Tree Search-Based Evolutionary Bandits for Protein Sequence Optimization

Jiahao Qiu
Princeton University
jq3984@princeton.edu

Hui Yuan
Princeton University
huiyuan@princeton.edu

Jinghong Zhang
University of California San Diego
zhangjinghong99@outlook.com

Wentao Chen
MLAB Biosciences Inc
wentao.chen@mlabbiosciences.com

Huazheng Wang
Oregon State University
huazheng.wang@oregonstate.edu

Mengdi Wang
Princeton University
mengdiw@princeton.edu

Abstract

While modern biotechnologies allow synthesizing new proteins and function measurements at scale, efficiently exploring a protein sequence space and engineering it remains a daunting task due to the vast sequence space of any given protein. Protein engineering is typically conducted through an iterative process of adding mutations to the wild-type or lead sequences, recombination of mutations, and running new rounds of screening. To enhance the efficiency of such a process, we propose a tree search-based bandit learning method, which expands a tree starting from the initial sequence with the guidance of a bandit machine learning model. Under simplified assumptions and a Gaussian Process prior, we provide theoretical analysis and a Bayesian regret bound, demonstrating that the combination of local search and bandit learning method can efficiently discover a near-optimal design. The full algorithm is compatible with a suite of randomized tree search heuristics, machine learning models, pre-trained embeddings, and bandit techniques. We test various instances of the algorithm across benchmark protein datasets using simulated screens. Experiment results demonstrate that the algorithm is both sample-efficient and able to find top designs using reasonably small mutation counts [1].

1 Introduction

Advances in biotechnology have demonstrated human's unprecedented capabilities to engineer proteins. They make it possible to directly design the amino acid sequences that encode proteins for desired functions, towards improving biochemical or enzymatic properties such as stability, binding affinity, or catalytic activity. Directed evolution (DE), for example, is a method for exploring new protein designs with properties of interest and maximal utility, by mimicking the natural evolution process. The development of DE was honored in 2018 with the awarding of the Nobel Prize in Chemistry to Frances Arnold for the directed evolution of enzymes, and George Smith and Gregory Winter for the development of phage display [3, 39, 46]. Traditional DE strategies are inherently screening (greedy search) strategies with limited ability to generate high-quality data for probing the

[1] Our code is available at https://anonymous.4open.science/r/TreeSearchDE-public-0D26/

The NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development (AI4D3 2023), New Orleans, LA, USA, 2023.

full sequence-function relationships. Recent advances in synthetic DNA generation and recombinant protein production make the measurement of protein sequence-function relationships reasonably scalable and high-throughput [28, 49].

Due to the bottleneck of wet-lab experimentation and the complex landscape of protein functions, identifying novel protein designs for maximal fitness remains one of the most difficult but high-value problems in modern medicine and biology. This has motivated scientists to apply machine learning approaches, beginning with Fox et al. [16] and followed by many, with increasing amounts of efforts utilizing *in silico* exploration and machine learning beyond experimental approaches [49, 15, 12, 36, 18, 44, 38]. More recent advances in large language models open up new opportunities for modeling and predicting protein functions and generalizing knowledge across protein domains [32, 37, 27, 21, 13].

The key research challenge with designing the iterative protein screening strategy is *exploration*, i.e., how to effectively explore in a large combinatorial space and learn the sequence-to-function landscape towards finding the optimal. While many attempts have been proved successful in simulation and sometimes in real experiments [8, 36], they often are limited by practical constraints and their performance is very sensitive to domain/distribution shifts. Even with the best and largest pre-trained protein language models such as ESM-1b [31] and ProGen2 [27], one often needs to explore an almost unknown domain and learn a new function map in order to discover new drugs. This is especially true with antibody engineering. Antibodies have highly diverse complementarity-determining region (CDR) sequences that can be altered, resulting in a huge sequence space to explore for optimal properties. The binding of antibodies to their targets are extrinsic properties of antibodies and it is difficult to accurately model the sequence-binding relationships solely from the sequences alone. Further, most of the exploration strategies used in practice lack theoretical guarantees.

Practical considerations in protein screens and a tree search view Protein engineering is typically done through trial-and-error approaches in altering the primary sequence by mutations or changing the length of certain regions. More high-throughput approaches (e.g. *in vitro* display systems) involve randomizing the amino acids for the positions-of-interest or region-of-interest. To obtain the optimal properties for a protein, the directed evolution approach is typically used by exploring a limited sequence space in each round of engineering, and starting the next round of engineering from the best one or the best few sequences in the previous round, in an iterative manner [47]. A typical protein engineering workflow by producing purified proteins is limited to up to a few hundred sequences due to the throughput limitations. *In vitro* display systems, on the other hand, can be used to screen millions of sequences, although the data generation (labeled data of sequence-function relationships) throughput is much smaller.

For practical reasons, especially for therapeutic proteins, the choice of mutations is dependent on the knowhows of the protein engineer, and the number of mutations is kept low in order to prevent unexpected issues associated with decreased protein stability, compromised binding specificity, and immunogenicity. Adding too many mutations may also lead to distribution-shift and reduce the robustness of machine-learning models. Thus, practitioners are reluctant to make large jumps in the screening/search process.

For example, [8] studied the engineering of Adeno-associated virus 2 capsid protein (AAV) and screened a total of 201,426 variants of the AAV2 wild-type (WT) sequence. It screened all single mutations in the first round, then generated variants with > 1 mutations via randomization and selection of high-value ones in later rounds. Such an iterative process mimics a tree search. To understand this process, we visualize those sequences from the dataset of [8] after downsampling in Figure 1. We see that the screen data map nicely to a tree, where the root node corresponds to the wide-type AAV and mutations connect parent and child nodes. These observations are consistent with practical screening strategies that add mutation sequentially and search for better alternatives. Note that the tree size grows exponentially as mutations are added. For the example of AAV, variants with up to 5 mutations form a tree with $\sum_{i=0}^{5} \binom{28}{i} \cdot 19^i$ nodes. Thus even with a bounded number of mutations, the problem is prohibitively difficult.

Our Approach In order to make the screening process more efficient, we borrow ideas from both the protein engineering practices and recent advances in bandit machine learning, hoping to get the best from both worlds. We follow two principles for algorithm design:

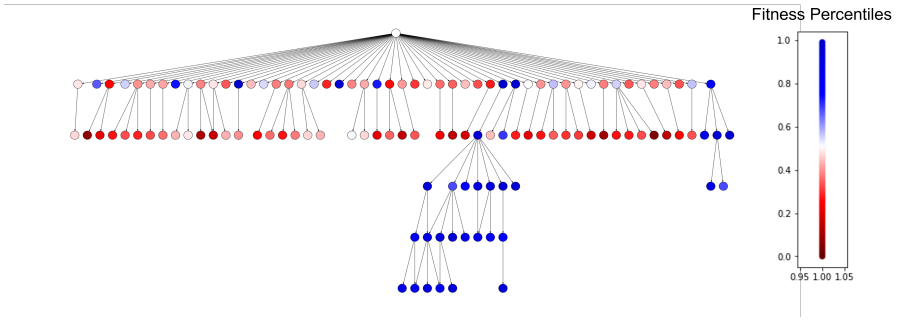


Figure 1: A Tree visualization of AAV screen dataset [8], generated by starting from the wild-type and building the tree via downsampling children with an editing distance of 1 from the parent. The wet-lab screen initiates with a wild-type design sequence (root node), and in each round new sequences are generated by adding randomization and keeping those with high fitness scores as parents. It was believed that nodes with high fitness are more likely to generate high-fitness children.

(1) We wish to largely follow a tree search process and identify optimal sequences with just a few mutations. As mentioned, practitioners are reluctant to make large jumps in the screening/search process. Being a *local search* strategy, tree search from lead sequences will keep total mutation counts small, which means better reliability of the found solution. Further, machine learning models for function prediction are more likely to generalize well to new designs that do not change too much from training data.

(2) We employ bandit exploration techniques to guide the tree branching process. Instead of searching greedily using a learned prediction model, we hope to more aggressively search designs with higher uncertainty. Two main techniques in bandit learning are upper confidence bound (UCB) and posterior sampling (also known as Thompson Sampling, aka TS). We will incorporate these bandit techniques, leveraging pre-trained protein sequence embedding and neural networks, into tree search to enhance exploration.

In this paper, we propose to combine tree search with bandit machine learning. We begin by presenting a meta-algorithm (Algorithm 1). It proceeds by mimicking the directed evolution process, growing a tree from the root node, and gradually expanding via mutation and recombination. It uses a pre-trained embedding and a machine learning model for predicting fitness, and during the search process, it adopts a bandit strategy to update the predictor and actively explore the tree. This meta-algorithm provides a versatile framework for analyzing exploration in sequence space.

Results For theoretical analysis, we study a Bayesian setting where the true function map has a Gaussian Process prior distribution. We also assume Lipschitz continuity of the embedding map and local convexity of the fitness function. Under these simplified conditions, we show that the meta-algorithm with GP bandit can provably identify the optimal sequence and achieves a regret $O\left(\gamma_T\sqrt{T}\right)$, where γ_T is known as the maximal information gain. The theoretical analysis may apply to a broader class of bandit algorithms and be of independent interest.

Next, we fully develop the algorithm for numerical implementation, and make it compatible with a suite of bandit models including UCB and Thompson Sampling. We experiment with instances of the algorithm and compare with a variety of baselines, using simulation oracles trained from real-life protein function datasets AAV [8], TEM [19] and AAYL49 antibody [14] datasets. Experiment results show that tree-based methods achieve top performances across benchmarks and can efficiently find near-optimal designs with single-digit mutation counts.

2 Related Work

Protein engineering. The traditional DE works by artificially evolving a population of variants, via mutation and recombination, while constantly selecting high-potential variants [9, 10, 23, 20, 43, 28]. Many variations of DE methods allow targeted randomization of positions-of-interest or regions-of-interest. It is also possible to synthesize specific variants and operate on the combinatorial space likewise with high-throughput method, at a cost, and this allows directly applying a Gaussian process bandit algorithm [33]. See [49] for a high-level survey of machine learning-assisted protein engineering, and see [17, 5] for more examples.

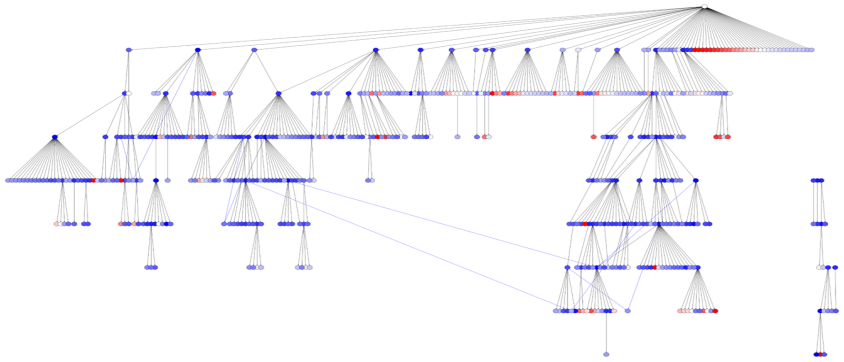


Figure 2: Visualization of tree search process of Algorithm 1 using the AAV oracle. The search initiates with a wild-type sequence (the root note) and in each round, we choose 100 sequences generated from the last round according to scores derived from UCB and TS by single mutation which leads to a node in the next layer and recombination of sequences (shown by blue edges) which leads to a jump to a layer with more mutations. A path to the optimal sequence is shown by the bold line.

