Bandit Theory and Thompson Sampling-Guided Directed Evolution for Sequence Optimization

Hui Yuan[1], Chengzhuo Ni[2], Huazheng Wang[3], Xuezhou Zhang[4], Le Cong[5], Csaba Szepesvári[6,7], and Mengdi Wang[7,8]

[1,2,4,8]Department of Electrical and Computer Engineering, Princeton University
[3]School of Electrical Engineering and Computer Science, Oregon State University
[5]Department of Pathology and Department of Genetics, Stanford University
[6]Department of Computing Science, University of Alberta
[7]DeepMind *†

Abstract

Directed Evolution (DE), a landmark wet-lab method originated in 1960s, enables discovery of novel protein designs via evolving a population of candidate sequences. Recent advances in biotechnology has made it possible to collect high-throughput data, allowing the use of machine learning to map out a protein's sequence-to-function relation. There is a growing interest in machine learning-assisted DE for accelerating protein optimization. Yet the theoretical understanding of DE, as well as the use of machine learning in DE, remains limited. In this paper, we connect DE with the bandit learning theory and make a first attempt to study regret minimization in DE. We propose a Thompson Sampling-guided Directed Evolution (TS-DE) framework for sequence optimization, where the sequence-to-function mapping is unknown and querying a single value is subject to costly and noisy measurements. TS-DE updates a posterior of the function based on collected measurements. It uses a posterior-sampled function estimate to guide the crossover recombination and mutation steps in DE. In the case of a linear model, we show that TS-DE enjoys a Bayesian regret of order $\widetilde{O}(d^2\sqrt{MT})$, where d is feature dimension, M is population size and T is number of rounds. This regret bound is nearly optimal, confirming that bandit learning can provably accelerate DE. It may have implications for more general sequence optimization and evolutionary algorithms.

1 Introduction

Protein engineering means to design a nucleic acids sequence for maximizing a utility function that measures certain fitness or biochemical/enzymatic properties, i.e., stability, binding affinity, or catalytic activity. Due to the combinatorial sequence space and lack of knowledge about the sequence-to-function map, engineering and identifying optimal protein designs were a quite daunting task. It is only until recently that synthesis of nucleic acid sequences and measurement of protein function became reasonably scalable [41, 57], allowing rational optimization or directed evolution

*Authors' emails are: {huiyuan, cn10, xz7392, mengdiw}@princeton.edu, huazheng.wang@oregonstate.edu, congle@stanford.edu, szepesva@ualberta.ca.

†Mengdi Wang acknowledges support by NSF grants DMS-1953686, IIS-2107304, CMMI-1653435, and ONR grant 1006977. Le Cong acknowledges support by NIH grant R35-HG011316, and Donald and Delia Baxter Foundation grant, and NSF grant DMS-1953686.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

of protein designs. Nonetheless, because of the complex landscape of protein functions and the bottleneck of wet-lab experimentation, this remains a very difficult problem.

Directed evolution (DE), one of the top molecular technology breakthroughs in the past century, demonstrates human's ability to engineer proteins at will. DE is a method for exploring new protein designs with properties of interest and maximal utility, by mimicking the natural evolution. It works by artificially evolving a population of variants, via mutation and recombination, while constantly selecting high-potential variants [8, 9, 33, 25, 49, 41]. The development of directed evolution methods was honored in 2018 with the awarding of the Nobel Prize in Chemistry to Frances Arnold for evolution of enzymes, and George Smith and Gregory Winter for phage display [4, 47, 53]. See Figure 1.1 for illustrations of mutation and crossover recombination.

DE practitioners' major considerations center on cost and data quality. First, the ability to synthesize and mutate new biological sequences have been exponentially improved thanks to synthetic chemistry advances. Second, given a population of sequences S, selecting and identifying the set of optimal sequences is straightforward, using low-cost parallel sequencing which works well with pooled selection assays. Third, using pooled measurement to evaluate the average value of protein function (mean fitness) over a population S is generally easy, as such bulk measurements is low-cost and high-quality. Finally, querying $f(x)$ for a given x is often expensive and time-consuming, and the cost adds up quickly if many queries are needed. It can be desirable to perform this procedure in small-scale batches to optimize time and resource consumption.

