Tree Search-Based Evolutionary Bandits for Protein Sequence Optimization

Jiahao Qiu[*1], **Hui Yuan**[*1], **Jinghong Zhang**[*2], **Wentao Chen**[3], **Huazheng Wang**[4], **Mengdi Wang**[1]

[1]Princeton University
[2]University of California San Diego
[3]MLAB Biosciences Inc
[4]Oregon State University
jq3984@princeton.edu, huiyuan@princeton.edu, zhangjinghong99@outlook.com, wentao.chen@mlabbiosciences.com, huazheng.wang@oregonstate.edu, 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 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, diversity-promoting, and able to find top designs using reasonably small mutation counts.

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 (Arnold 1998; Smith and Petrenko 1997; Winter et al. 1994). Traditional DE strategies are inherently screening (greedy search) strategies with limited ability to generate high-quality data for probing the 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 (Packer and Liu 2015; Yang, Wu, and Arnold 2019).

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. (2003) and followed by many, with increasing amounts of efforts utilizing *in silico* exploration and machine learning beyond experimental approaches (Yang, Wu, and Arnold 2019; Fannjiang and Listgarten 2020; Doppa 2021; Shin et al. 2021; Freschlin, Fahlberg, and Romero 2022; Wang et al. 2022; Sinai et al. 2020). More recent advances in large language models open up new opportunities for modeling and predicting protein functions and generalizing knowledge across protein domains (Rives et al. 2021; Shuai, Ruffolo, and Gray 2022; Nijkamp et al. 2022; Hsu et al. 2022; Elnaggar et al. 2020).

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 (Bryant et al. 2021; Shin et al. 2021), 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 (Rives et al. 2019) and ProGen2 (Nijkamp et al. 2022), 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

[*]These authors contributed equally.
Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

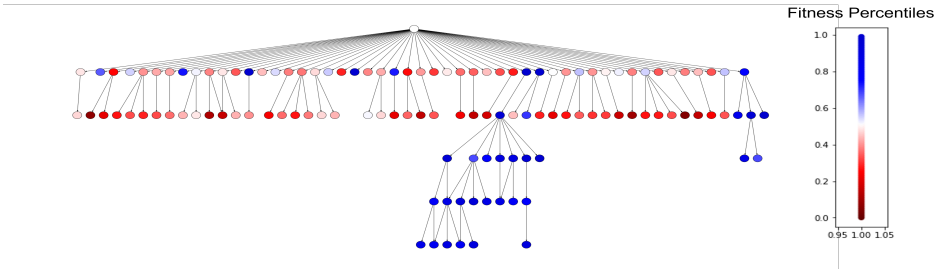

Figure 1: A Tree visualization of AAV screen dataset (Bryant et al. 2021), generated by starting from the wild-type and building the tree via downsampling children with an editing distance of 1 from the parent. The wet-lab screen initiates with a wild-type design sequence (root node), and in each round 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.

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 (Wu et al. 2019). A typical protein engineering workflow by producing purified proteins is limited to up to a few hundred sequences due to 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 know-how 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, (Bryant et al. 2021) 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 (Bryant et al. 2021) 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:

(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 (Bryant et al. 2021), TEM (Gonzalez and Ostermeier 2019) and AAYL49 antibody (Engelhart et al. 2022) 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 (Chen and Arnold 1991, 1993; Kuchner and Arnold 1997; Hibbert and Dalby 2005; Turner 2009; Packer and Liu 2015). 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 (Romero, Krause, and Arnold 2013). A preliminary version of this paper appeared at (Wang et al. 2022). See (Yang, Wu, and Arnold 2019) for a high-level survey of machine learning-assisted protein engineering, and see (Fox et al. 2007; Bedbrook et al. 2017) for more examples.