Search algorithms for protein sequence design. Researchers have tried out various machine learning-based methods for sequence optimization. Bayesian Optimization(BO)[26] is a classical method for optimizing protein sequences. We use the code developed by Sinai et al. [38] who uses an ensemble of models for BO as one of the baselines. LaMBO[41] is also a BO-based algorithm that supports both single-objective and multi-objective. Design by adaptive sampling (DbAS; Brookes and Listgarten [7]) and conditioning by adaptive sampling (CbAS; Brookes et al. [6]) use a probabilistic framework. DyNA PPO[2] uses proximal policy optimization for sequence design. PEX MufacNet[30] is a local search method based on the principle of proximal optimization. We discussed more about the search algorithms in Appendix A.

Bandit learning. Bandit is a well-studied framework for optimizing with uncertainty, powerful in balancing exploration-exploitation trade-off – a core challenge in protein sequence optimization. Therefore, it is potential to study protein optimization from a bandit perspective and there is recent work [50] emerging from this middle ground. To balance exploration and exploitation, typical strategies are being optimistic in the face of uncertainty (OFU) based on the upper confidence bound (UCB) [1] and Thompson Sampling (TS) [34], which randomizes policies based on the posterior of the optimal policy. Regret guarantees have been established for different function classes of rewards, starting from the line of linear bandits. It was shown for the d-dim linear model upper and lower regret bounds meet at $\tilde{O}\left(d\sqrt{T}\right)$[4, 25, 1, 25, 34]. Later on, results from linear bandits have been extended to kernelized/ Gaussian process (GP) bandits. With γ_T defined as the maximal information gain and d being the dim of actions, [40] showed an upper bound of $\tilde{O}\left(\sqrt{dT\gamma_T}\right)$ and one of $\tilde{O}\left(\gamma_T\sqrt{T}\right)$ in the agnostic setting achieved by GP-UCB. Exploded by a factor of $\sqrt{d}$, $\tilde{O}\left(\gamma_T\sqrt{dT}\right)$ regret was shown by [11] for agnostic GP-TS.

Recently, there has been a growing interest in bridging the predictive power of deep neural networks with the exploration mechanism of bandit learning, which is known as neural bandits [52, 51, 22, 48]. Building upon the neural tangent kernel (NTK) technique to analyze deep neural network [22], NeuralUCB [52] can be viewed as a direct extension of kernel bandits [40]. Zhang et al. [51] proposed the Thompson Sampling version of neural bandits. Xu et al. [48] proposed NeuralLinUCB, which learns a deep representation to transform the raw feature vectors and performs UCB-type exploration in the last linear layer of the neural network.

3 Method

3.1 Problem Formulation

Let $x \in \mathcal{X}$ denote an amino-acid sequence, and let $F(x)$ denote a function measuring the fitness of the sequence. In practice, a known embedding map $\phi : \mathcal{X} \to \mathbb{R}^d$, mapping a sequence x to its

Algorithm 1 Meta algorithm

Input: wildtype x^{wt}, total rounds T
Initialization: Add x^{wt} to active node set $\mathcal{A}_0$ as root node.
for $t = 1, 2, ...T$ **do**
  Update bandit model: return a scoring function $F_t(\cdot) := f_t(\phi(\cdot))$.
  Tree search by Algorithm 2: $\mathcal{A}_t \leftarrow \text{TREESEARCH}(\mathcal{A}_{t-1}, F_t)$.
  Query the fitness $F(x)$ for all $x \in \mathcal{A}_t$ and append $\{(\phi(x), \tilde{F}(x))\}_{x \in \mathcal{A}_t}$ to the dataset.
end for

embedding vector $\phi(x)$, is often utilized to model the fitness $F(x)$ as $f^\star(\phi(x))$. Thus, we formulate the sequence design problem as

$$\max_{x \in \mathcal{X}} F(x) := f^\star(\phi(x)), \tag{1}$$

where $f^\star$ represents the unknown ground-truth function. Here $\mathcal{X}$ corresponds to the tree rooted at the $x^{wt} \in \mathcal{X}$ of a fixed depth.

The learning problem is to explore $\mathcal{X}$, conduct screens and collect data points of the form $(\phi(x), \tilde{F}(x))$, refine estimates of $f^\star$ in an iterative fashion. Here $\tilde{F}(x)$ represents a noisy measurement of the unknown true fitness $F(x)$.

The hardness of the problem is due to searching over the combinatorial space $\mathcal{X}$. Although we use an embedding map $\phi(x)$ to help learn F more accurately, we still have to work with discrete sequences and cannot make jumps in the embedding space. This nature of discrete sequence optimization is in sharp contrast to typical bandit settings.

3.2 Meta-algorithm

We present a meta-algorithm that combines bandit machine learning with local tree search; see Algorithm 1. Implementation details in Alg. 1 are delayed till Section 4.1 and Section 5.

Analysis of simple tree search For simplicity, let TREESEARCH_1 be the simplest tree search scheme that expands each active node in the tree by one level per iteration. Consider the max-fitness sequence in the active set:

$$x_t = \text{argmax}\{F_t(x) | x \in \mathcal{A}_t\}, \quad \mathcal{A}_t \leftarrow \text{TREESEARCH}_1(\mathcal{A}_{t-1}, F_t) \tag{2}$$

where $F_t(\cdot) = f_t(\phi(\cdot))$, f_t is sampled from Gaussian Process with monitoring (see Section 3.3 and (13) in Appendix H.0.2 for details).

For subsequent theory, we need a condition that the local search method makes sufficient improvement per iteration. In practice, this condition can be satisfied by tuning parameters and stopping conditions of the tree search heuristic.

Condition 3.1. *There exists $r > 0$ such that each iteration of tree search is able to find a solution better than the maximum in the local region of radius r, i.e., for $\forall t$*

$$F_t(x_t) \geq \max\{F_t(x) \mid \|\phi(x) - \phi(x_{t-1})\| \leq r\}.$$

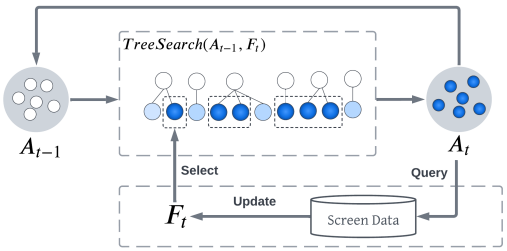


Figure 3: Diagram of the meta algorithm (Alg. 1)

In practice, it is believed that a good sequence embedding map ϕ shall capture the latent structure of the sequence. Thus, similar sequences often share similar embedding vectors, so $\{x : \|\phi(x) - \phi(x_{t-1})\| \leq r\}$ characterizes a local search region around x_{t-1}. In other words, properties of the embedding map ϕ critically connect locally searching in the discrete sequences to searching in the embedding space towards improving function value.

3.3 Regret Theory under GP Fitness

In this section, we consider a *simplified* mathematical setting to gain a basic theoretical understanding of the tree search-based bandit learning process. By using a Bayesian regret analysis, we provide a

theoretical abstraction of complicated real-world practice, and the results may have implications in a broader class of methods. We put some assumptions and definitions in Appendix G.

Theorem 3.2. *Under Assumption G.1, G.2, G.4 and Condition 3.1, Alg.1 updates f_t for $O(\gamma_T)$ times and the series of max-fitness sequences $\{x_t\}_{t=1}^T$ (defined by (2)) attains*

$$\mathrm{BayesRGT}(T) = O\left(\beta_T\sqrt{\lambda T\gamma_T} + B\gamma_T\left(1+\frac{4L^2N^2}{r^2}\right)\right), \tag{3}$$

where $\beta_T = \mathbb{E}\left[\|f^\star\|_k\right] + 2\sqrt{2\ln T + 1 + \gamma_{T-1}} + \sqrt{2\ln T}$ when the noise level $\lambda = 1 + \frac{1}{T}$ and the expectation is taken over the prior GP. r is inherit from Condition 3.1 and N is the depth of tree search.

Novelty and significance compared to classical results of bandits/Bayesian optimization. One may ask how Theorem 3.7 and its analysis compare to results from the classical bandits/Bayesian literature. We highlight that classical results *do not* apply our tree search algorithm that uses local search to gradually explore the action space. In particular, classical methods (such as plain vanilla UCB or TS) require finding the global maximum of a surrogate function in each iteration. This is far from the protein design practice where one wants to stay not too far from the wild type: looking for global max of $\hat{f}_t$ suffers from substantial computation overhead and can lead to instability and invalid solution.

In contrast, our method adds mutations gradually in a tree search process mimicking real-world screen practice. Analysis of such methods is much more complicated: We adapt classic arguments in optimization theory to analyze the progression of local search, which is further entangled with bandit exploration and information gain when new samples are collected. The proof idea is to view each local update as a form of proximal point optimization update and derive a novel recursive decomposition of the overall regret. Please see Appendix F for full proof details. Our analysis may be of interest to analyzing a broader class of bandit-guided evolutionary optimization algorithms.

Despite these differences, our regret bound nearly matches regret bound of classical bandit methods. The message is quite positive: The use of iterative local search does not impair quality of learning, with provable guarantee to explore the space of interests.

Universality of GP assumption GP provides a universal function approximation and it comes with both practical and theoretical implications. GP model and Bayesian optimization have been directly applied in protein engineering practice and proved effective in wet-lab experiments [33], which support our assumptions here. Further, GP can represent any kernel function space, which together with the modern deep learning theory [22] imply a powerful theoretical approximation to neural networks used in practice.

Figure 4: A demonstration of mutation and recombination.

Synergy between Tree search and Bandit Our analysis demonstrates how to create synergy between tree search and bandit machine learning. If we just use pure tree search without any bandit learning, tree search has a complexity that grows exponentially with depth. With the use of bandit learning, the sample complexity reduces to quadratic with tree depth.

4 Full Algorithm

4.1 Tree Search Heuristics

In this section, we present the full algorithm.

Algorithm 2 TREESEARCH($\mathcal{A}, F(\cdot)$)

Input: Active node set $\mathcal{A}$, scoring function $F(\cdot)$
Parameter: n_1, n_2, n_3, ρ
Initialization: Candidate node set $\mathcal{C} = \emptyset$, Query node set $\mathcal{Q} = \emptyset$
for $i = 1, 2, \cdots, n_1$ **do**
    Add random mutation to a random sequence $x \in \mathcal{A}$.
    Add the new sequence to candidate node set $\mathcal{C}$.
end for
for $i = 1, 2, \cdots, n_2$ **do**
    Sample x, y from $\mathcal{A}$ uniformly with replacement.
    Recombine x, y and add the new sequence to $\mathcal{C}$.
end for
Set new active node set $\mathcal{A}$ with n_3 sequence where ρ portion is from $\mathcal{A}$ and $1 - \rho$ portion is from the top scored sequences in $\{F(x) : x \in \mathcal{C}\}$.
return $\mathcal{A}$

Algorithm 2 expands a tree starting from the root wide-type sequence, i.e., $\mathcal{A} = \{x^{wt}\}$. Similar to the practice of directed evolution, we generate child nodes from parents in two ways: adding random mutation to one sequence and pairwise recombining two sequences. See Figure 4 for an illustration of these two operations. We use hyperparameters n_1, n_2 to control the rate of mutation and rate of recombination.

We use an active set $\mathcal{A}$ to keep track of the frontier of the tree search. At each round, the tree expands in a randomized way. New nodes with high scores measured by F will be kept track of and later used for updating the active set $\mathcal{A}$.

Ideally, one would want to keep the full search history in $\mathcal{A}$, but then the runtime will quickly blow up due to the exponential tree size. To make it more computationally affordable, we do not expand the active set $\mathcal{A}_t$, but keep it at a constant size in our implementation. We use a parameter ρ to control the portion of previously visited nodes to keep in the active set. This parameter can also be viewed as balancing depth and width in tree search.