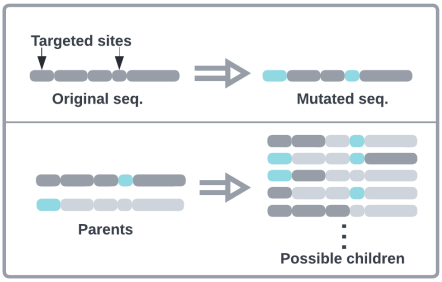

Figure 1.1: Illustration of mutation and crossover recombination. Mutating a sequence means to replace a targeted or random entry (site) by a random or designated value. Recombination involves two or multiple sequences. For example, parent sequences can crossover, exchange subsequences and generate children.

Such difficulties have motivated scientists to apply machine learning approaches to accelerate DE, beginning with Fox et al. [16] and followed by many. Recent development of directed evolution have increasingly utilized *in silico* exploration and machine learning beyond experimental approaches [57, 15, 12, 45, 18, 50, 46]. While these attempts have proved to be successful in simulation and sometimes in real experiments, little is known about the statistical theory of DE.

In this paper, a primary objective is to bridge the directed evolution process with bandit learning theory. In particular, we want to express machine learning-assisted DE as a bandit optimization process, with a theoretical justification. Further, we aim to understand how a machine learning model, as simple as linear, can accelerate DE and reduce the overall cost of evaluation. Specifically, we propose a Bayesian bandit model for DE, namely the Thompson Sampling-guided Directed Evolution framework, which combines posterior model sampling with directed mutation and recombination. The theoretical analysis shows that the crossover selection mimics an optimization iteration, and the optimization progress is proportional to a level of population diversity. In the case of the linear model, we establish a Bayesian regret bound $\widetilde{O}(d^2\sqrt{MT})$[3] that depends polynomially on feature dimension d[4], and optimally in batch size M and time steps T. We finally harmonize our theoretical analysis with a set of simulation and real-world experiment.

Important Remark The scope of this work is to provide a simplified mathematical model and basic theoretical understanding of an evolutionary-based process that is common in directed evolution. We emphasize that our framework is a theoretical simplification, assuming linear objective over a hy-

[3] $\widetilde{O}(\cdot)$ ignores the logarithmic terms.

[4] Our TS-DS regret has two extra $\sqrt{d}$ factors compared to the classic result that the optimal regret for linear bandits is of $\widetilde{O}(d\sqrt{MT})$. One $\sqrt{d}$ comes from our problem setting where the l_2 norm of each action is $O(\sqrt{d})$. Another factor of $\sqrt{d}$ is due to the evolutionary nature of DE. See the remark in §5.2 for more details.

percube. In real-world experimental systems, one needs to consider prior knowledge about the system and applying any machine learning method would require careful calibration and customization.

2 Related work

From a theoretical perspective, our analysis is related to the literature on evolutionary algorithms and linear bandits.

Evolutionary algorithm. The success of DE motivated a large body of works on evolutionary algorithms for optimization. Evolutionary algorithm (EA) [6] is a large class of randomized optimization algorithms, based on the heuristic of mimicking natural evolution. Despite many variants, a typical EA usually maintains a population of solutions and improves the solutions by alternating between reproduction step which produces new offspring solutions, and selection step where solutions are evaluated by the objective function and only the good ones are saved to the next round. Theoretical understandings of EA are focusing on specific EAs, among which the most well-studied setting is $(1+1)$-EA, with parent population size and offspring population size are both 1 to optimize linear objective function on the Boolean space $\{0,1\}^d$, see [14, 24, 27, 28, 35, 55]. EA analysis focuses on optimization and reducing the running time instead of minimizing total regret as in bandit theory. There are other results on population based EAs, such as $(1+\lambda)$-EA [11, 19], $(\mu+1)$-EA [54] and the most general $(\mu+\lambda)$-EA, where μ and λ represent the parent population size and the offspring population size respectively. However, this group of works only adopted mutation. The understanding of the role played by recombination in evolutionary algorithms was left as blank in the $(\mu+\lambda)$-EA framework, while our paper provides a population-based regret minimization analysis with both mutation and recombination.

There are a few works [30, 29, 51, 32] studying EAs with recombination (which are also called genetic algorithms (GAs)). However, their algorithms and analysis are tailored to artificial test objectives and the results are not able to generalize even to linear objectives. Recently, the running time analysis of some natural EAs with recombination has been conducted [39, 40], but still their results are constrained under specific objectives such as ONEMAX and JUMP. We refer readers to the book by [59] for a more comprehensive review of EA.