Search algorithms for protein sequence design. Researchers have tried out various machine learning-based methods for sequence optimization. Bayesian Optimization(BO)(Mockus 1989) is a classical method for optimizing protein sequences. We use the code developed by Sinai et al. (2020) who uses an ensemble of models for BO as one of the baselines. LaMBO(Stanton et al. 2022) is also a BO-based algorithm that supports both single-objective and multi-objective. Design by adaptive sampling (DbAS; Brookes and Listgarten (2018)) and conditioning by adaptive sampling (CbAS; Brookes, Park, and Listgarten (2019)) use a probabilistic framework. DyNA PPO(Angermueller et al. 2020) uses proximal policy optimization for sequence design. PEX MufacNet(Ren et al. 2022) is a local search method based on the principle of proximal optimization.

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 (Yuan et al. 2022) 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) Abbasi-Yadkori, Pál, and Szepesvári (2011) and Thompson Sampling (TS) Russo and Van Roy (2014), 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 $\tilde{O}\left(d\sqrt{T}\right)$(Auer 2002; Li et al. 2010; Abbasi-Yadkori, Pál, and Szepesvári 2011; Li et al. 2010; Russo and Van Roy 2014), with extensions to reinforcement learning (Yang and Wang 2020, 2019; Li et al. 2022) and sparsity-aware or decentralized settings (Hao, Lattimore, and Wang 2020; Li et al. 2022). 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, (Srinivas et al. 2009) 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 (Chowdhury and Gopalan 2017) 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 (Zhou, Li, and Gu 2020; Zhang et al. 2020; Jacot, Gabriel, and Hongler 2018; Xu et al. 2020). Building upon the neural tangent kernel (NTK) technique to analyze deep neural network (Jacot, Gabriel, and Hongler 2018), NeuralUCB (Zhou, Li, and Gu 2020) can be viewed as a direct extension of kernel bandits (Srinivas et al. 2009). Zhang et al. (2020) proposed the Thompson Sampling version of neural bandits. Xu et al. (2020) 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 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 $\mathcal{X}$, screen and collect data 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 the embedding map $\phi(x)$ helps to 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 in Algorithm 1. Detailed implementation of the subroutine TREESEARCH (Alg. 2) is delayed till Section 4.1 and 5.

Algorithm 1: Meta algorithm

1: **Input:** wildtype x^{wt}, total rounds T
2: **Initialization:** Add x^{wt} to active node set $\mathcal{A}_0$ as root node.
3: **for** $t = 1, 2, ...T$ **do**
4: Update bandit model: return a scoring function $F_t(\cdot) := f_t(\phi(\cdot))$.
5: Tree search by Alg. 2: $\mathcal{A}_t \leftarrow \text{TREESEARCH}(\mathcal{A}_{t-1}, F_t)$.
6: Query the fitness $F(x)$ for some (or all) x's in $\mathcal{A}_t$ and append $\{(\phi(x), \tilde{F}(x))\}$ to the dataset.
7: **end for**

Analysis of simple tree search Motivated by the practical observation that high-potential variants usually appear within a small count of mutations from a wildtype and the practical consideration that 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).

Bayesian regret Given the search region $\bar{\mathcal{X}}$, we measure the performance of an algorithm by its Bayesian regret defined as follows.

Definition 3.1. Suppose the algorithm finds a series of protein sequences $\{x_t\}_{t=1}^T$ within $\bar{\mathcal{X}}$, then its accumulated Bayesian regret is defined as

$$\text{BayesRGT}(T) = \mathbb{E}\left[\sum_{t=1}^{T}\left(\max_{x\in\bar{\mathcal{X}}} F(x) - F(x_t)\right)\right], \quad (2)$$

where $\mathbb{E}$ is taken over the prior distribution of $f^\star$ (which we will assume to be a Gaussian process in Section 3.3), and taken over other randomness in finding $\{x_t\}_{t=1}^T$.

Simple tree search In order to obtain a basic theoretical understanding of the tree search-based bandit learning process, we consider a *simplified* mathematical abstraction of Alg. 1: suppose in each round t, protein x_t is filtered out through the subroutine $\text{TREESEARCH}(\mathcal{A}_{t-1}, F_t)$, and $\{x_t\}_{t=1}^T$ satisfies the following condition. And it only collects $\{(\phi(x_t), \tilde{F}(x_t))\}$ in the active set for updating f_t.