4.2 Bandit Exploration

We use two classic exploration methods for scoring, Upper Confidence Bound (UCB) [25] and Thompson Sampling (TS; Russo and Van Roy [34]), to enable our algorithm to consider the potential of each node and look ahead. Details are referred to Appendix B.1.

5 Experiment

In this section, we evaluate our algorithm using oracles simulated to mimic the exploration process in wet-lab protein screens. Following Ren et al. [30] and Sinai et al. [38], we use oracles to mimic the fitness landscape as the wet-lab experiment is both time-consuming and cost-intensive. We build experiments around real-world protein datasets, such as AAV [8], TEM [19] and AAYL49 antibody [14]. Experiments results show that the tree search-based bandit outperforms existing baselines and leads to several interesting observations.

5.1 Datasets and Setup

5.1.1 Datasets

We experiment using three datasets from protein engineering studies and train oracles to simulate the ground-truth wet-lab fitness scores $f^\star$ of the landscape. The AAV and TEM oracles use pre-trained TAPE embedding [29] with a CNN model and the AAYL49 oracle is a downstream task from the pre-trained TAPE transformer model [29], more details about oracle could be found in B. For each round of the experiment, the model will query sequences from the black-box oracle, and the oracle will produce a fitness score depending on the sequence similar to the wet lab.

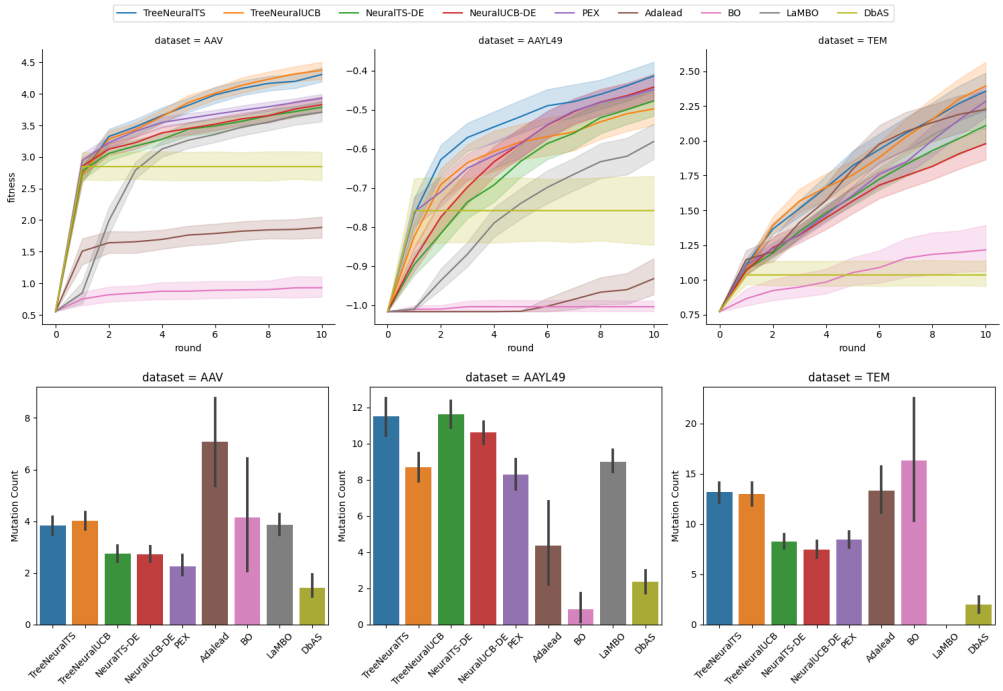


Figure 5: Learning curves of algorithms with comparison to baselines, tested over three datasets.

In particular, we use datasets of Adeno-associated virus 2 capsid protein (AAV) [8] which aimed for viral viability with search space 20^{28}; TEM-1 β-Lactamase (TEM) [19] which aimed for thermodynamic stability with search space 20^{286}; and, Anti-SARS-CoV-2 antibody (AAYL49) [14] which aimed for binding affinity with search space of 20^{118}. A full description can be found in B.2

5.1.2 Experiment Setup

In the experiment, we run each algorithm for 10 rounds with 100 query sequences per round for a fair comparison with our baselines. The algorithm cannot get any information about the oracle except the channel of queried sequences. Each test is run for 50 repeats using 50 different random seeds and measured the average performance across all seeds.

We use PEX [30], Adalead [38], Bayesian Optimization (BO) implemented by Sinai et al. [38], DbAS [7] implemented by Sinai et al. [38] and LaMBO[41] as our main baselines. We also use NeuralUCB-DE and NeuralTS-DE as the other two baselines which can be viewed as two downgraded versions of our algorithm without tree search. In detail, NeuralUCB-DE and NeuralTS-DE use uniform mutation with a mutation probability and recombination to generate new sequences. We also tested CbAS [6] but choose not report it, because it performs almost the same as DbAS, consistent with results Ren et al. [30]. The performance of the LaMBO on the TEM dataset is missing because the LaMBO algorithm has a memory limit and run-time efficiency issue on the TEM dataset due to the long sequence length. We do not use DyNAPPO [2] as our baseline due to tensorflow incompatibility.

For tree search-based algorithms, we use a neural network and follow [48]'s approach and use the second last dense layer's output of the neural network as $\phi(x)$ used by the UCB/TS formula. We reported tree search-based algorithms both uses UCB(TreeNeuralUCB) and TS(TreeNeuralTS). More details are included in Appendix B.

We run all the exploration algorithms on the same neural network structure for fair comparison. One hot encoding with CNN was used for the TEM and AAYL49 antibody datasets. We use TAPE [29] embedding with a simple 2-layer fully connected neural network for the AAV dataset to compensate for the limited sequence length.

5.2 Results and Analysis

Performances and Comparison. The experiment results are shown in Figure 5. Our tree search-based algorithm generally outperforms our baselines regarding the max fitness performance. The detailed ranking for all algorithms on each dataset can be seen in Appendix E. On the datasets TEM and AAYL49, all algorithms' curves seem not to reach convergence in 10 rounds. This is because TEM and AAYL49 deal with much longer sequences, so convergence is slower no matter what algorithm is used. We also conducted additional tests on the landscapes build by Thomas et al. [42] reported in B, the results are consistent with our oracle and the tree search method outperforms other baselines.

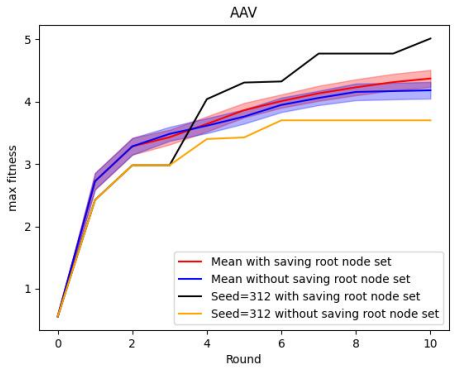


Figure 6: Effect of keeping parent nodes in active set.

For the performance of mutation count, tree search-based algorithms are not worse than PEX-based algorithms. On the AAV dataset, all algorithms can be controlled within 5 mutation counts. However, on datasets with larger sequence lengths, both NeuralUCB-DE and NeuralTS-DE have very large mutation counts.

We also run an experiment on the 3gfb oracles build by Thomas et al. [42] using the same setting. We report the results in appendix B.6. The results is consistent with other oracles and the tree search-based algorithms outperform the baselines regarding the max fitness performance.

Effect of Tree Search compared to DE In Figure 5, tree search-based algorithms generally outperform two baselines NeuralTS-DE and NeuralUCB-DE. Meanwhile, tree search-based algorithms only need fewer mutation counts than those non-treesearch methods.

Effect of Recombination. We also compare different recombination rates. The line with a zero recombination rate means that the algorithm only includes mutation and doesn't consider the recombination process. In Figure 7, we show that adding the recombination process to generating candidate children nodes is beneficial for improving performance, especially for small recombination rates. However, if the recombination rate becomes further larger, performance will gradually decrease and become even worse than the algorithm with a zero recombination rate.

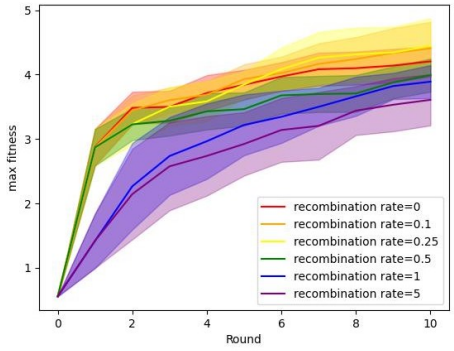


Figure 7: Effects of recombination of TreeNeuralBandit tested with the AAV oracle. By varying the rate of recombination, we see that performance tops with a proper choice of positive recombination rate (yellow curve).

Effect of keeping parent/root nodes in active set. We use a parameter ρ to control updates of the active set, i.e., saving part of the root node set in each round. Figure 6 shows one seed's curve where the exploration got stuck when the active set is not updated to save enough parents/root. By saving previous nodes, it balances width-first and depth-first better and allows the algorithm to find higher-value nodes.

Effect of diversity promotion. We tested adding an additional diversity bonus and observe a trade-off between group diversity and group fitness. Details are included in Appendix D.

6 Conclusion

This paper proposes a tree-based bandit learning method for enhancing the process of protein engineering. The methods expand the tree from the initial sequence (wild-type) with the guidance of bandit machine learning models using randomized tree search heuristics, machine learning models, pre-trained embeddings. We have shown the algorithm can discover a near-optimal design under simplified assumptions, with experimental validation.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24:2312–2320, 2011.

[2] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklxbgBKvr.

[3] Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

[4] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[5] Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS computational biology*, 13(10):e1005786, 2017.

[6] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pages 773–782. PMLR, 2019.

[7] David H Brookes and Jennifer Listgarten. Design by adaptive sampling. *arXiv preprint arXiv:1810.03714*, 2018.

[8] Drew H. Bryant, Ali Bashir, Sam Sinai, Nina K. Jain, Pierce J. Ogden, Patrick F. Riley, George M. Church, Lucy J. Colwell, and Eric D. Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, Jun 2021. ISSN 1546-1696. doi: 10.1038/s41587-020-00793-4. URL https://doi.org/10.1038/s41587-020-00793-4.

[9] Keqin Chen and Frances H Arnold. Enzyme engineering for nonaqueous solvents: random mutagenesis to enhance activity of subtilisin e in polar organic media. *Bio/Technology*, 9(11): 1073–1077, 1991.

[10] Keqin Chen and Frances H Arnold. Tuning the activity of an enzyme for unusual environments: sequential random mutagenesis of subtilisin e for catalysis in dimethylformamide. *Proceedings of the National Academy of Sciences*, 90(12):5618–5622, 1993.

[11] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

[12] Janardhan Rao Doppa. Adaptive experimental design for optimizing combinatorial structures. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4940–4945, 2021.

[13] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing, 2020. URL https://arxiv.org/abs/2007.06225.

[14] Emily Engelhart, Ryan Emerson, Leslie Shing, Chelsea Lennartz, Daniel Guion, Mary Kelley, Charles Lin, Randolph Lopez, David Younger, and Matthew E. Walsh. A dataset comprised of binding interactions for 104,972 antibodies against a sars-cov-2 peptide. *Scientific Data*, 9(1): 653, Oct 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01779-4. URL https://doi.org/10.1038/s41597-022-01779-4.

[15] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

[16] Richard Fox, Ajoy Roy, Sridhar Govindarajan, Jeremy Minshull, Claes Gustafsson, Jennifer T Jones, and Robin Emig. Optimizing the search algorithm for protein engineering by directed evolution. *Protein engineering*, 16(8):589–597, 2003.

