are inherently stochastic, for example deep mutational scan (DMS) experiments are noisy in their return values. A different set of uncertainties, come with computational oracles which is either a non-deterministic oracle or the (biophysical) simulator is only approximately correct for a downstream task.

Adversarial environments There exist different cases of adversarial losses during a protein optimization campaign. First, the more we mutate a protein sequences, the more the later changes are affected by previous trials. When we apply too many modifications we further have to account for the interaction of the proposed changes and the existing configuration. For example introducing all single variants can be argued to be sufficiently close to the wild-type for a faithful label distribution. However mutating multiple amino acids, especially in functional sites of a protein can be argued to completely change the label distribution (ie. fitness values). Secondly, during a campaign a protein may be optimized respective different traits, such as thermal stability and yield (1). During the individual rounds one trait may be preferred over the other, such that the losses that are computed change between trials, appear adversarial to us.

Delayed Feedback During an engineering campaign, not all data may be observed at once. A common setup includes running computational simulations or fitting surrogate models to a subset of observations (21; 13; 1). An experimental batch is assembled and the true function values only become available at the next iteration, or even later if the experiments are time-consuming. This poses the question how large a maximal delay is allowed to be and what proportion of our designs require labels before entering the next round of experiments; finding a trade-off between rapid low-fidelity observations and delayed high-fidelity observations.

Best of both worlds Given noisy measurements for our true function values and given potentially changing losses, during optimization, an optimal algorithm has to address both worlds. In the learning literature this is known as a best-of-both-worlds algorithm (BOBW) that addresses both stochasticity of the observations and the adversarial component (25).
The selection of the algorithm is always environment dependent. If there is no adversarial component an algorithm for stochastic environments can be used such as (regularized) Follow-The-Leader (FTRL). If there is no stochasticity but change of domains or target properties over time such that losses are adversarial we may choose for example an Exp3 algorithm (see (25) for both). One state of the art example that does well across both environments is the BOBW FTRL algorithm in (14), which we investigate in Section 2.4.

2.2 Naive combinatorial actions yield vacuous bounds

A protein sequence of length L can take $|AA| - 1$ alternative configurations per site, where AA is the set of natural occurring amino acids. Naturally, for m (multiple) mutants this becomes $\binom{L}{m} \times (|AA| - 1)^m$, which is $> 10^9$ possible actions for triple mutants if $L \geq 150$ – a prohibitively large space from which to choose one design (action I_t) (see appendix Fig. 1). The worst-case bounds in this case are vacuous.

2.3 Constraining the action-space

Not all proposals are sensible and some may be infeasible or highly unlikely. Given that a domain model exists that has learned the occurrence of valid amino acid sequences (ie. `esm-2`, `esm-3`, etc.) we propose to leverage such models as prior knowledge to limit our action space sufficiently yielding tighter bounds and improved learning behavior.

Proposition: Sample a fixed number of actions from a probability-simplex derived from prior domain model (parameterized by ϕ) likelihoods respective the Δ to a reference.

Specifically we propose a probability simplex over the set of amino acids AA of the protein sequence(s) of length L to optimize.[2]

$$\mathbb{P}_\phi := \left\{ \mathbf{p} \in [0,1]^{L \times AA} | \mathbf{p} \geq 0 \wedge \forall l \in L : \sum_{aa}^{AA} \mathbf{p}_{l,aa} = 1 \right\}. \tag{1}$$

We note that this can be biased to sampling the wild-type (WT) or the most likely sequence under that model – preserved as the fittest sequence. To select an action at a likely position we therefore propose the action simplex as the difference to the WT.: Let $X_{(WT)}$ be a wild-type protein sequence:

$$\Delta_\phi^{(\overline{\text{WT}})} := \mathbb{P}_\phi - \mathbb{P}_\phi \mathbb{1}[X = X_{(\text{WT})}] \qquad \mathbb{P}\Delta_\phi^{(\overline{\text{WT}})} := \forall l \in L : \frac{\Delta_{\phi,l}^{(\overline{\text{WT}})}}{\sum_{aa \in AA} \Delta_{\phi,l,aa}^{(\overline{\text{WT}})}} \tag{2}$$