Condition 3.2. *There exists $r > 0$ such that each iteration of tree search is able to find a sequence that has equal or better F_t value than the local maximum within radius r in the embedding space, i.e.,*

$$F_t(x_t) \geq \max\{F_t(x) \mid x : \|\phi(x) - \phi(x_{t-1})\| \leq r\}. \quad (3)$$

This condition shows that the local search method makes sufficient improvement w.r.t. F_t per iteration. In practice, this condition can be satisfied by tuning parameters and stopping conditions of the tree search subroutine. To understand this, when ϕ is Lipschitz, with notation $\phi(\bar{\mathcal{X}}) := \{\phi(x) : x \in \bar{\mathcal{X}}\}$, we have $\phi(\mathcal{X}) \subset \mathcal{D} := \{\phi(x) : \|\phi(x) - \phi(x^{wt})\| \leq R := L_\phi N\}$, $\mathcal{D}$ is the local region around $\phi(x^{wt})$ in the embedding space. Thus in practice, if a good embedding map ϕ successfully captures the latent structure of $\mathcal{X}$ such that $\phi(\bar{\mathcal{X}})$ spans well in $\mathcal{D}$, then Condition 3.2 is satisfied whenever TREESEARCH sufficiently exhausts the local variants around x_{t-1} in the discrete sequence space. 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 provide a Bayesian regret theory for the proposed meta algorithm, under some necessary assumptions and simplifications. We assume a GP prior-posterior modelling of $f^\star$ and f_t's, as well as some local properties of f_t's. The basic theoretical understanding of the tree search-based bandit learning process we obtained in this section may have implications for the broader settings beyond our assumptions/simplifications.

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 (Seeger 2004), especially in classic bandit optimization (Srinivas et al. 2009; Chowdhury and Gopalan 2017).

Assumption 3.3. $f^\star : \mathbb{R}^d \to R$ is a sample from the Gaussian process $\mathcal{GP}(0, k(\cdot, \cdot))$ (with known kernel function $k(\cdot, \cdot)$) as priori, i.e.

$$f^\star(\phi(x)) \sim \mathcal{GP}\left(0, k(\phi(x), \phi(x'))\right). \quad (4)$$

Assumption 3.4. For any sequence x, querying its fitness $F(x)$ returns a noisy feedback $\tilde{F}(x)$ corrupted by a Gaussian, i.e.

$$\tilde{F}(x) = f^\star(\phi(x)) + \epsilon, \quad (5)$$

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

To get a sharper regret bound, we sample f_t from the posterior Gaussian Process with monitoring. For the subsequent theory, we also need some assumptions on the local properties of f_t's. Recall $\mathcal{D} := \{\phi(x) : \|\phi(x) - \phi(x^{wt})\| \leq R := L_\phi N\}$ is the local region surrounding $\phi(x^{wt})$ in $\mathbb{R}^d$.

Assumption 3.5. At each time step t, f_t is bounded in $\mathcal{D}$. For $\forall t, z \in \mathcal{D}$,

$$|f_t(z)| \leq B. \quad (6)$$

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

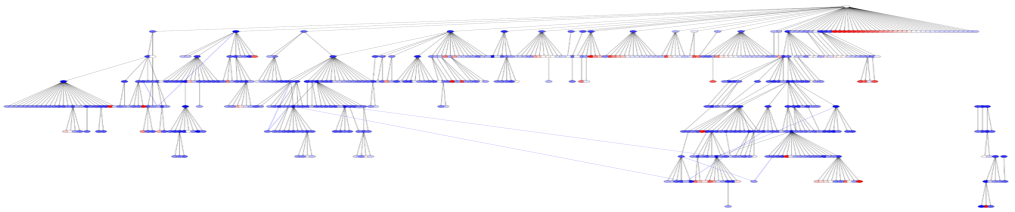

Figure 2: Visualization of tree search process of Algorithm 1 using the AAV oracle. The search initiates with a wild-type sequence (the root note) and in each round, we choose 100 sequences generated from the last round according to scores derived from UCB and TS by single mutation which leads to a node 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.