Linear bandits. Bandit is a powerful framework formulating the sequential decision making process under uncertainties. Under this framework, linear bandits is a central and fruitful branch where in each round a learner makes her decision and receives a noisy reward with its mean value modelled by a linear function of the decision, aiming to maximize her total reward (or minimize total regret equivalently) over multiple rounds [5, 36, 1], with extension to sparse linear bandits [23] and linear MDP [58]. In the same spirit, the process evolving a population of genetic sequences to maximize a linear utility over the evolution trajectory, while getting access to noisy utility values through evaluating sequences along the way, can be mathematically formulated from the perspective of linear bandits. One of the main solutions in linear bandits is the upper confidence bound-based (UCB) strategy represented by LinUCB [36], where the learner makes decision according to upper confidence bounds of the estimated reward and the accumulated regret is proven to be $\widetilde{O}\left(d\sqrt{T}\right)$. A similar strategy is optimism in the face of uncertainty (OFU) principle in Abbasi-Yadkori et al. [1]. The other approach is the Thompson Sampling (TS) strategy, which randomizes actions on the basis of their probabilities to be optimal. Russo and Van Roy [43] proved the Bayesian regret of TS algorithm is also of order $\widetilde{O}\left(d\sqrt{T}\right)$. And there are more results on the regret of TS(-like) algorithms solving linear bandits in the frequentist view [3, 2, 21, 3]. We also refer readers to the book by [34] for a delicate review of bandit theory.

Non-evolutionary methods for protein sequence design. Though our framework applies to an evolution-based DE process, there exist many other methods that are not evolution-based. Protein engineering is a rich field and it is not restricted to methods that are based on mutagenesis and recombination. Protein sequence engineering constantly evolves as new bio-technologies keep emerging. For example, new biotechnology makes it possible to synthesize specific variants and operate on the combinatorial space likewise with high-throughput method, and this allows directly applying a Gaussian process bandit algorithm [42]. See [57] for a high-level survey of this active

area of research, and see [17, 7] for more examples. This active and exciting field brings many new opportunities for machine learning.

Remark. It is important to note that our problem is *not* a multi-armed bandit problem. In bandits, one can choose actions freely from the full action set. However, in biological experiments, it is expensive to synthesize a new protein design sequence out of thin air. Instead, mutation and recombination are used to generate new designs easily at a low cost. Thus our algorithm can only guide the selection step in the DE process. Its regret is not directly comparable with the regret of multi-arm bandits. To the best of our knowledge, this is the first work that studies the bandit theory and regret bound of mutation and recombination-enabled DE.

3 Bandit model for directed evolution

3.1 Process overview

We illustrate the Thompson Sampling-guided Directed Evolution (TS-DE) process in Figure 3.1. A population S_t at time t consists of M candidate sequences. It evolves via mutation, crossover recombination, selection, and function evaluation to the next generation S_{t+1}. The mutation and crossover selection are guided using a learnt function $f_{\tilde{\theta}_t}$, in order to filter out unwanted candidates and keep only a small batch for costly evaluation. Collected data are fed into a Thompson Sampling module for posterior update of $f_{\tilde{\theta}_t}$. Full details of the mutation, crossover selection, and Thompson Sampling modules will be given in Section 4.

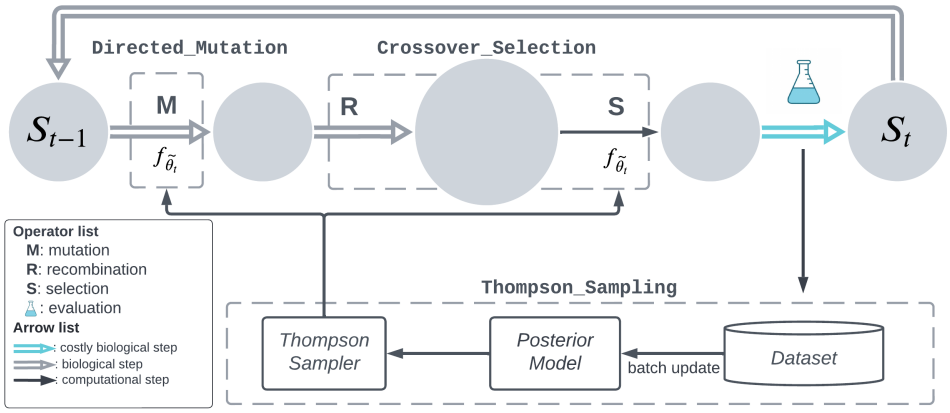

Figure 3.1: **Thompson sampling-guided directed evolution**

3.2 Motif feature, utility model, recombination and mutation operators

A genetic sequence comprises of functional motifs, i.e., functional subsequences that may encode particular features of protein, also known as protein motifs [37, 48, 10]. Such genetic motifs are known to be "evolutionarily conserved", in the sense that they tend to evolve as units, under mutation and recombination.