This is the difference between the likelihoods of all available amino acids and the wildtype probabilities, where the indicator $\mathbb{1}[X = X_{(\text{WT})}]$ defines a mask giving us the WT residues in its position - requiring us to renormalized to obtain valid probabilities. We can now use the simplex $\mathbb{P}\Delta_\phi^{(\overline{\text{WT}})}$ to sample actions of interest and apply our online learning algorithm.

2.4 A prior constrained FTRL algorithm

The proposed algorithm follows Alg.1 in (14) closely. We set K to be the maximum number of possible designs. Note that the larger the delays (size D_t) and also high noise levels result in detrimental guarantees (see Supplementary Fig. 3).

Algorithm 1 Practical **prior FTRL** with Delayed Bandit Tuning ; contributions are in red.

Initialize: $D_0 = 0$, budget n, K, prior ϕ, action-simplex $\mathbb{P}\Delta_\phi^{(\overline{\text{WT}})}$, $L_1^{\text{obs}} = \mathbf{0}_K$
$\{\kappa\}_{1..K} \sim \mathbb{P}\Delta_\phi^{(\overline{\text{WT}})}$ ▷ sample action-space from prior odds
for $t = 1, \ldots, n$ **do**
  set $\sigma_t = \sum_{s=1}^{t-1} \mathbb{1}(s + d_s > t)$ ▷ remaining observations
  update $D_t = D_{t-1} + \sigma_t$
  set $x_t = \arg\min_{x \in \Delta^\kappa} \langle L_t^{\text{obs}}, x \rangle + F_t(x)$ ▷ optimal distribution over arms, see Eq. (3)
  sample $I_t \sim x_t$
  observe (s, l_{s,I_s}) for all s that satisfy $s + d_s = t$
  $\hat{L}_{t+1}^{\text{obs}} = \hat{L}_t^{\text{obs}} + \sum_{s=1}^{t} \hat{l}_s \mathbb{1}(s + d_s = t)$ ▷ importance weighted losses, see Eq. (4)
end for

We further use this algorithm with no delays using a low-fidelity proxy function to assess, the change in regret and decisions given a lower quality, rapid screening proxy. The same algorithm can be used to decide between which domain model to query by introducing an outer loop over the available models: $\{\phi_{\texttt{esm1b}}, \phi_{\texttt{protT5}}, ...\}$ and sampling a set of actions from each of the available models to determine the lowest regret.

2.5 Resolving simplex optimization by stacking bandits

One of the key challenges in Algorithm 1 is the simplex optimization with the Tsallis-Inf regularization: $\arg\min_{x \in \Delta^\kappa} \langle \hat{L}_t^{\text{obs}}, x \rangle + F_t(x)$. The larger the action-space Δ^κ, the harder it becomes to optimize a probability distribution over it, with an optimizer - especially for hundreds to thousand of proposed designs. We sketch how to resolve this issue in Fig. 2. One possible approach is to decompose the actions into decisions about positions and residues at that positions, implying the action (ie. the bandit) is not a particular design anymore, but a design decision instead. First, select a position, secondly select a candidate for that position. While we are now faced with optimizing multiple simplices, these are significantly lower-dimensional and reduce the design space. This

[2]For brevity $\mathbf{p}_{l,aa}$ is the probability vector from the probability matrix $\mathbf{p} \in [0,1]^{L,AA}$ (the sequence length by number of amino-acid matrix) such that $\mathbf{p}_{l,aa} = \mathbf{p}[L = l, AA = aa]$ see Appendix A.5 for a detailed description.