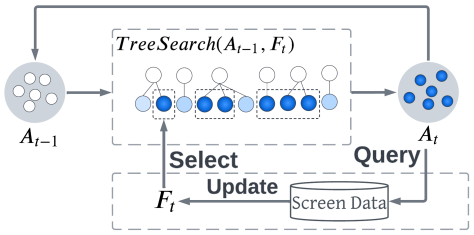

Figure 3: Diagram of the meta algorithm (Alg. 1)

Next in Theorem 3.6, Alg. 1 is proven to explore the local sequence space via tree search and bandit learning, while attaining low regret.

Theorem 3.6. *Under Assumption 3.3, 3.4, 3.5 and Condition 3.2, Alg.1 updates f_t for $O(\gamma_T)$ times its Bayesian regret is bounded by*

$$\mathrm{BayesRGT}(T) = O\left(\beta_T\sqrt{\lambda T\gamma_T} + B\gamma_T\left(1+\frac{4L_\phi^2 N^2}{r^2}\right)\right), \tag{7}$$

where $\beta_T = O\left(\mathbb{E}\left[\|f^\star\|_k\right] + \sqrt{d\ln T} + \sqrt{\gamma_{T-1}}\right)$. γ_T is the information gain, r is inherit from Condition 3.2 and N is the depth of tree search.

Rate of regret The highest order term in (7) is of $\tilde{O}\left(\gamma_T\sqrt{T} \vee \sqrt{d\gamma_T T}\right)$, which matches the results by (Srinivas et al. 2009; Chowdhury and Gopalan 2017) studying GP bandits. (7) 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 (Yuan et al. 2022) by a factor of $\sqrt{d}$, benefiting from the stability brought by f_t's rare switching. Here γ_T is closely related to the "effective dimension" of the chosen kernel, which is a natural and common complexity metric for online exploration.

Novelty and significance compared to classical results of bandits and Bayesian optimization. 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 maximum of a surrogate function within the search region. This is far from the protein design practice where solutions with large surrogate values are approached by search: looking for the argmax of $\hat{f}_t$ 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. 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 (Romero, Krause, and Arnold 2013), which supports our assumptions here. Further, GP can represent any kernel function space, which together with the modern deep learning theory (Jacot, Gabriel, and Hongler 2018) imply a powerful theoretical approximation to neural networks used in practice.

4 Full Algorithm

In this section, we present the full algorithm.

4.1 Tree Search Heuristics

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:

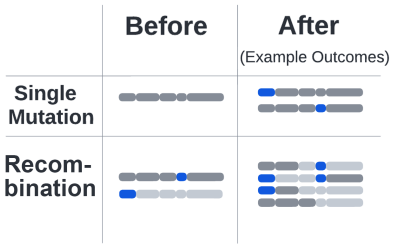

Figure 4: A demonstration of mutation and recombination. 1

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.

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 $\mathcal{A}$

4.2 Bandit Exploration

We use two classic exploration methods for scoring, Upper Confidence Bound (UCB) (Li et al. 2010) and Thompson Sampling (TS; Russo and Van Roy (2014)), to enable our algorithm to consider the potential of each node and look ahead.

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. (2022) and Sinai et al. (2020), 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 (Bryant et al. 2021), TEM (Gonzalez and Ostermeier 2019) and AAYL49 antibody (Engelhart et al. 2022). Experiments results show that the tree search-based bandit outperforms existing baselines and leads to several interesting observations.

5.1 Datasets and Setup

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 (Rao et al. 2019) with a CNN model and the AAYL49 oracle is a downstream task from the pre-trained TAPE transformer model (Rao et al. 2019). 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.