Suppose a genetic sequence seq is made up of d genetic motifs, given by $seq = (seq_{(1)}, seq_{(2)}, \cdots, seq_{(d)})$. Machine learning models for protein utility prediction are often based on motif features [56, 10, 38]. Let $\mathcal{X}$ be the space of genetic sequences of interest. We assume that a binary motif feature map is given, defined as follows.

Definition 3.1 (Binary Motif Feature Embedding)**.** Let ϕ be the genetic motif feature map given by:

$$\phi : \mathcal{X} \to \{0,1\}^d, \quad \phi(seq) := (\phi_1(seq_{(1)}), \cdots, \phi_d(seq_{(d)})) \tag{3.1}$$

such that at each dimension i, $\phi_i(seq_{(i)})$ is a binary feature of motif $seq_{(i)}$.

The binary motif feature provides a minimalist abstraction for evolutionary processes where $0, 1$ correspond to favorable and nonfavorable directions, respectively, for each motif. Theoretical analysis for evolutionary optimization algorithms made the same assumption and viewed binary sequence optimization as a fundamental problem [14, 24, 27, 28, 35, 55].

Since a protein function is largely determined by its motif, it is common to model the protein utility $f : \mathcal{X} \to \mathbb{R}$ as a function of motif features, i.e., $f(seq) := f_{\theta^\star}(x), x = \phi(seq), \forall seq \in \mathcal{X}$, under a parameterization by $\theta^\star$ [16, 57, 45, 18].

In this work, we study the most elementary Bayesian linear model, where f is a linear model parameterized by θ^* with a Gaussian prior, given as follows.

Assumption 3.2. (Linear Bayesian Utility Model) Assume the utility $f_{\theta^\star}$ is a linear function parameterized by $\theta^\star \in \mathbb{R}^d$, which is sampled from a Gaussian prior, i.e.

$$f_{\theta^\star}(x) = \langle \theta^\star, x \rangle, \qquad \theta^\star \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}), \quad \lambda > 0. \tag{3.2}$$

Since motifs tend to mutate and recombine with one another in units, it is often sufficient to focus on recombination and mutation on the motif level, rather than on the entry level. Further, recombination that breaks a motif often result in insignificant low-fitness descendants. Therefore, it suffices to focus on motif-level directed evolution for simplicity of presentation and theory. For theoretical simplicity, we define recombination and mutation operators **on the motif level**:

Definition 3.3 (Directed Mutation Operator)**.** Let x be the motif feature sequence, $\mathcal{I} \subset [d]$ be a collection of targeted sites and $\mu \in (0, 1)$ be a mutation rate. The mutation operator $\texttt{Mut}(x, \mathcal{I}, \mu)$ generates a sequence x' such that while for $\forall j \notin \mathcal{I}, x'_j = x_j$, for $\forall i \in \mathcal{I}, x'_i$ is independently induced to be

$$\begin{cases} x'_i \sim \text{unif}(\{0, 1\}), & \text{w.p. } \mu, \\ x'_i = x_i, & \text{otherwise.} \end{cases} \tag{3.3}$$

Definition 3.4 (Recombination Operator)**.** Let x, y be the motif features associated with two parental genetic sequences. The recombination operator $\texttt{Rcb}(x, y)$ generates a child sequence z such that z_i's are independent and

$$z_i = \begin{cases} x_i & \text{w.p. } \frac{1}{2} \\ y_i & \text{w.p. } \frac{1}{2} \end{cases}, \ \forall i \in [d]. \tag{3.4}$$

We remark that Definitions 3.3, 3.4 are *mathematical simplifications* of their real-world counterparts. In real world, mutation and recombination can take various forms depending on the context. In our analysis, we define them in a minimalist-style to keep theory generalizable and interpretable.