Setting (delay)	Regret±SE (↓)
prior FTRL (one)	**95.32** ± 0.91
prior FTRL (zero-noisy)	95.52 ± 0.59
prior FTRL (10-zero)	96.93 ± 0.42
Sampling prior	95.88 ± 0.85
Sampling uniform-random	96.06 ± 0.46

Table 1: RFP optimization. We assess pseudo Regret (across 5 seeds) for $d = 1$ delays, noisy instant observations, and observations in batches of 10 (decreasing delays). Baseline values are sampling uniformly from the available designs (sampled with `esm-2` likelihoods) and completely at random (last).

proposed change can treat the sampling of actions I_t independently or we consider them as a Markov decision process with dependence on all previous decisions in the hierarchy. Regardless, it requires us to propagate importance weighted losses during the optimization process, such that we obtain losses for each decision in the hierarchy. The analysis for this proposal and its realization is out of the scope of this paper and is left for future work.

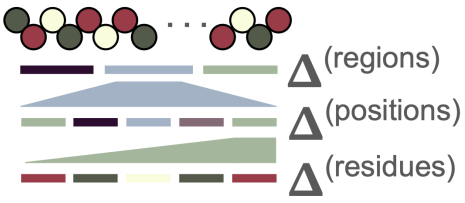


Figure 2: Overview for stacking decisions. Depending on the coarseness of the discretization we can decide by: i) region (ie. parts of the protein (active site or not)), ii) positions in a selected region, iii) amino acids for that position.

2.6 Optimizing RFP stability with delays and stochasticity

We optimize the protein DsRed protein (RFP, PDB 2VAD) by its stability using the RaSP oracle as proxy for stability $\widehat{\text{REU}} \propto \Delta\Delta\text{G}$ (26).[3] We combine the oracle with an RMF functional landscape (27) implying noisy stability values and change in function the more mutations are proposed. An oracle function with higher noise serves as a rapid-screening proxy to simulate fast lower fidelity experiments. Losses are computed as squared residuals against the optimal $\Delta\Delta\text{G} \times |\text{RMF}|$ value from the available designs. We are given a finite budget of 100 iterations to decide among $K = 50$ variants with `esm2` (650M) as our domain model. Our goal is to learn the optimal design which corresponds to accumulating the lowest regret (see Eq. (5)) – see Table 1. As baseline we sample uniformly from all available mutations generated by the prior model or completely at random. We compare observations without delay, and a batch-delay setting and test what regret and strategy the noisy rapid observations give us.

3 Discussion

Valid distribution objects In this work we have used PLMs due to their availability and ease of use, we note however that any model, which yields a valid probability matrix for protein sequences may be used, for example: HMMs, VAEs, and others (28; 29; 30).

Extensions Our work presents an introductory example of how to apply results from online learning algorithm research to biochemical sequence optimizations. Protein engineering in practice should consider batches of observations under different fidelities, and delays, while accounting for different objectives. To that end, we can consider alleviating the greedy non-myopic approach and instead should consider batches of candidates per iteration. This requires delayed feedback batched multi-armed bandits, for which results are an ongoing endeavour.

4 Conclusion

We have demonstrated how formulate a protein engineering task as a bandit problem via constraint action spaces and how to apply a recent best-of-both world algorithm.

[3]We assume additive effects of mutations. This implies that the change in function value for a multi-mutant is the sum of the individual mutation function values.

Acknowledgments and Disclosure of Funding

RM is funded by the Danish Data Science Academy, which is funded by the Novo Nordisk Foundation (NNF21SA0069429) and VILLUM FONDEN (40516). Further funding includes the Pioneer Centre for AI (DRNF grant number P1).

References

[1] J. Yang, R. G. Lal, J. C. Bowden, R. Astudillo, M. A. Hameedi, S. Kaur, M. Hill, Y. Yue, and F. H. Arnold, "Active Learning-Assisted Directed Evolution," July 2024.