[17] Richard J Fox, S Christopher Davis, Emily C Mundorff, Lisa M Newman, Vesna Gavrilovic, Steven K Ma, Loleta M Chung, Charlene Ching, Sarena Tam, Sheela Muley, et al. Improving catalytic function by prosar-driven enzyme evolution. *Nature biotechnology*, 25(3):338–344, 2007.

[18] Chase R Freschlin, Sarah A Fahlberg, and Philip A Romero. Machine learning to navigate fitness landscapes for protein engineering. *Current Opinion in Biotechnology*, 75:102713, 2022.

[19] Courtney E Gonzalez and Marc Ostermeier. Pervasive pairwise intragenic epistasis among sequential mutations in TEM-1 β-lactamase. *J. Mol. Biol.*, 431(10):1981–1992, May 2019.

[20] Edward G Hibbert and Paul A Dalby. Directed evolution strategies for improved enzymatic performance. *Microbial Cell Factories*, 4(1):1–6, 2005.

[21] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Learning protein fitness models from evolutionary and assay-labeled data. *Nature Biotechnology*, 40(7):1114–1122, Jul 2022. ISSN 1546-1696. doi: 10.1038/s41587-021-01146-5. URL https://doi.org/10.1038/s41587-021-01146-5.

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[23] Olga Kuchner and Frances H Arnold. Directed evolution of enzyme catalysts. *Trends in biotechnology*, 15(12):523–530, 1997.

[24] Benjamin Lansdell, Sofia Triantafillou, and Konrad Kording. Rarely-switching linear bandits: optimization of causal effects for the real world. *arXiv preprint arXiv:1905.13121*, 2019.

[25] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[26] Jonas Mockus. Bayesian approach to global optimization: Theory and applications. 1989. URL https://api.semanticscholar.org/CorpusID:120322743.

[27] Erik Nijkamp, Jeffrey Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani. Progen2: Exploring the boundaries of protein language models, 2022. URL https://arxiv.org/abs/2206.13517.

[28] Michael S Packer and David R Liu. Methods for the directed evolution of proteins. *Nature Reviews Genetics*, 16(7):379–394, 2015.

[29] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, 2019.

[30] Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning*, pages 18520–18536. PMLR, 2022.

[31] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019. doi: 10.1101/622803. URL https://www.biorxiv.org/content/10.1101/622803v4.

[32] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021. doi: 10.1073/pnas.2016239118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2016239118.

[33] Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3): E193–E201, 2013.

[34] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[35] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[36] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):1–11, 2021.

[37] Richard W. Shuai, Jeffrey A. Ruffolo, and Jeffrey J. Gray. Generative language modeling for antibody design. *bioRxiv*, 2022. doi: 10.1101/2021.12.13.472419. URL https://www.biorxiv.org/content/early/2022/12/20/2021.12.13.472419.

[38] Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric Kelsic. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint*, 2020.

[39] George P Smith and Valery A Petrenko. Phage display. *Chemical reviews*, 97(2):391–410, 1997.

[40] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[41] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20459–20478. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/stanton22a.html.

[42] Neil Thomas, Atish Agarwala, David Belanger, Yun S. Song, and Lucy J. Colwell. Tuned fitness landscapes for benchmarking model-guided protein design. *bioRxiv*, 2022. doi: 10.1101/2022.10.28.514293. URL https://www.biorxiv.org/content/early/2022/10/30/2022.10.28.514293.

[43] Nicholas J Turner. Directed evolution drives the next generation of biocatalysts. *Nature chemical biology*, 5(8):567–573, 2009.

[44] Chenyu Wang, Joseph Kim, Le Cong, and Mengdi Wang. Neural bandits for protein sequence optimization. In *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, pages 188–193. IEEE, 2022.

[45] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[46] Greg Winter, Andrew D Griffiths, Robert E Hawkins, and Hennie R Hoogenboom. Making antibodies by phage display technology. *Annual review of immunology*, 12(1):433–455, 1994.

[47] Zachary Wu, S. B. Jennifer Kan, Russell D. Lewis, Bruce J. Wittmann, and Frances H. Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019. doi: 10.1073/pnas.1901979116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1901979116.

[48] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. *CoRR*, abs/2012.01780, 2020. URL https://arxiv.org/abs/2012.01780.

[49] Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019.

[50] Hui Yuan, Chengzhuo Ni, Huazheng Wang, Xuezhou Zhang, Le Cong, Csaba Szepesvári, and Mengdi Wang. Bandit theory and thompson sampling-guided directed evolution for sequence optimization. *arXiv preprint arXiv:2206.02092*, 2022.

[51] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

[52] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

A Related Work Details

Bayesian Optimization(BO)[26] is a classical method for optimizing protein sequences. Sinai et al. [38] implements the code of an ensemble of models for Bayesian Optimization(BO) as one of its baselines. However, BO is computationally inefficient because it searches the entire sequence space in each round. LaMBO[41] is also a BO-based algorithm that supports both single-objective and multi-objective, but it is again computationally inefficient on sequences with long lengths. Design by adaptive sampling (DbAS; Brookes and Listgarten [7]) uses a probabilistic framework as well as an adaptive sampling method to maximize the fitness landscape. Similar to DbAS, conditioning by adaptive sampling (CbAS; Brookes et al. [6]) shares the same probabilistic method with regularization to solve the problem that data distribution may differ from the training distribution. DyNA PPO[2] uses proximal policy optimization for sequence design. However, DyNA PPO may search the space far from the wild type, which leads to a high mutation count. AdaLead[38] is an evolutionary algorithm with greedy selection: In each round, it selects the sequences near the best sequence derived in the last round on the fitness landscape to do recombination and mutation. Then it chooses the best sequence set according to the approximate model to query the ground-truth landscape model. PEX MufacNet[30] is a local search method based on the principle of proximal optimization, with a MutFAC network tailored to mutated sequence data. It also follows a greedy strategy to select sequences to query and does not employ bandit-based exploration.

B Experiment Details

B.1 Bandit Exploration

We apply the bandit-based exploration mechanism to select sequences for fitness evaluation. In the UCB version, we use a parameter α to control the explore-exploit tradeoff and we calculate the UCB score using $F_{\text{UCB}}(x) = \theta_{t-1}^T \psi(x) + \alpha\sqrt{\psi^T(x) A_{t-1}^{-1} \psi(x)}$, where ψ is some feature map, θ_t is the model parameter and the covariance matrix is defined as $M_t = \lambda \mathbf{I} + \sum_{i=1}^{t-1} \psi(x)\psi(x)^\top$. In the Thompson Sampling version, F_t takes the form $F_{\text{TS}}(x) = \tilde{\theta}^T \psi(x)$, where $\tilde{\theta}$ are parameters sampled from a posterior distribution $\mathcal{N}\left(\theta_{t-1}, \alpha M_{t-1}^{-1}\right)$.

We consider two predictive models in our tree search bandits, linear model, and neural networks. For the linear model, we can simply take ψ to be a pre-trained embedding $\phi = \psi$, and our experiment uses the TAPE [29] embedding, and θ_t is the closed form solution of ridge regression. For neural networks, we construct a neural net taking $\phi(x)$ as input and use the second last dense layer's output of the neural network as $\psi(x)$ for calculating bonus or posterior distribution, following the shallow exploration idea of [48].

B.2 Full Dataset Description

Adeno-associated virus 2 capsid protein (AAV) AAVs have been widely used in gene therapies. A particular interest is changing the sequence of the capsid protein to address the issues of tissue tropism and manufacturability. We collected dataset from Bryant et al. [8], which contains 283953 different length sequences engineered a region (positions 561-588) on the AAV2 VP1 protein for optimizing the viral viability. We limited the search space to the sequences which have the same length (28 amino acids) as the wildtype sequence and the search space is 20^{28}.

TEM-1 β-Lactamase (TEM) TEM-1 β-Lactamase is an E. coli protein that efficiently hydrolyzes penicillins and many cephalosporins, affording antibiotics resistance. It is used to understand the mutational effects and epistatic interactions between mutations in the fitness landscape. We collected the fitness data from [19] which contains 16924 sequences. The sequence length is 286 which results in 20^{286} search space.

Anti-SARS-CoV-2 antibody (AAYL49) Recombinant antibody is a common therapeutic modality that is widely adopted in treating many diseases, including cancers, metabolic disorders, and viral infections. The Fv region of antibodies is the most important region for antigen binding, affording target specificity of the binding event. Engelhart et al. [14] presented a dataset of quantitative binding scores for mutational variants of several antibodies against a SARS-CoV2 peptide. We use the

AAYL49 heavy chain part of the dataset which includes 26454 sequences and the sequence length is 118 which results in 20^{118} search space.

Table 1: Dataset Summary

Dataset	Aim	Size	WT Length	Search Space
AAV [8]:	Viral Viability	283953	28	20^{28}
TEM [19]:	Thermodynamic Stability	16926	286	20^{286}
AAYL49 [14]:	Binding Affinity	13922	118	20^{118}

B.3 Data Preprocessing

We preprocess each of the datasets to keep only the sequence and fitness information. The preprocessing of each dataset is as follows:

- **AAV:** We removed the sequence with type "stop" as the stop codon also remove the amino acids after the editable regions. The column "score" was used for fitness. The dataset after the preprocessing has 283953 sequences.
- **TEM:** We converted the mutated position to the full single and double mutation sequences using the wild-type and assign the corresponding single and double mutation fitness. The dataset after the preprocessing contains 16924 sequences
- **AAYL49:** We keep "MIT_Target" and the "AAYL49" part of the dataset. We take the mean of the non-empty fitness score across different experimental repeats. We dropped the data with empty fitness scores for all three repeats. The preprocessed dataset contains 13922 sequences.

B.4 Oracle Training

For AAV and TEM datasets, we used the pretrained transformer model of TAPE [29] to generate a 768-dimension embedding as the input of the CNN model from [38] shown as in 2. We used a 80/20 train-test split for the preprocessed datasets with shuffle. The oracle was trained using the training set with learning rate 1×10^{-4}, batch size of 256 for maximum of 3000 rounds with an Adam optimizer. The training will be early stopped if the loss does not decrease for 100 epochs.

Table 2: CNN Architecture used by Oracle

Input:	TAPE Embedding
1D - CNN	Number of Filters: 32, Kernal Size: 5
	ReLU Activation
1D - CNN	Number of Filters: 32, Kernal Size: 5
	ReLU Activation
1D - CNN	Number of Filters: 32, Kernal Size: 5
	ReLU Activation
	Global Max Pooling
Dense	Output Size: 256
	ReLu Activation
Dense	Output Size: 256
	ReLu Activation
	Dropout: $p = 0.25$
Dense	Output Size : 1

For the AAYL dataset, the method used for AAV and TEM results in a low spearman correlation of 0.273. Hence, we trained a downstream task from the pretrained TAPE transformer model from Rao et al. [29]. We also used the same used the same 80/20 train-test split and train the model with parameter suggested by Rao et al. [29] with batch size of 32, learning rate 1×10^{-5} with linear warm up until loss does not decrease for 10 rounds.