3.3 Regret minimization problem formulation

Evaluating the protein function for a design sequence x is a most costly and time-consuming step in protein engineering. In the DE process, we consider that regret is incurred only when sequences are evaluated. We also assume that each evaluation is subject to a Gaussian noise with known variance.

Assumption 3.5. (Noisy Feedback) Upon querying the utility of x, we get an independent noisy evaluation given by

$$u(x) \sim \mathcal{N}(f_{\theta^\star}(x), \sigma^2). \tag{3.5}$$

Our goal is to minimize the Bayesian regret, i.e., the cumulative sum of optimality gaps between evaluated sequences and the optimal.

Definition 3.6 (Bayesian Regret)**.** Denote by $f_{\theta^\star}(x^\star)$ the optimal utility value over $\mathcal{X}$, $\{x_{t,i}\}_{i=1}^M$ are the evaluated individuals in each iteration. Throughout T iteration, the accumulated regret is defined as

$$\text{BayesRGT}(T, M) = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{M}(f_{\theta^\star}(x^\star) - f_{\theta^\star}(x_{t,i}))\right],$$

where M is number of sequences selected for evaluation per timestep, and $\mathbb{E}$ is taken over the prior of $\theta^\star$ and all randomness in the DE process.

4 Thompson Sampling-guided directed evolution (TS-DE)

We restate our goal as to direct a population of genetic sequence to evolve towards higher utility value, until its population-average converges to the optimum $f_{\theta^\star}(x^\star)$. Our knowledge of f is to be learned from noisy evaluations of selected sequences along the way. In this section, by integrating the biological technique - directed evolution - with Thompson Sampling, a Bayesian bandit method, we propose the Thompson Sampling-guided Directed Evolution algorithm (TS-DE) as shown in Alg.1, where in each round Thompson sampling gives an estimate of $\theta^\star$, based on which key operators of DE: mutation, recombination and selection are implemented.

4.1 Crossover-then-selection and directed mutation

Pairwise crossover is a most common type of recombination in natural evolution. Let x, y be a random pair of parents, and let $z = \texttt{Rcb}(x, y)$ be a child. If given a utility function f, we select z only if the child performs better than the parents' average. Module 1 formulates this procedure.

Module 1 `Crossover_Selection`(f, S)

Inputs: utility function $f(x) = \langle \theta, x \rangle$, a population of sequences S
Initialization: $S' \leftarrow \emptyset$
while $|S'| < |S|$ **do**
  Sample x and y from S uniformly with replacement.
  Recombination: $z \leftarrow \texttt{Rcb}(x, y)$ (Definition 3.4).
  Selection: $S' \leftarrow S' \cup \{z\}$ if $f(z) \geq \frac{f(x)+f(y)}{2}$.
end while
Output: S'

Next we turn to designing the strategy for adding directed mutation under a given f as guidance and propose Module 2. An ideal mutation will diversify the population while preserving its fitness level as much as possible. So we add directed mutation to sites where the single site fitness over the population is less than of a uniformly distributed sequence. Formally, we only add mutation to site i if $\frac{1}{M}\sum_{x \in S} \theta_i \cdot x_i \leq \theta_i \cdot \bar{x}_i$, where $\bar{x}_i$ is the mean of uniformly random x_i.

Module 2 `Directed_Mutation`(f, S, μ)

Inputs: utility function $f(x) = \langle \theta, x \rangle$, a population of sequences S, mutation rate μ
Initialization: $\mathcal{I} \leftarrow \emptyset, \mathcal{S}' \leftarrow \emptyset$
for $i \in [d]$ **do**
  if $\frac{1}{M}\sum_{x \in S} \theta_i \cdot x_i \leq \theta_i \cdot \bar{x}_i$ **then**
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$.
  end if
end for
Directed Mutation: $x' = \texttt{Mut}(x, \mathcal{I}, \mu)$ (Definition 3.3) and $S' \leftarrow S' \cup \{x'\}$ for all $x \in S$.
Output: S'

4.2 Full algorithm

Finally, we are ready to combine all modules and state the full algorithm in Algorithm 1. At each time step t, a posterior distribution is first computed using the data collected in history. Then we sample a $\widetilde{\theta}_t$ from the posterior and do the corresponding directed mutation and crossover selection using this sampled weight, and augment the dataset for the next iteration with the measurements of resulting new population. The procedure is repeated until the time limit T is reached.