[2] K. S. Sarkisyan, D. A. Bolotin, M. V. Meer, D. R. Usmanova, A. S. Mishin, G. V. Sharonov, D. N. Ivankov, N. G. Bozhanova, M. S. Baranov, O. Soylemez, N. S. Bogatyreva, P. K. Vlasov, E. S. Egorov, M. D. Logacheva, A. S. Kondrashov, D. M. Chudakov, E. V. Putintseva, I. Z. Mamedov, D. S. Tawfik, K. A. Lukyanov, and F. A. Kondrashov, "Local fitness landscape of the green fluorescent protein," *Nature*, vol. 533, pp. 397–401, May 2016. Number: 7603 Publisher: Nature Publishing Group.

[3] P. A. Romero, A. Krause, and F. H. Arnold, "Navigating the protein fitness landscape with Gaussian processes," *Proceedings of the National Academy of Sciences*, vol. 110, pp. E193–E201, Jan. 2013. Publisher: Proceedings of the National Academy of Sciences.

[4] K. E. Johnston, P. J. Almhjell, E. J. Watkins-Dulaney, G. Liu, N. J. Porter, J. Yang, and F. H. Arnold, "A combinatorially complete epistatic fitness landscape in an enzyme active site," June 2024.

[5] K. T. Simons, R. Bonneau, I. Ruczinski, and D. Baker, "Ab initio protein structure prediction of CASP III targets using ROSETTA," *Proteins: Structure, Function, and Bioinformatics*, vol. 37, no. S3, pp. 171–176, 1999. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-0134%281999%2937%3A3%2B%3C171%3A%3AAID-PROT21%3E3.0.CO%3B2-Z.

[6] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, and J. M. Jumper, "Accurate structure prediction of biomolecular interactions with AlphaFold 3," *Nature*, pp. 1–3, May 2024. Publisher: Nature Publishing Group.

[7] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song, "Evaluating Protein Transfer Learning with TAPE," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[8] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost, "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 7112–7127, Oct. 2022. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[9] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives, "Evolutionary-scale prediction of atomic level protein structure with a language model," preprint, Synthetic Biology, July 2022.

[10] T. Hayes, R. Rao, H. Akin, N. J. Sofroniew, D. Oktay, Z. Lin, R. Verkuil, V. Q. Tran, J. Deaton, M. Wiggert, R. Badkundri, I. Shafkat, J. Gong, A. Derry, R. S. Molina, N. Thomas, Y. A. Khan, C. Mishra, C. Kim, L. J. Bartie, M. Nemeth, P. D. Hsu, T. Sercu, S. Candido, and A. Rives, "Simulating 500 million years of evolution with a language model," July 2024.

[11] M. Xu, Z. Zhang, J. Lu, Z. Zhu, Y. Zhang, M. Chang, R. Liu, and J. Tang, "PEER: A Comprehensive and Multi-Task Benchmark for Protein Sequence Understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35156–35173, Dec. 2022.

[12] P. Notin, A. Kollasch, D. Ritter, L. Van Niekerk, S. Paul, H. Spinner, N. Rollins, A. Shaw, R. Orenbuch, R. Weitzman, *et al.*, "Proteingym: Large-scale benchmarks for protein fitness prediction and design," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[13] L. Chen, Z. Zhang, Z. Li, R. Li, R. Huo, L. Chen, D. Wang, X. Luo, K. Chen, C. Liao, and M. Zheng, "Learning protein fitness landscapes with deep mutational scanning data from multiple sources," *Cell Systems*, vol. 14, pp. 706–721.e5, Aug. 2023.

[14] S. Masoudian, J. Zimmert, and Y. Seldin, "A best-of-both-worlds algorithm for bandits with delayed feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 11752–11762, 2022.

[15] J. Zimmert and Y. Seldin, "An Optimal Algorithm for Stochastic and Adversarial Bandits," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pp. 467–475, PMLR, Apr. 2019. ISSN: 2640-3498.

[16] J. Zimmert and Y. Seldin, "Tsallis-INF: An Optimal Algorithm for Stochastic and Adversarial Bandits," *Journal of Machine Learning Research*, vol. 22, no. 28, pp. 1–49, 2021.