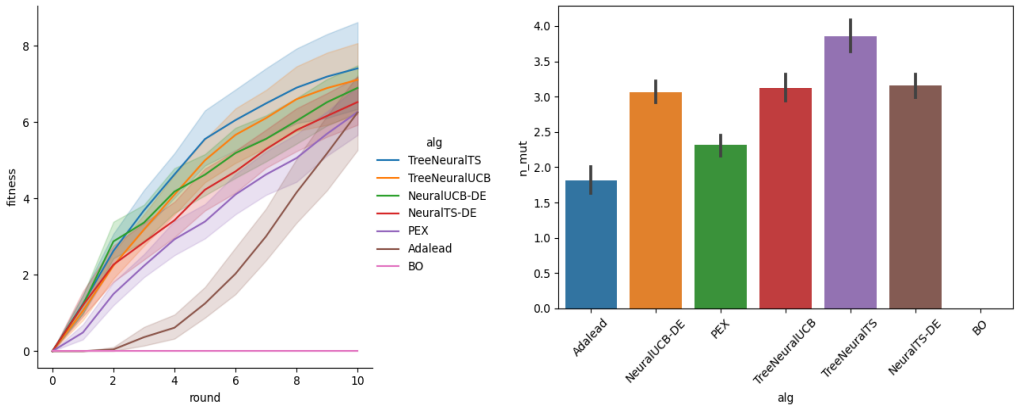


Figure 8: Experiment Result on 3gfb [42] Oracle

We tested the correlation between the oracle prediction and ground-truth fitness on the test set which shown in 3

Table 3: Oracle Correlations

DATASET:	AAV	TEM	AAYL49
SPEARMAN:	0.838	0.713	0.502
PEARSON:	0.876	0.681	0.584

B.5 Experiment Details

We run all the exploration algorithms on the same neural network structure for fair comparison. Onehot encoding with CNN was used for the TEM and AAYL49 antibody datasets. We use TAPE [29] embedding with simple 2 layer fully connected neural network for the AAV dataset to compensate the limited sequence length.

All algorithms was run on NVIDIA A10G GPU for 50 different random seeds. With 4-5 algorithms running in parallel, the total time is around 6 hours. The neural network is run with learning rate of 10^{-4}, batch size of 64 for maximum of 3000 epochs with a early stopping when loss does not decreases for 50 rounds. The baseline algorithms was run using their recommended setting.

B.6 Additional Experiments on Other Landscape

We also run experiment on the 3gfb oracle designed by Thomas et al. [42] for all algorithm using the same setting as 5 on the same model of onehot encoding and CNN. The maximum fitness curve with the mutation count is shown in 8. For convience, the legend of the fitness plot was in descending order regarding max fitness performance. We could see the performance is consistent with the results on other oracles with TreeNeuralTS and TreeNeuralUCB outperform other exploration algorithms. The mutation counts of TreeNeuralTS and TreeNeuralUCB are comparable to other algorithms and are reletively low compare to the wild-type sequence length of 347. The mutation count and fitness of BO are both zero as it fail to find any sequence with higher fitness than wild-type so the maximum fitness sequence was the initial wild-type sequence for all rounds.

B.7 Simulation Results and Details

We are also interested in testing our algorithms on synthetic data to bridge the gap between theory and. We use Gaussian Process to generate synthetic data and build a new oracle[45]. For consistency, our simulation is controlled within 10 rounds. In Figure 9, we can find that tree-based algorithms can achieve higher fitness scores in 10 rounds and are closer to the maximum fitness than those algorithms which are not tree-based. Also, in the left panel of Figure 9, tree-based algorithms' regret converges

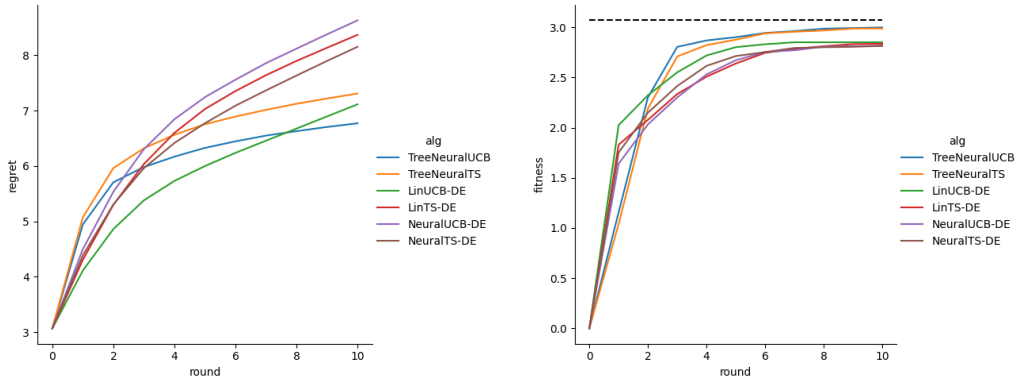


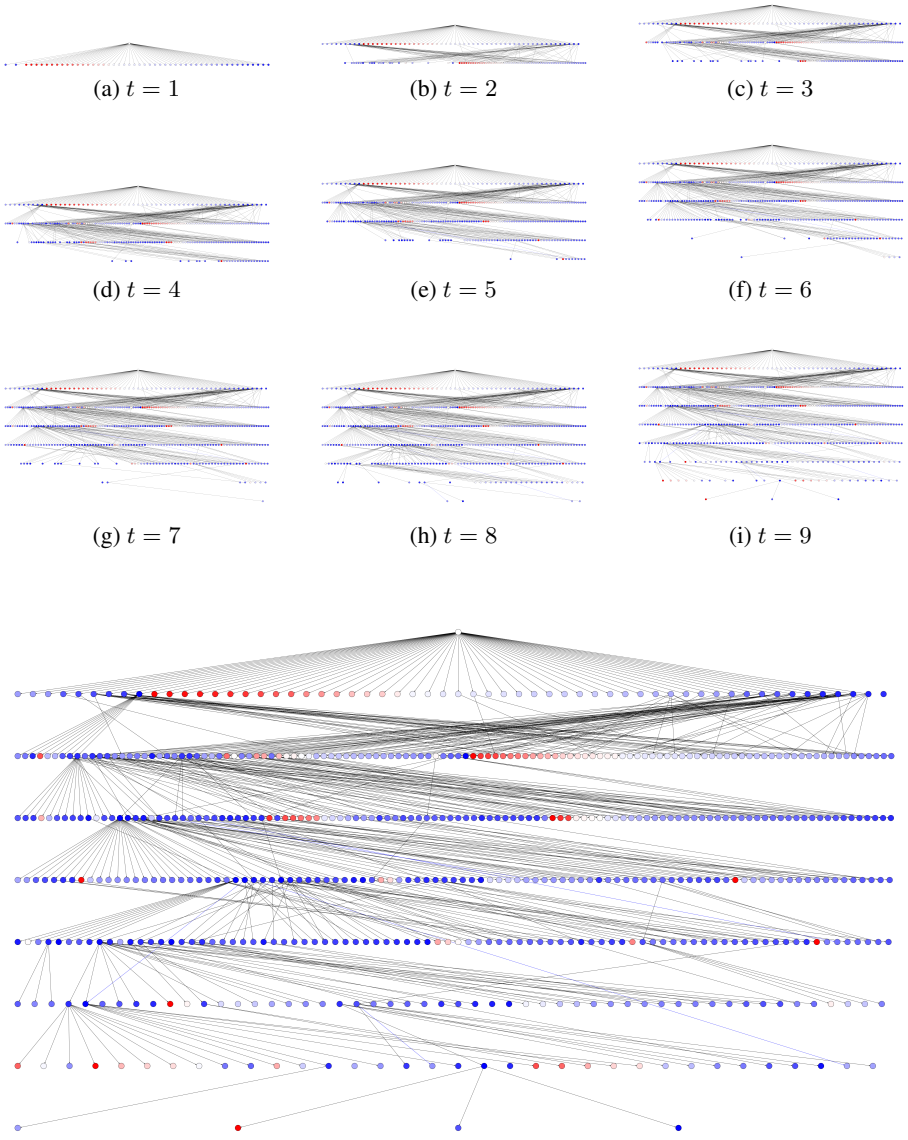
Figure 9: Regret and fitness curves of algorithms with comparison to baselines over synthetic data.

and maintains a low level while non-tree algorithms have the trend to increase in future rounds like LinUCB-DE or have higher regret.

B.8 Code Availability

We make our code public anonymously to help confirm the details. Our baseline methods like BO, PEX, Adalead, CbAS and DbAS are implemented by their authors and we directly use their code. The anonymous link to our code is https://anonymous.4open.science/r/TreeSearchDE-public-0D26/README.md

C Visualization of the Full Tree Search Process by Round



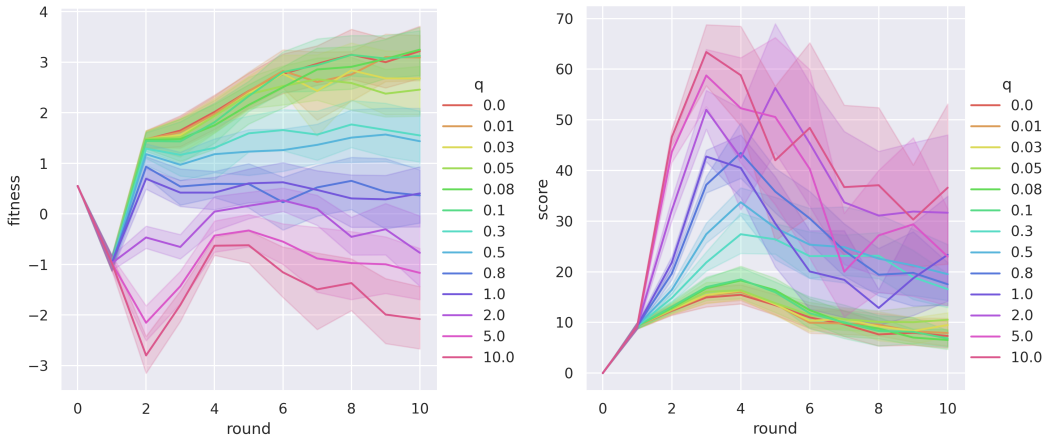
(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

(d) $t = 4$ (e) $t = 5$ (f) $t = 6$

(g) $t = 7$ (h) $t = 8$ (i) $t = 9$

(j) $t = 10$

Figure 10: Visualization of the full tree search process by round



(a) Average Fitness

(b) Diversity Score

Figure 11: Experiment Results for Different q's

D Diversity-Fitness Trade Off

We aim to improve the diversity of our algorithm by adding a diversity constraint. The diversity score of the children node is defined as its distance to the center of nodes explored in the previous rounds in the embedding space, i.e.,

$$S_{diversity} = \sqrt{\sum_{i=1}^{n} (\phi(x)_i - \overline{\phi(x^{explored})}_i)^2}, \tag{4}$$

where $\overline{\phi(x^{explored})}$ is the center of explored sequences in the embedding space and n is the dimension of the embedding space. Then the score function for determining the candidate node set is modified as:

$$S_{total} = S_{origin} + q * S_{diversity}, \tag{5}$$

where q is the weight for $S_{diversity}$.

D.1 Experiment

We tested the effect of different parameters of q's on both diversity and fitness. In particular, we run the algorithm with $q \in \{0, 0.01, 0.03, 0.05, 0.08, 0.1, 0.3, 0.5, 0.8, 1, 2, 5, 10\}$ each for 10 random seeds. From 11, we observed a trade-off between fitness and diversity, i.e., a higher q will lead to a higher diversity but at the same time a lower average fitness.

D.2 t-SNE

We plot a t-SNE plot for all q's for one particular random seed shown in Figure 12. The t-SNE plot verifies the tread-off we discussed in D.1. As the q increases, the queried sequences are more spread out on the t-SNE plot which verifies the improvement in diversity. In contrast, when the q is low, the queried sequences are concentrated on a smaller region of the t-SNE plot (local optimal) especially in the later round (shown in red color), and have a better fitness caused by high fitness sequences in this local optimal region.