5 Main results

In this section, we analyze the performance of TS-DE (Algorithm 1). We will show that the crossover selection module essentially mimics an optimization iteration that strictly improves the population's

Algorithm 1 Thompson Sampling-Guided Directed Evolution (TS-DE)

Inputs: number of rounds T, initial population $S_0 = \{x_{0,i}\}_{i=1}^M$ of size M, mutation rate μ, σ
Initialization: dataset $D_0 \leftarrow \emptyset$, $\Phi_{t-1} = 0$, $U_0 = 0$
for $t = 1$ to T **do**
  Posterior update

$$V_t = \frac{1}{\sigma^2}\Phi_{t-1}^\top \Phi_{t-1} + \lambda I, \qquad \widehat{\theta}_t = \frac{1}{\sigma^2} V_t^{-1} \Phi_{t-1}^\top U_{t-1}. \tag{4.1}$$

  Thompson Sampling $\widetilde{\theta}_t \sim \mathcal{N}(\widehat{\theta}_t, V_t^{-1})$.
  $S'_{t-1} = \texttt{Directed_Mutation}(f_{\widetilde{\theta}_t}, S_{t-1}, \mu)$ (Module 2).
  $S_t = \texttt{Crossover_Selection}(f_{\widetilde{\theta}_t}, S'_{t-1})$ (Module 1).
  Evaluation and data collection Evaluate the utilities of all individuals in S_t and $D_t \leftarrow D_{t-1} \cup \{x_{t,i}, u(x_{t,i})\}_{i=1}^M$. Update $\Phi_t^\top \leftarrow \left(\Phi_{t-1}^\top, x_{t,1}, \cdots, x_{t,M}\right)$, $U_t \leftarrow \left(U_{t-1}^\top, u(x_{t,1}), \cdots, u(x_{t,M})\right)^\top$.
  $t \leftarrow t + 1$.
end for

fitness along the designated direction. By using a Bayesian regret analysis, we show the DE modules, when combined with posterior sampling, can effectively optimize towards the best protein design while learning $\theta^\star$.

5.1 Crossover selection as an optimization iteration

Let f by any utility function, and let $F(S) := \text{avg}_{x \in S} f(x)$ denote the population average utility. Our first result states an ascent property showing that `Crossover_Selection` strictly improves the population average.

Theorem 5.1 (Ascent Property of Recombination-then-Selection)**.** Let $f(x) = \langle \theta, x \rangle$ and let S be a set of sequences. Let $S' = \texttt{Crossover_Selection}(f, S)$, then it satisfies

$$\mathbb{E}\left[F(S')\right] \geq F(S) + \frac{\mathbb{E}_{x,y}\left[\|\theta \cdot (x - y)\|\right]}{2\sqrt{2}} \geq F(S) + \frac{1}{\sqrt{2d}} \sum_i |\theta_i| \, \text{Var}_i(S), \tag{5.1}$$

where $\text{Var}_i(S)$ denotes the variance of x_i when x is uniformly sampled from S.

Proof sketch. See Figure 5.1 for illustration. Given x and y, $z = \texttt{Rcb}(x, y)$ can be represented by $z = \frac{x+y}{2} + \frac{x-y}{2} \cdot e$, where the $\cdot$ denotes the entrywise multiplication between two vectors and $e = (e_i, \cdots, e_d)$ with e_i's being independent Rademacher variables. Then $f(z)$ equals $\frac{f(x)+f(y)}{2} + \frac{1}{2}\sum_{i=1}^d \theta_i (x_i - y_i) e_i$. After the selection step, the expected amount by which $f(z)$ exceeds its parents' average is at least $\frac{1}{2}\mathbb{E}\left[\left|\sum_{i=1}^d \theta_i (x_i - y_i) e_i\right|\right]$, which has a tight lower bound of $\frac{1}{2\sqrt{2}}\|\theta \cdot (x - y)\|$ according to Haagerup [20]. The full proof is given in Appendix C.1. ■

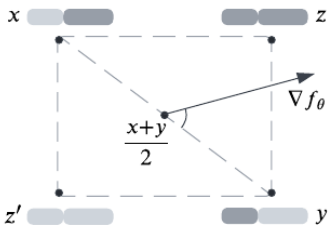

Figure 5.1: Ascent property of crossover recombination

Remark on diversity. Analysis above reveals an intriguing observation: the optimization progress of `Crossover_Selection` scales linearly with $\sum_i \theta_i \text{Var}_i(S)$, i.e., sum of per-motif variances across population S. It measures a level of "diversity" of S with respect to direction θ. More diverse population would enjoy larger progress from crossover selection. This observation is consistent with the natural evolution theory that diversity is key to the adaptability of a population to cope with evolving environment where fitness traits are essential [52].