[17] K. K. Yang, Z. Wu, and F. H. Arnold, "Machine-learning-guided directed evolution for protein engineering," *Nature Methods*, vol. 16, pp. 687–694, Aug. 2019. Publisher: Nature Publishing Group.

[18] S. Mazurenko, Z. Prokop, and J. Damborsky, "Machine learning in enzyme engineering," *ACS Catalysis*, vol. 10, no. 2, pp. 1210–1223, 2019.

[19] B. L. Hie and K. K. Yang, "Adaptive machine learning for protein engineering," *Current opinion in structural biology*, vol. 72, pp. 145–152, 2022.

[20] C. R. Freschlin, S. A. Fahlberg, and P. A. Romero, "Machine learning to navigate fitness landscapes for protein engineering," *Current opinion in biotechnology*, vol. 75, p. 102713, 2022.

[21] S. Stanton, W. Maddox, N. Gruver, P. Maffettone, E. Delaney, P. Greenside, and A. G. Wilson, "Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders," Tech. Rep. arXiv:2203.12742, arXiv, July 2022. arXiv:2203.12742 [cs, q-bio, stat] type: article.

[22] J. T. Darmawan, Y. Gal, and P. Notin, "Sampling Protein Language Models for Functional Protein Design," Oct. 2023.

[23] P.-S. Huang, S. E. Boyken, and D. Baker, "The coming of age of de novo protein design," *Nature*, vol. 537, no. 7620, pp. 320–327, 2016.

[24] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos, C. Xiong, Z. Z. Sun, R. Socher, *et al.*, "Large language models generate functional protein sequences across diverse families," *Nature Biotechnology*, vol. 41, no. 8, pp. 1099–1106, 2023.

[25] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 1 ed., July 2020.

[26] L. M. Blaabjerg, M. M. Kassem, L. L. Good, N. Jonsson, M. Cagiada, K. E. Johansson, W. Boomsma, A. Stein, and K. Lindorff-Larsen, "Rapid protein stability prediction using deep learning representations," *eLife*, vol. 12, p. e82593, May 2023. Publisher: eLife Sciences Publications, Ltd.

[27] J. Neidhart, I. G. Szendro, and J. Krug, "Adaptation in tunably rugged fitness landscapes: the rough mount fuji model," *Genetics*, vol. 198, no. 2, pp. 699–721, 2014.

[28] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1 ed., Apr. 1998.

[29] J. Frazer, P. Notin, M. Dias, A. Gomez, J. K. Min, K. Brock, Y. Gal, and D. S. Marks, "Disease variant prediction with deep generative models of evolutionary data," *Nature*, vol. 599, pp. 91–95, Nov. 2021. Number: 7883 Publisher: Nature Publishing Group.

[30] P. M. Notin, L. Van Niekerk, A. W. Kollasch, D. Ritter, Y. Gal, and D. Marks, "TranceptEVE: Combining Family-specific and Family-agnostic Models of Protein Sequences for Improved Fitness Prediction," preprint, Genetics, Dec. 2022.

[31] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proceedings of the National Academy of Sciences*, vol. 118, p. e2016239118, Apr. 2021. Publisher: Proceedings of the National Academy of Sciences.

A Supplementary Material

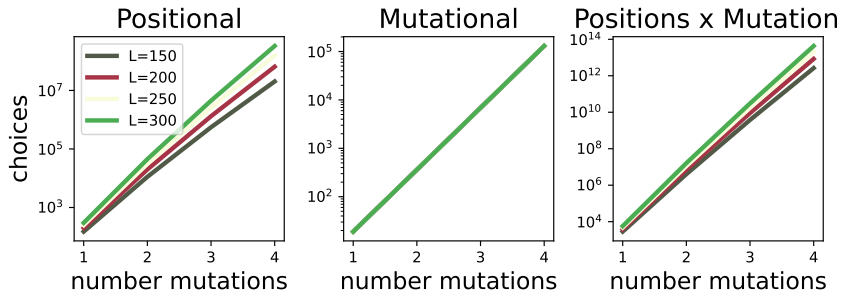
A.1 Availability

An implementation of Algorithm 1, testing-suite and black-box library is available under the MIT-license and will be made available upon acceptance.