E Ranking of Algorithms

In Figure 13, the legend ranks the algorithm in descending order regarding max fitness performance for convenience. In Table 4, we show the detailed values for all algorithms.

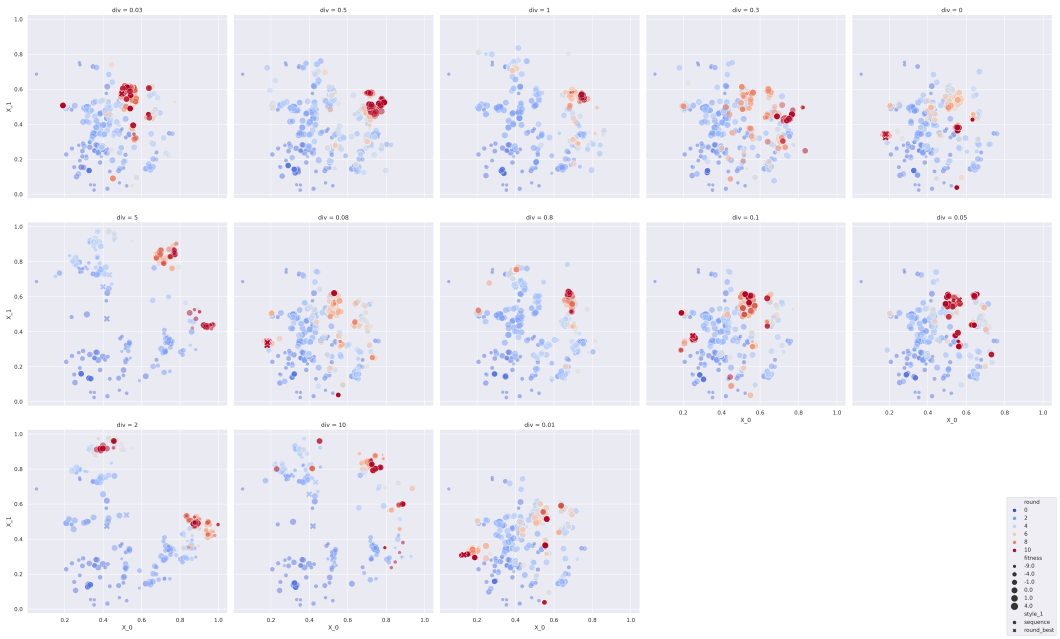


Figure 12: t-SNE Plot for Different q's

Table 4: Average Max Fitness of Algorithms

DATASET:	AAV	AAYL49	TEM
TREENEURALUCB:	4.3723	-0.4976	2.394493
TREENEURALTS:	4.3056	-0.4132	2.356243
NEURALTS-DE:	3.7363	-0.4773	2.108297
NEURALUCB-DE:	3.7697	-0.4421	1.978440
PEX:	3.9286	-0.4482	2.283808
ADALEAD:	1.8835	-0.9322	2.225310
BO:	0.9299	-1.0038	1.215830
LAMBO:	3.7085	-0.5813	N/A
DBAS:	2.8481	-0.7580	1.035354

F Discussion on Limitations

Our theoretical result serves its purpose well to provide an understanding of using local mutation to optimize protein utility. Our bound is a Bayesian regret bound. Also, it is an open question to derive a frequentist bound for, which is of interest in practice when ground truth f is agnostic. Current Bayesian regret does not directly apply to a frequentist one.

The functionalities of the protein sequences are complicated. So far, none of our oracles or oracles from Adalead and PEX-Mufacnet, or the pre-trained protein language models such as ESM or TAPE could accurately predict the whole landscape. However, we believe the significance of our experiment is not to build an oracle that could perfectly represent the landscape but instead to demonstrate a new exploration algorithm that could be applied to all protein landscapes. As from the experiments that our algorithm performs well on all oracles we built as well as on the new dataset from antibodies, we should expect our algorithm will perform well on the actual landscape.

Also, we are actively building our wet lab to apply our algorithm into practice, but it is acknowledged that wet-lab experiment takes a very long time for validation.

We do not allow using our models in production without authorization due to potential negative social impacts.

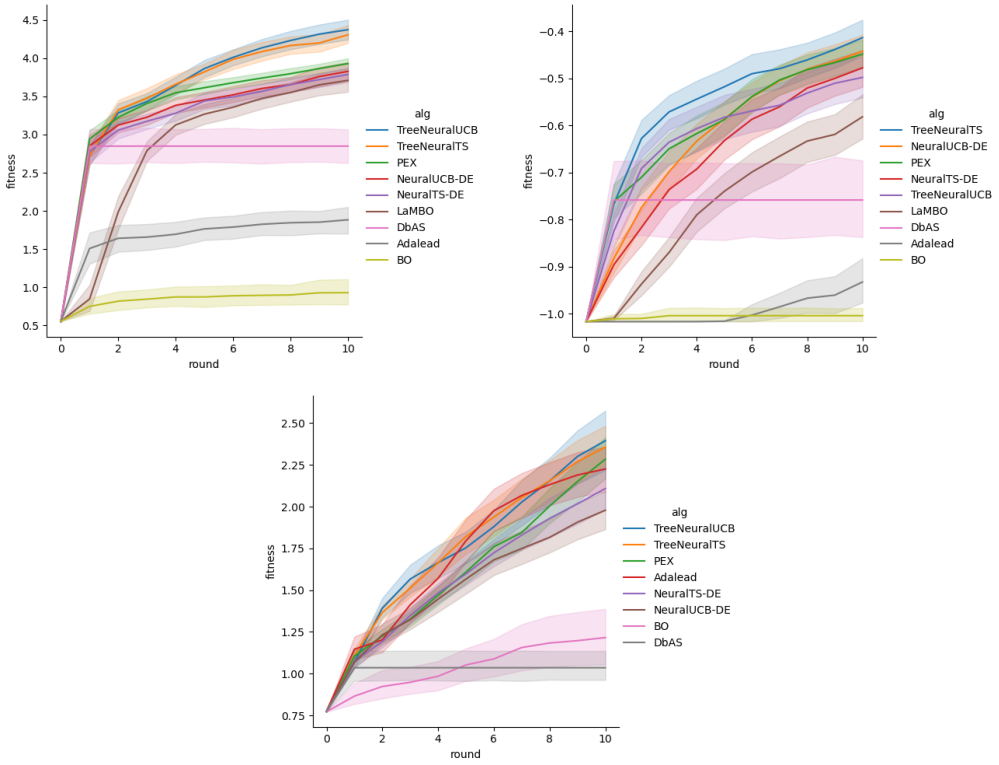


Figure 13: TreeBandit Compared to baselines

G Regret Theory under GP Fitness Supplementary Materials

G.1 GP Modeling of $f^\star$

For theoretical analysis, we consider a Bayesian learning setting where the ground truth $f^\star$ is assumed to follow a Gaussian Process (GP) prior, and the evaluation of $F(x)$ is assumed to be corrupted by Gaussian noise. GP model is well studied and widely applied in machine learning [35], especially in classic bandit optimization [40, 11].

Assumption G.1. $f^\star : \mathbb{R}^d \to R$ is a sample from the Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ (with known kernel function $k(\cdot,\cdot)$) as priori, i.e.

$$f^\star(\phi(x)) \sim \mathcal{GP}\left(0, k(\phi(x), \phi(x'))\right). \tag{6}$$

Assumption G.2. For any sequence x, querying its fitness $F(x)$ returns a noisy feedback $\tilde{F}(x)$ corrupted by a Gaussian, i.e.

$$\tilde{F}(x) = f^\star(\phi(x)) + \epsilon, \tag{7}$$

where $\epsilon \sim \mathcal{N}\left(0, \lambda\right), \lambda > 1$ and is sampled independently from the history.

For more details on the Gaussian Process and the update rule of f_t, please refer to the appendix.

G.2 Bayesian Regret

Motivated by the practical consideration that high-potential variants usually appear within a small count of mutations from a wildtype and function approximation generalizes better in a local region where most data lies, we focus on a local searching region $\bar{\mathcal{X}} := \{x : d(x, x^{wt}) \leq N\} \subset \mathcal{X}$ and aim to identify the optimal sequences within $\bar{\mathcal{X}}$. Here $d(\cdot,\cdot)$ denotes the hamming distance between any two arbitrary sequences in $\mathcal{X}$ and N controls the mutation counts (the depth of tree search).

Therefore, under the setups on GP function and local searching, the Bayesian regret of a learning algorithm is defined as :

Definition G.3 (Bayesian Regret)**.** For a series of sequences $\{x_t\}_{t=1}^T$ within the searching region $\bar{\mathcal{X}} \subset \mathcal{X}$, the accumulated Bayesian regret is defined as

$$\mathrm{BayesRGT}(T) = \mathbb{E}\left[\sum_{t=1}^{T}\left(\max_{x\in\bar{\mathcal{X}}} F(x) - F(x_t)\right)\right], \tag{8}$$

where $\mathbb{E}$ is taken over the prior Gaussian process from which $f^\star$ is sampled and other randomness in $\{x_t\}_{t=1}^T$.

Next in Theorem 3.2, Alg. 1 is proven to explore the local sequence space via tree search and bandit learning, while attaining low regret.

Assuming ϕ is Lipschitz, the local searching region $\bar{\mathcal{X}}$ is mapped to $\mathcal{D} := \{\phi(x) : \|\phi(x) - \phi(x^{wt})\| \leq R := LN\}$, a region surrounding $\phi(x^{wt})$ in $\mathbb{R}^d$.

Assumption G.4. At each time step t, f_t is bounded in $\mathcal{D}$. For $\forall t, z \in \mathcal{D}$,

$$|f_t(z)| \leq B. \tag{9}$$

Also, assume f_t is locally concave in $\mathcal{D}$ and attains its maximal value in the interior of $\mathcal{D}$.

G.3 Rate of regret

The highest order term in (3) is of $O\left(\gamma_T\sqrt{T}\right)$, which matches the results by [40, 11] studying GP bandits. (3) turns out to be $\tilde{O}\left(d\sqrt{T}\right)$ (with $\gamma_T = d\log T$) when $k(\cdot,\cdot)$ in the prior is linear, which downgrades the GP model to the d-dim Bayesian linear model. It's worth mentioning that the recovered $\tilde{O}\left(d\sqrt{T}\right)$ bound improves [50] by a factor of $\sqrt{d}$, benefiting from the stability brought by f_t's rare switching.Here the information gain is closely related to the "effective dimension" of the chosen kernel, which is a natural and common complexity metric for online exploration.

H Theory Details

H.0.1 Preliminaries of GP

Conditioned the set of screen data $\{(\phi(x_i), \tilde{F}(x_i))\}_{i=1}^{t-1}$ collected up to step t, the posterior distribution of $f^\star$ corresponding to the prior $\mathcal{GP}\left(0, k(\cdot,\cdot)\right)$ is still a Gaussian process $\mathcal{GP}(m_t, k_t)$.

Definition H.1 (Posterior $\mathcal{GP}(m_t, k_t)$)**.** $m_t : \mathbb{R}^d \to \mathbb{R}$ and $k_t : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are defined on the embedding space and have the form:

$$m_t(\phi(x)) = \mathbf{k_{t-1}}(\phi(x))^\top \left(K_{t-1} + \lambda I\right)^{-1} \tilde{F}_{t-1}, \tag{10}$$

$$\begin{aligned} k_t\left(\phi(x), \phi(x')\right) &= k\left(\phi(x), \phi(x')\right) \\ &\quad - \mathbf{k_{t-1}}(\phi(x))^\top \left(K_{t-1} + \lambda I\right)^{-1} \mathbf{k_{t-1}}(\phi(x')), \end{aligned} \tag{11}$$

where $\mathbf{k_t}(\phi(x)) = \left(k\left(\phi(x_1), \phi(x)\right), \cdots, k\left(\phi(x_t), \phi(x)\right)\right)^\top$, K_t is the kernel matrix $\left[k(\phi(x), \phi(x'))\right]_{x,x'\in A_t}$, $A_t = \{x_1, \cdots, x_t\}$ and $\tilde{F}_t = \left(\tilde{F}(x_1), \cdots, \tilde{F}(x_t)\right)^\top$.