In particular, we use datasets of Adeno-associated virus 2 capsid protein (AAV) (Bryant et al. 2021) which aimed for viral viability with search space 20^{28}; TEM-1 β-Lactamase (TEM) (Gonzalez and Ostermeier 2019) which aimed for thermodynamic stability with search space 20^{286}; and, Anti-SARS-CoV-2 antibody (AAYL49) (Engelhart et al. 2022) which aimed for binding affinity with search space of 20^{118}.

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 (Ren et al. 2022), Adalead (Sinai et al. 2020), Bayesian Optimization (BO) implemented by Sinai et al. (2020), DbAS (Brookes and Listgarten 2018) implemented by Sinai et al. (2020) and LaMBO(Stanton et al. 2022) 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 test CbAS (Brookes, Park, and Listgarten 2019) but choose not to report it, because it performs almost the same as DbAS, consistent with results Ren et al. (2022). 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 (Angermueller et al. 2020) as our baseline due to TensorFlow incompatibility.

For tree search-based algorithms, we use a neural network and follow (Xu et al. 2020)'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).

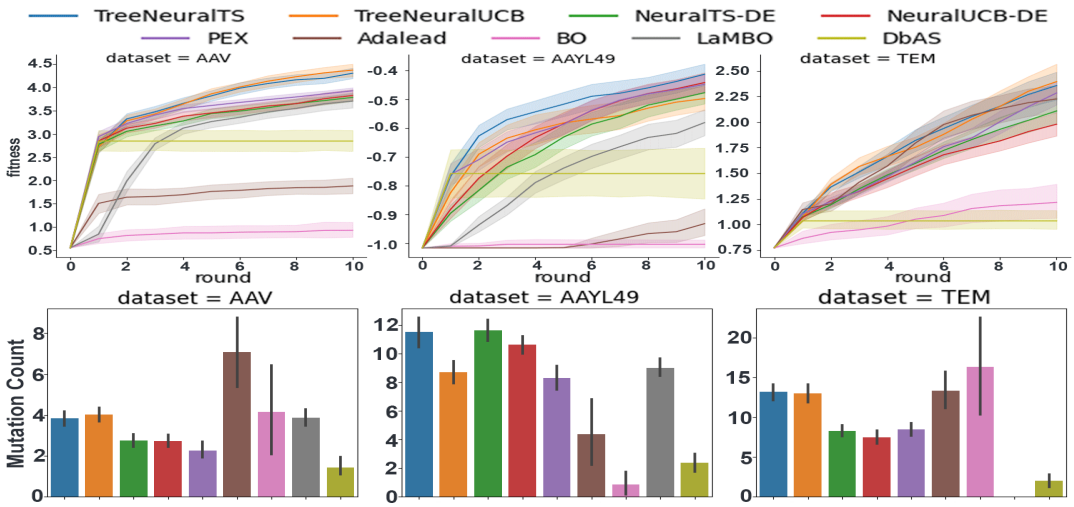

Figure 5: Learning curves of algorithms with comparison to baselines, tested over three datasets.

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 (Rao et al. 2019) 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. 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 built by Thomas et al. (2022), the results are consistent with our oracle and the tree search method outperforms other baselines.

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 built by Thomas et al. (2022) using the same setting. The results are 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 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.

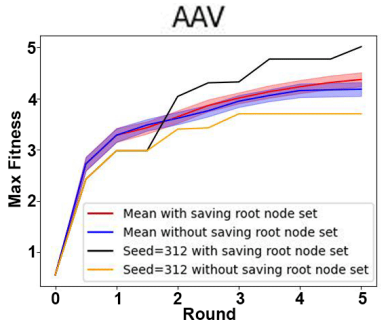

Figure 6: Effect of keeping parent nodes in active set.

6 Limitations

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. We do not allow using our models in production without authorization due to potential negative social impacts. More discussion on limitations is available in Appendix.

7 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.

Acknowledgments

We extend our gratitude to MLAB Biosciences Inc. for their assistance in this work by providing abundant computing resources. We also extend our gratitude to Peiru Xu, who contributed to the implementation of several baseline algorithms during her internship at MLAB Biosciences.