A.2 Implementation and Availability

We use the protein oracles (RaSP, RMF) from [redacted][4] (available under MIT license) as black-box functions to optimize. We use `esm2` under MIT License. All code to obtain all results and figures will be made available upon acceptance.

A.3 Combinatorial Space



Supplementary Figure 1: Number of available choices available during protein engineering given sequences of different lengths (L) given number of mutations to make (x-axis) for positions (left), mutations (regardless of position) (middle), and mutations at positions (right).

A.4 Algorithm Details

We use the Tsallis entropy regularizer exactly as defined in Eq. 1 of (14):

$$F_t(x) := -2\eta_t^{-1}\left(\sum_{i\in K}\sqrt{x_i}\right) + \gamma_t^{-1}\left(\sum_{i\in K} x_i(\log x_i - 1)\right) \tag{3}$$

which in practice this requires a small epsilon term such that the RHS $\log$ term can be calculated. The losses are importance weighted losses as proposed in (14):

$$\hat{L}_t^{obs} = \sum_{s=1}^{t-1}\hat{l}_s \mathbb{1}(s + d_s < t) \qquad \hat{l}_{t,i} = \frac{l_{t,i}\mathbb{1}(I_t = i)}{x_{t,i}} \tag{4}$$

[4]Library name censored to preserve anonymity of the authors and will be made available upon acceptance.

Empirical evaluation includes the pseudo-regret, which is defined as the expected cumulative sum of the loss occurred against the best loss in hind-sight.

$$\bar{\text{Reg}}_T = \mathbb{E}\left[\sum_{t=1}^{T}(l_{t,I_t} - l_{t,i_T^*})\right] \tag{5}$$

Losses are bounded square losses between the best possible observation $y^* \in \mathbb{R}$ and the prediction $\hat{y} \in \mathbb{R}$ ensuring that $l \in [0,1]$, the slope is determined by constant $c \in \mathbb{R}_+$:

$$l := \frac{(y^* - \hat{y})^2}{(y^* - \hat{y})^2 + c} \tag{6}$$

.

A.5 Exact likelihood computations

To obtain likelihood estimates for protein sequences via `esm-2` we follow the `wt-marginal` approach described in the appendix of (31). Specifically we compute for all positions the masked likelihoods from a sequence and computing the last-layer softmax on the logits of a forward pass. Given a sequence X and with masking at position i as $X_{\neg i}$, the logits of the last-layer as z_i given AA tokens (ie. amino acids), for all positions i:

$$P(X_i|X_{\neg i}) = \frac{\exp(z_i)}{\sum_{a \in AA}\exp(z_a)} \tag{7}$$

A.6 FTL in stochastic environments

One fundamental result is that FTL in stochastic environments achieves:

$$R_T = \mathcal{O}\left(\min\left\{\frac{\log N}{\Delta \min}, \sqrt{T \log N}\right\}\right) \tag{8}$$

given N models to choose from, and stochastic observations from a distribution s.t. loss $l_{t,i}$ is independent for all $t \in T$ and with $\Delta_i := \mathbb{E}[l_{ti}] - \mathbb{E}[l_{ti^*}]$ the minimal delta to the optimal $\Delta_{\min} = \min_{i \in N\{i^*\}} \Delta_i$.

This is linear in adversarial environments and not optimal (25).