To establish cumulative regret bounds for GP optimization, a quantity to define is the maximum information gain γ_t:

Definition H.2 (Maximum Information Gain)**.** Define the maximum information gain over t rounds as

$$\gamma_t := \max_{A\subset\mathbb{R}^d : |A| = t} \mathrm{I}\left(\mathbf{y}_A; \mathbf{f^\star}_A\right), \tag{12}$$

where $\mathrm{I}\left(\mathbf{y}_A; \mathbf{f^\star}_A\right)$ denotes the mutual information between $\mathbf{f^\star}_A := [f^\star(\phi(x))]_{x\in A}$ and $\mathbf{y}_A := \mathbf{f^\star}_A + \boldsymbol{\epsilon}_A$, where $\boldsymbol{\epsilon}_A \sim \mathcal{N}\left(0, \lambda I_{|A|}\right)$. Thus $\mathrm{I}\left(\mathbf{y}_A; \mathbf{f^\star}_A\right) = \frac{1}{2}\log\det\left(I + \lambda^{-1}K_A\right)$, where $K_A = \left[k\left(\phi(x), \phi(x')\right)\right]_{x,x'\in A}$.

H.0.2 Rarely Switching in f_t

Posterior sampling is a natural choice to update f_t, that is sample $f_t \sim \mathcal{GP}(m_t, k_t)$ at each step t. For the sake of a tighter regret bound, we analyse the version of Alg.1 where let $V_t = K_{t-1} + \lambda I$,

$$\begin{cases} f_t \sim \mathcal{GP}(m_t, k_t), & \text{if } \det(V_t)/\det(V_{pre(t)}) > 2; \\ f_t \leftarrow f_{t-1}, & \text{otherwise.} \end{cases} \tag{13}$$

with $pre(t)$ denotes the last time step provious to t when f_t is sampled from the posterior. Here, we keep track of the growth in $\det(K_t + \lambda I)$ and keep f_t the same unless the determinant is doubled. Rarely switching is a common technique in online learning [1, 24] to save computation and it helps stabilize Alg.1 by reducing the fluctuation in f_t over time. By using this adaptive update schedule, we will show in Theorem 3.2 that a total of $O(\gamma_T)$ updates is sufficient without compromising the efficiency in learning $f^\star$.

I Proof of Themorem 3.2

Recall the definition of Bayesian regret and write out an equivalent expression with embedding $\phi(x)$:

$$\begin{aligned} \text{BayesRGT}(T) &= \mathbb{E}\left[\sum_{t=1}^{T}\left(\max_{x\in\mathcal{X}} F(x) - F(x_t)\right)\right] \\ &\Leftrightarrow \mathbb{E}\left[\sum_{t=1}^{T}\left(\max_{\phi(x)\in\mathcal{D}} f^\star(\phi(x)) - f^\star(\phi(x_t))\right)\right], \end{aligned} \tag{14}$$

as well as Condition 3.1 characterizing the max-fitness series $\{x_t\}_{t=1}^T$ to study:

$$\begin{aligned} & F_t(x_t) \geq \max\{F_t(x) \mid \|\phi(x) - \phi(x_{t-1})\| \leq r\} \\ \Leftrightarrow\ & f_t(\phi(x_t)) \geq \max\{f_t(\phi(x)) \mid \|\phi(x) - \phi(x_{t-1})\| \leq r\}. \end{aligned} \tag{15}$$

From the RHS of (14) and (15), the embedding vectors $\{\phi(x_t)\}_{t=1}^T$ fully determine the regret and identify the max-fitness sequence $\{x_t\}_{t=1}^T$. Therefore, the proof of Theorem 3.2 can be conducted on the embedding level, that is, essentially it suffices to prove Alg.3(a revision of Alg.1) satisfies

$$\begin{aligned} \text{BayesRGT}(T) &= \mathbb{E}\left[\sum_{t=1}^{T}\left(\max_{x\in\mathcal{D}} f^\star(x) - f^\star(x_t)\right)\right] \\ &= O\left(\beta_T\sqrt{\lambda T\gamma_T} + B\gamma_T\left(1 + \frac{4L^2N^2}{r^2}\right)\right), \end{aligned} \tag{16}$$

where $\beta_T = \mathbb{E}\left[\|f^\star\|_k\right] + \sqrt{\lambda}\sqrt{2\ln T + 1 + \gamma_{T-1}} + \sqrt{2\ln T}$ when the noise level $\lambda = 1 + \frac{1}{T}$.

Important Remark on Notations and Alg.3. Throughout the proof, we abuse the notation x for an arbitrary sequence in $\mathcal{X}$ to represent its known d-dim embedding vector $\phi(x)$ unless we specifically clarify in text. Alg.3 rewrites Alg.1 on the embedding level (translating each step of Alg.1 into what happens to the embedding vectors) to reflect our reduction in analysis. Also, Alg.3 integrates the pseudo-code for updating f_t corresponding to (13) (Line 4-9) and Condition 3.1 by rewriting it as (17) (Line 22). Recall $\mathcal{D}$ is the local embedding space, K_t is the kernel matrix, λ is the std. of noise, m_t and k_t define the posterior GP, and $u(x_t)$ (x_t: embedding vector) corresponds to the noisy feedback $\tilde{F}(x_t)$ (x_t: original sequence in $\mathcal{X}$).

I.1 Tree Search and Proximal Iteration

Given the previous iterate x_{t-1} and an estimate f_t of the fitness, (17) ensures x_t is at least as good as the optimal solution of the locally constrained optimization:

$$\begin{aligned} &\max_x f_t(x), \\ s.t.\quad &\|x - x_{t-1}\|_2 \leq r. \end{aligned} \tag{18}$$

Algorithm 3 Revision of Alg. 1 with Monitored Posterior Updates

Input: total rounds T.
Initialize: $x_0 \in \mathcal{D}$, $k = 0, V_0 = \mathbf{0}$ and dataset $S_0 \leftarrow \emptyset$.
for $t = 1, \ldots, T$ **do**
  Kernel matrix monitoring:

$$V = K_{t-1} + \lambda I.$$

  if $\det(V)/\det(V_k) > 2$ **then**
    Update: $k \leftarrow k + 1, V_k \leftarrow V$.
    Thompson Sampling:

$$f_t(\cdot) \sim \mathcal{GP}(m_t, k_t).$$

  else
    $f_t(\cdot) \leftarrow f_{t-1}(\cdot)$.
  end if
  Tree Search: filter out x_t s.t.

$$f_t(x_t) \geq \max\{f_t(x) \mid \|x - x_{t-1}\| \leq r\}. \tag{17}$$

  Data collection: $S_t \leftarrow S_{t-1} \cup \{(x_t, u(x_t))\}$.
end for

Consider the proximal point of x_{t-1} under f_t with step size α:

$$\tilde{x}_t = \mathcal{P}_{\alpha, f_t}(x_{t-1}) := \arg\max_{x \in \mathcal{D}} f_t(x) - \frac{1}{2\alpha}\|x - x_{t-1}\|^2. \tag{19}$$

Note that the expression of (19) differs from the classic definition of proximal point by the constraint on $\mathcal{D}$. However, this is a minor difference as the objective $f_t(x) - \frac{1}{2\alpha}\|x - x_{t-1}\|^2$ is strongly concave on $\mathcal{D}$ and $\tilde{x}_t$ is in the interior of $\mathcal{D}$ under Assumption G.4. The optimal condition of (19) ensures

$$\tilde{x}_t - x_{t-1} \in \alpha \cdot \nabla f_t(\tilde{x}_t). \tag{20}$$

By setting a proper α, we are able to show x_t has more improvement in f_t value than $\tilde{x}_t$.

Proposition I.1. *When* $\alpha = \frac{r^2}{4B}$, $f_t(x_t) \geq f_t(\tilde{x}_t)$.

Proof. It's obvious that $f_t(x_t) \geq f_t(\tilde{x}_t)$ holds when $\tilde{x}_t \in \mathcal{B}(x_{t-1}, r) := \{x : \|x - x_{t-1}\|_2 \leq r\}$. If $\alpha = \frac{r^2}{4B}$, then for $\forall x \notin \mathcal{B}(x_{t-1}, r)$, $f_t(x) - \frac{1}{2\alpha}\|x - x_{t-1}\|^2 \leq f_t(x) - \frac{r^2}{2\alpha} = f_t(x) - 2B \leq -B \leq f_t(x_{t-1})$, thus $\tilde{x}_t \in \mathcal{B}(x_{t-1}, r)$. □

I.2 Regret Decomposition

Recall from the preliminaries of GP, $\{f_t\}$ is a sequence of online learning models of the ground truth $f^\star$ s.t.

$$\mathbb{E}[\max_{x \in \mathcal{D}} f_t(x)] = \mathbb{E}[\max_{x \in \mathcal{D}} f^\star(x)],$$

then the total regret of $\{x_t\}$ is decomposed as

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T} \max_{x \in \mathcal{D}} f^\star(x) - f^\star(x_t)\right] \\
=&\mathbb{E}\left[\sum_{t=1}^{T} \max_{x \in \mathcal{D}} f_t(x) - f_t(x_t)\right] + \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - f^\star(x_t)\right].
\end{aligned}$$

Re-indexing $\{x_t\}_{t=1}^T$ (Phase Index). Suppose f_t is updated at time steps $\{t_k\}_{k=0}^K$, where $t_1 = 1, t_2 = 1 + n_1, \cdots, t_K = 1 + \sum_{k=1}^{K-1} n_k, T = \sum_{k=1}^{K} n_k$ with $n_k, K \in \mathbb{N}^\star$. It indicates that the whole timeline $t \in [T]$ is partitioned into K phases of length $\{n_k\}_{k=1}^K$. The k-th phase is governed by one unchanging function f_{t_k}, i.e. $\forall t \in [T], k \in [K]$ s.t. $t_k \leq t < t_{k+1}$, we have $f_t = f_{t_k}$. To reflect which phase x_t is in, we assign the "phase index" $x_{k,i}, i \in [n_k]$ to $x_t, t \in [t_k, t_{k+1})$.

I.3 Regression Error $\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - f^\star(x_t)\right]$

Lemma I.2. *Under Assumption G.1 and G.2, the total number of updates satifies $K \leq O(\gamma_T)$ and*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - f^\star(x_t)\right] = O\left(\beta_T \sqrt{\lambda T \gamma_T}\right),$$

where $\beta_T = 2\sqrt{2\ln T + 1 + \gamma_{T-1}} + \sqrt{2\ln T}$ when $\lambda = 1 + \frac{1}{T}$ and γ_T is the maximum information gain.