References

Abbasi-Yadkori, Y.; Pál, D.; and Szepesvári, C. 2011. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24: 2312–2320.

Angermueller, C.; Dohan, D.; Belanger, D.; Deshpande, R.; Murphy, K.; and Colwell, L. 2020. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*.

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

Auer, P. 2002. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research*, 3: 397–422.

Bedbrook, C. N.; Yang, K. K.; Rice, A. J.; Gradinaru, V.; and Arnold, F. H. 2017. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS computational biology*, 13(10): e1005786.

Brookes, D.; Park, H.; and Listgarten, J. 2019. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, 773–782. PMLR.

Brookes, D. H.; and Listgarten, J. 2018. Design by adaptive sampling. *arXiv preprint arXiv:1810.03714*.

Bryant, D. H.; Bashir, A.; Sinai, S.; Jain, N. K.; Ogden, P. J.; Riley, P. F.; Church, G. M.; Colwell, L. J.; and Kelsic, E. D. 2021. Deep diversification of an AAV capsid protein by machine learning. *Nature Biotechnology*, 39(6): 691–696.

Chen, K.; and Arnold, F. H. 1991. Enzyme engineering for nonaqueous solvents: random mutagenesis to enhance activity of subtilisin E in polar organic media. *Bio/Technology*, 9(11): 1073–1077.

Chen, K.; and Arnold, F. H. 1993. 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.

Chowdhury, S. R.; and Gopalan, A. 2017. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, 844–853. PMLR.

Doppa, J. R. 2021. Adaptive experimental design for optimizing combinatorial structures. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 4940–4945.

Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rihawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; Bhowmik, D.; and Rost, B. 2020. ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing.

Engelhart, E.; Emerson, R.; Shing, L.; Lennartz, C.; Guion, D.; Kelley, M.; Lin, C.; Lopez, R.; Younger, D.; and Walsh, M. E. 2022. A dataset comprised of binding interactions for 104,972 antibodies against a SARS-CoV-2 peptide. *Scientific Data*, 9(1): 653.

Fannjiang, C.; and Listgarten, J. 2020. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33: 12945–12956.

Fox, R.; Roy, A.; Govindarajan, S.; Minshull, J.; Gustafsson, C.; Jones, J. T.; and Emig, R. 2003. Optimizing the search algorithm for protein engineering by directed evolution. *Protein engineering*, 16(8): 589–597.

Fox, R. J.; Davis, S. C.; Mundorff, E. C.; Newman, L. M.; Gavrilovic, V.; Ma, S. K.; Chung, L. M.; Ching, C.; Tam, S.; Muley, S.; et al. 2007. Improving catalytic function by ProSAR-driven enzyme evolution. *Nature biotechnology*, 25(3): 338–344.

Freschlin, C. R.; Fahlberg, S. A.; and Romero, P. A. 2022. Machine learning to navigate fitness landscapes for protein engineering. *Current Opinion in Biotechnology*, 75: 102713.

Gonzalez, C. E.; and Ostermeier, M. 2019. Pervasive pairwise intragenic epistasis among sequential mutations in TEM-1 β-lactamase. *J. Mol. Biol.*, 431(10): 1981–1992.

Hao, B.; Lattimore, T.; and Wang, M. 2020. High-dimensional sparse linear bandits. *Advances in Neural Information Processing Systems*, 33: 10753–10763.

Hibbert, E. G.; and Dalby, P. A. 2005. Directed evolution strategies for improved enzymatic performance. *Microbial Cell Factories*, 4(1): 1–6.

Hsu, C.; Nisonoff, H.; Fannjiang, C.; and Listgarten, J. 2022. Learning protein fitness models from evolutionary and assay-labeled data. *Nature Biotechnology*, 40(7): 1114–1122.

Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Kuchner, O.; and Arnold, F. H. 1997. Directed evolution of enzyme catalysts. *Trends in biotechnology*, 15(12): 523–530.