A.7 Hedging on models

How to achieve better than linear regret in adversarial environments? Learn a weighting across the selection of algorithms. At each iteration, after observing the losses update the weights such that $w_{t+1,i} = w_{ti} + exp(-\eta l_{ti})$

A.8 Hedging with entropy regularization

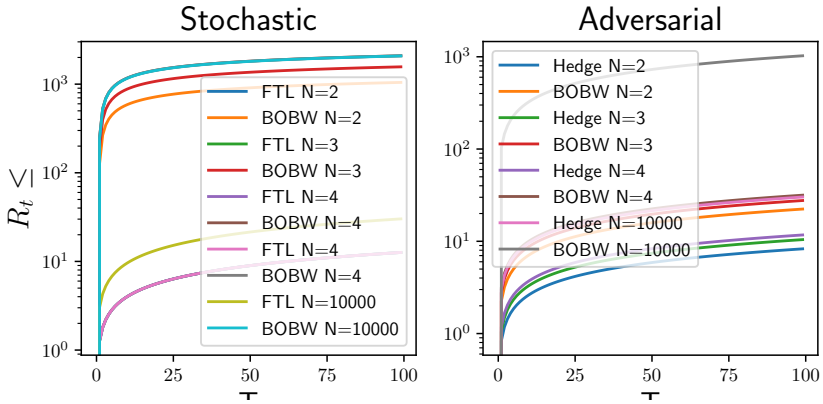
Given Shannon entropy H and the probability simplex over K actions Δ^K

$$p_t \in \arg\min_{p \in \Delta}\left\{\eta \sum_{s=1}^{t-1}\langle l_s, p\rangle - H(p)\right\} \tag{9}$$

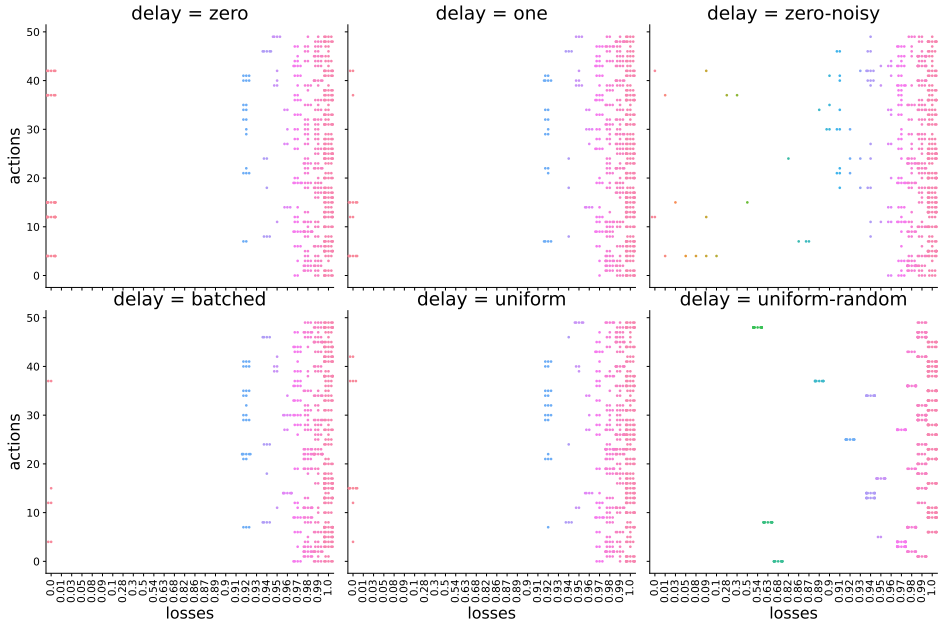
decomposes into a stability term (LHS) and regularizing penalty term (RHS) - see Ch. 28 in (cf. Chapter 28 Alg. 16 in (25)).

A.9 Visualization regret bounds

A.10 Additional result figures



Supplementary Figure 2: Visualization of pseudo-regret bounds in the stochastic and adversarial setting, given N actions available over 100 iterations, this visualizes the worst-case upper bound.



Supplementary Figure 3: Losses observed given actions taken. We note that the majority of losses are between $[0.7, 0.9]$ indicating that a better bounding function can be found for this particular problem. Note that the actions are shuffled between the seeded runs, which implies that the actions with the lowest loss are the same design with a different index per seed.