I.4 Optimization Error $\mathbb{E}\left[\sum_{t=1}^{T} \max_{x\in\mathcal{D}} f_t(x) - f_t(x_t)\right]$

To stabilize evolution, we keep using a same model f_t throughout multiple rounds. Re-indexing $\{x_t\}$ to reflect the update of f_t , we have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T} \max_{x\in\mathcal{D}} f_t(x) - f_t(x_t)\right] \\
=&\mathbb{E}\left[\sum_{k=1}^{K}\sum_{i=1}^{n_k} \max_{x\in\mathcal{D}} f_{t_k}(x) - f_{t_k}(x_{k,i})\right].
\end{aligned} \tag{21}$$

Here $\{x_{k,i}\}_{i=1}^{n_k}$ is n_k-long segment of the whole trajectory seeking to maximize f_{t_k} in an evolutionary fashion. As shown in Proposition I.1, for every iterate x_t, there exists a proximal point iterate $\tilde{x}_t$ whose f_t value lower bounds $f_t(x_t)$, i.e. $\forall k \in [K]$,

$$\begin{aligned}
&\tilde{x}_{k,1} := x_{k,1}, \quad \tilde{x}_{k,i+1} := \mathcal{P}_{\alpha,f_{t_k}}(x_{k,i}), \forall i \in [n_k - 1]; \\
&f_{t_k}(\tilde{x}_{k,i}) \leq f_{t_k}(x_{k,i}).
\end{aligned}$$

Lemma I.3. *Under Assumption G.4 and Condition 3.1 and with $\alpha = \frac{r^2}{4B}$,*

$$\begin{aligned}
\sum_{k=1}^{K}\sum_{i=1}^{n_k} \max_{x\in\mathcal{D}} f_{t_k}(x) - f_{t_k}(\tilde{x}_{k,i}) &\leq O\left(K\left(B + \frac{R^2}{\alpha}\right)\right) \\
&= O\left(KB\left(1 + 4\frac{R^2}{r^2}\right)\right).
\end{aligned} \tag{22}$$

where the RHS also upper bounds $\mathbb{E}\left[\sum_{t=1}^{T} \max_{x\in\mathcal{D}} f_t(x) - f_t(x_t)\right]$ and by the assumption that the embedding map ϕ is Lipschitz, $R = LN$.

I.5 Final Regret Bound

Combining Lemma I.2 and I.3, the proof of Theorem 3.2 completes.

J Omitted Proofs in Appendix I

J.1 Proof of Lemma I.2

Proof. By our updating rule for the posterior

$$\begin{aligned}
2^K \leq &\frac{\det(\lambda I + K_{t_1-1})}{\det(\lambda I)} \cdot \frac{\det(\lambda I + K_{t_2-1})}{\det(\lambda I + K_{t_1-1})} \cdots \\
&\frac{\det(\lambda I + K_{t_K-1})}{\det\left(\lambda I + K_{t_{K-1}-1}\right)} \leq \det\left(I + \lambda^{-1} K_T\right).
\end{aligned} \tag{23}$$

Therefore,

$$K \leq \ln\left(\det\left(I + \lambda^{-1} K_T\right)\right) \leq 2\gamma_T. \tag{24}$$

Denote by $\|f^\star\|_k$ the norm of function $f^\star$ in the RKHS associated with kernel $k(\cdot,\cdot)$. From the Theorem 2 in [11], then with probability at least $1-\delta$, the following holds for all $x \in \mathbb{R}^d$ and $t \geq 1$:

$$
\begin{aligned}
&|m_t(x) - f^\star(x)| \\
\leq &\left(\|f^\star\|_k + \sqrt{\lambda}\sqrt{2\left(\ln\frac{1}{\delta}\right) + \ln\left(\det\left(\lambda I + K_{t-1}\right)\right)} \right) \sigma_t(x) \qquad (25) \\
\leq &\left(\|f^\star\|_k + \sqrt{\lambda}\sqrt{2\left(\ln\frac{1}{\delta}\right) + (t-1)\ln\lambda + \gamma_{t-1}} \right) \sigma_t(x), \qquad (26)
\end{aligned}
$$

where $m_t(x), \sigma_t^2(x)$ are the mean and variance of the posterior distribution of $f^\star(x)$ defined as in (10) and (11). Also, according to the sampling distribution of $f_t(x)$, which is $\mathcal{N}\left(m_t(x), \sigma_t(x)\right)$, with probability $1-\delta$, it holds for all $\{x_t\}_{t=1}^T$ (or $\{\{x_{k,i}\}_{i=1}^{n_k}\}_{k=1}^K$ indexed by phases) that

$$
\left|f_{t_k}\left(x_{k,i}\right) - m_{t_k}(x_{k,i})\right| \leq \sigma_{t_k}\left(x_{k,i}\right)\sqrt{\ln\left(\frac{T}{\delta}\right)}. \qquad (27)
$$

Let $\beta_T = \|f^\star\|_k + \sqrt{\lambda}\sqrt{2\left(\ln\frac{1}{\delta}\right) + (T-1)\ln\lambda + \gamma_{T-1}} + \sqrt{\ln\left(\frac{T}{\delta}\right)}$, then

$$
\begin{aligned}
&\sum_{k=1}^{K}\sum_{i=1}^{n_k} \left|f_{t_k}(x_{k,i}) - f^\star(x_{k,i})\right| \\
\leq &\beta_T \sum_{k=1}^{K}\sum_{i=1}^{n_k} \sigma_{t_k}\left(x_{k,i}\right) \\
\leq &\beta_T \sqrt{T\lambda \sum_{k=1}^{K}\sum_{i=1}^{n_k} \lambda^{-1}\sigma_{t_k}^2\left(x_{k,i}\right)} \\
\leq &\beta_T \sqrt{4T\lambda \sum_{k=1}^{K}\sum_{i=1}^{n_k} \frac{1}{2}\ln\left(1 + \lambda^{-1}\sigma_{t_k}^2\left(x_{k,i}\right)\right)}.
\end{aligned}
$$

According to Lemma 3 in [11], we have

$$
\frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \lambda^{-1}\sigma_t^2\left(x_t\right)\right) \leq \gamma_t. \qquad (28)
$$

Suppose $x_{k,i}$ corresponds to x_t under the phase indexing, consider the following ratio

$$
\frac{\sigma_{t_k}^2\left(x_{k,i}\right)}{\sigma_t^2\left(x_t\right)},
$$

where $t_k \leq t-1$ is the time step where the posterior Gaussian process was updated for the k-th time. Recall from (11),

$$
\sigma_t^2(x) = k_t\left(x,x\right) = k\left(x,x\right) - \mathbf{k_{t-1}}(x)^\top \left(K_{t-1} + \lambda I\right)^{-1}\mathbf{k_{t-1}}(x),
$$

where $k(x,x') = \phi(x)^\top\phi(x')$ with $\phi(x)$ being the possibly infinite dimensional feature of x. Hence by defining the $t \times \infty$ matrix $\Phi_t = \{\phi(x_1), \cdots, \phi(x_t)\}^\top$, we have $K_{t-1} = \Phi_{t-1}\Phi_{t-1}^\top$ and

$$
\begin{aligned}
\sigma_t^2(x) &= \phi(x)^\top\phi(x) - \phi(x)^\top\Phi_{t-1}^\top\left(K_{t-1} + \lambda I\right)^{-1}\Phi_{t-1}\phi(x) \\
&= \phi(x)^\top\left(I_{t-1} - \Phi_{t-1}^\top\left(K_{t-1} + \lambda I\right)^{-1}\Phi_{t-1}\right)\phi(x) \\
&= \lambda\phi(x)^T\left(\Phi_{t-1}^T\Phi_{t-1} + \lambda I\right)^{-1}\phi(x).
\end{aligned}
$$

Therefore,

$$\begin{aligned}\frac{\sigma_{t_k}^2(x_{k,i})}{\sigma_t^2(x_t)} &\leq \frac{\det\left(\lambda^{-1}\Phi_{t-1}^T\Phi_{t-1}+I\right)}{\det\left(\lambda^{-1}\Phi_{t_k-1}^T\Phi_{t_k-1}+I\right)} \\ &= \frac{\det\left(\lambda^{-1}K_{t-1}+I\right)}{\det\left(\lambda^{-1}K_{t_k-1}+I\right)} \\ &\leq \frac{\det\left(\lambda^{-1}K_{t_{k+1}-1}+I\right)}{\det\left(\lambda^{-1}K_{t_k-1}+I\right)} \leq 2, \end{aligned} \tag{29}$$

where we used $t_k \leq t \leq t_{k+1}$ and we end up with

$$\sum_{k=1}^{K}\sum_{i=1}^{n_k}|f_{t_k}(x_{k,i}) - f^\star(x_{k,i})| \leq O\left(\beta_T\sqrt{\lambda T\gamma_T}\right). \tag{30}$$

holds for any realization of $f^\star$ with probability $\frac{1}{\delta}$ when $\beta_T = \|f^\star\|_k + \sqrt{\lambda}\sqrt{2\left(\ln\frac{1}{\delta}\right)+(T-1)\ln\lambda+\gamma_{T-1}} + \sqrt{\ln\left(\frac{T}{\delta}\right)}$.

Taking $\delta = \frac{1}{T}$ and $\lambda = 1+\frac{1}{T}$ and taking expectation of 30 on both side, we have

$$\mathbb{E}\left[\sum_{k=1}^{K}\sum_{i=1}^{n_k}|f_{t_k}(x_{k,i}) - f^\star(x_{k,i})|\right] \leq O\left(\beta_T\sqrt{\lambda T\gamma_T}\right)$$

with $\beta_T = \mathbb{E}\left[\|f^\star\|_k\right] + \sqrt{\lambda}\sqrt{2\ln T+(T-1)\ln\lambda+\gamma_{T-1}} + \sqrt{2\ln T} \leq \mathbb{E}\left[\|f^\star\|_k\right] + 2\sqrt{2\ln T+1+\gamma_{T-1}} + \sqrt{2\ln T}$. □

J.2 Proof of Lemma I.3

Lemma J.1. *For a concave f with $z^\star$ being one of its maxima, a sequence of iterates z_t following the proximal point update*

$$z_{t+1} = \mathcal{P}_{\alpha,f}(z_t) \tag{31}$$

satisfies

$$\sum_t f(z^\star) - f(z_t) \leq \frac{1}{2\alpha}\|z_1 - z^\star\|^2. \tag{32}$$

Proof. Given $z_{t+1} = \mathcal{P}_{\alpha,f}(z_t)$, we have

$$z_{t+1} - z_t \in \alpha\cdot\partial f(z_{t+1}). \tag{33}$$

For any z, by the concavity of f

$$\begin{aligned} f(z) - f(z_{t+1}) &\leq \langle\partial f(z_{t+1}), z - z_{t+1}\rangle \\ &= \frac{1}{\alpha}\langle z_{t+1} - z_t, z - z_{t+1}\rangle \\ &= \frac{1}{2\alpha}\left[\|z - z_t\|^2 - \|z_{t+1} - z_t\|^2 - \|z - z_{t+1}\|^2\right] \\ &\leq \frac{1}{2\alpha}\left[\|z - z_t\|^2 - \|z - z_{t+1}\|^2\right]. \end{aligned}$$

Taking z as $z^\star$, we have

$$f(z^\star) - f(z_{t+1}) \leq \frac{1}{2\alpha}\left[\|z^\star - z_t\|^2 - \|z^\star - z_{t+1}\|^2\right].$$

□

By Lemma J.1, for $\forall k \in [K]$, under Assumption G.4, suppose $x_k^\star \in \arg\max_{x\in\mathcal{D}} f_{t_k}(x)$, we have

$$\sum_{i=1}^{n_k}\max_{x\in\mathcal{D}} f_{t_k}(x) - f_{t_k}(\tilde{x}_{k,i}) \leq \max_{x\in\mathcal{D}} f_{t_k}(x) - f_{t_k}(\tilde{x}_{k,1}) + \frac{1}{2\alpha}\|x_k^\star - \tilde{x}_{k,1}\|^2 \leq 2B + \frac{2R^2}{\alpha}.$$