Li, C.; Wang, H.; Wang, M.; and Wang, H. 2022. Communication efficient distributed learning for kernelized contextual bandits. *Advances in Neural Information Processing Systems*, 35: 19773–19785.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, 661–670. ACM.

Mockus, J. 1989. Bayesian Approach to Global Optimization: Theory and Applications.

Nijkamp, E.; Ruffolo, J.; Weinstein, E. N.; Naik, N.; and Madani, A. 2022. ProGen2: Exploring the Boundaries of Protein Language Models.

Packer, M. S.; and Liu, D. R. 2015. Methods for the directed evolution of proteins. *Nature Reviews Genetics*, 16(7): 379–394.

Rao, R.; Bhattacharya, N.; Thomas, N.; Duan, Y.; Chen, X.; Canny, J.; Abbeel, P.; and Song, Y. S. 2019. Evaluating

Protein Transfer Learning with TAPE. In *Advances in Neural Information Processing Systems*.

Ren, Z.; Li, J.; Ding, F.; Zhou, Y.; Ma, J.; and Peng, J. 2022. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning*, 18520–18536. PMLR.

Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; and Fergus, R. 2019. Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences. *PNAS*.

Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; and Fergus, R. 2021. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15): e2016239118.

Romero, P. A.; Krause, A.; and Arnold, F. H. 2013. Navigating the protein fitness landscape with Gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3): E193–E201.

Russo, D.; and Van Roy, B. 2014. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4): 1221–1243.

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

Shin, J.-E.; Riesselman, A. J.; Kollasch, A. W.; McMahon, C.; Simon, E.; Sander, C.; Manglik, A.; Kruse, A. C.; and Marks, D. S. 2021. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1): 1–11.

Shuai, R. W.; Ruffolo, J. A.; and Gray, J. J. 2022. Generative language modeling for antibody design. *bioRxiv*.

Sinai, S.; Wang, R.; Whatley, A.; Slocum, S.; Locane, E.; and Kelsic, E. 2020. AdaLead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint*.

Smith, G. P.; and Petrenko, V. A. 1997. Phage display. *Chemical reviews*, 97(2): 391–410.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Stanton, S.; Maddox, W.; Gruver, N.; Maffettone, P.; Delaney, E.; Greenside, P.; and Wilson, A. G. 2022. Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 20459–20478. PMLR.

Thomas, N.; Agarwala, A.; Belanger, D.; Song, Y. S.; and Colwell, L. J. 2022. Tuned Fitness Landscapes for Benchmarking Model-Guided Protein Design. *bioRxiv*.

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

Wang, C.; Kim, J.; Cong, L.; and Wang, M. 2022. Neural Bandits for Protein Sequence Optimization. In *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, 188–193. IEEE.

Winter, G.; Griffiths, A. D.; Hawkins, R. E.; and Hoogenboom, H. R. 1994. Making antibodies by phage display technology. *Annual review of immunology*, 12(1): 433–455.

Wu, Z.; Kan, S. B. J.; Lewis, R. D.; Wittmann, B. J.; and Arnold, F. H. 2019. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18): 8852–8858.

Xu, P.; Wen, Z.; Zhao, H.; and Gu, Q. 2020. Neural Contextual Bandits with Deep Representation and Shallow Exploration. *CoRR*, abs/2012.01780.

Yang, K. K.; Wu, Z.; and Arnold, F. H. 2019. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8): 687–694.

Yang, L.; and Wang, M. 2019. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, 6995–7004. PMLR.

Yang, L.; and Wang, M. 2020. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, 10746–10756. PMLR.

Yuan, H.; Ni, C.; Wang, H.; Zhang, X.; Cong, L.; Szepesvári, C.; and Wang, M. 2022. Bandit Theory and Thompson Sampling-Guided Directed Evolution for Sequence Optimization. *arXiv preprint arXiv:2206.02092*.

Zhang, W.; Zhou, D.; Li, L.; and Gu, Q. 2020. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*.

Zhou, D.; Li, L.; and Gu, Q. 2020. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, 11492–11502. PMLR.