5.2 Regret bound of TS-DE

Our main result is a Bayesian regret bound for TS-DE. Recall from Definition 3.6 that $\text{BayesRGT}(T, M) = \mathbb{E}[\sum_{t=1}^{T}\sum_{i=1}^{M}(f_{\theta^\star}(x^\star) - f_{\theta^\star}(x_{t,i}))]$.

Theorem 5.2. Under Assumption 3.2 and 3.5, when the population size is sufficient s.t. $M = \Omega\left(\frac{\log(dT)}{\mu^2}\right)$, Alg.1 admits its Bayesian regret s.t.

$$\text{BayesRGT}(T, M) = \widetilde{O}\left(\frac{d}{\mu\sqrt{\lambda}} \cdot d\sqrt{MT}\right). \tag{5.2}$$

If we let $\lambda = 1, \mu = 1/2, \sigma^2 = 1$, the Bayesian regret simplifies to $\widetilde{O}(d^2\sqrt{MT})$.

Remark on regret bound. Regret bound of Theorem 5.2 is optimal in M, T. For comparison, the Bayesian regret of Gaussian linear model is $\widetilde{O}(d\sqrt{T})$ [31], also in contextual linear bandit with batch update, the optimal regret is $\widetilde{O}(d\sqrt{MT})$ [22]. Our TS-DS regret has two extra factors of $\sqrt{d}$. One $\sqrt{d}$ is due to that the l_2 norm of our feature vectors are $\sqrt{d}$, while linear bandit theory often assumes feature to have norm 1. Another factor of $\sqrt{d}$ is due to the evolutionary nature of DE, i.e., TS-DE is not allowed to any possible action but have to select those from the evolving population.

5.3 Proof sketch

Main challenge. Classic bandit method/analysis does not apply to our setting, because each round of DE is limited to actions that are reachable by mutation and recombination based on the current population. It means that we cannot simply explore the optimistic actions that maximize each function estimate $f_{\widetilde{\theta}_t}$. This leads to an optimization gap that complicates the regret proof.

Denote by $x^\star$ and $x_t^\star$ the maximums of $f_{\theta^\star}$ and $f_{\widetilde{\theta}_t}$. Denote by $F_t^\star := f_{\widetilde{\theta}_t}(x_t^\star)$ the maximum value of $f_{\widetilde{\theta}_t}$ and denote by $F_t(S)$ the average value of $f_{\widetilde{\theta}_t}$ over set S.

Step 1: Regret decomposition. With expectation taken over all stochasticity, posterior sampling guarantees $\text{BayesRGT}(T, M) = \sum_{t=1}^{T}\sum_{i=1}^{M}\mathbb{E}\left[f_{\widetilde{\theta}_t}(x_t^\star) - f_{\theta^\star}(x_{t,i})\right]$ since conditioned on data D_{t-1}, $f_{\theta^\star}(x^\star)$ and $f_{\widetilde{\theta}_t}(x_t^\star)$ are identically distributed. Then by breaking $f_{\widetilde{\theta}_t}(x_t^\star) - f_{\theta^\star}(x_{t,i})$ down to the sum of $f_{\widetilde{\theta}_t}(x_t^\star) - f_{\widetilde{\theta}_t}(x_{t,i})$ and $f_{\widetilde{\theta}_t}(x_{t,i}) - f_{\theta^\star}(x_{t,i})$, we decompose the total regret into

$$\text{BayesRGT}(T, M) = M \cdot \underbrace{\mathbb{E}\left[\sum_{t=1}^{T}(F_t^\star - F_t(S_t))\right]}_{H_1} + \underbrace{\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{M}\langle\widetilde{\theta}_t - \theta^\star, x_{t,i}\rangle\right]}_{H_2}. \tag{5.3}$$

Step 2: Bounding H_1 using linear convergence. H_1 is the accumulated optimization error under a time-varying objective $f_{\widetilde{\theta}_t}$. After calling $S'_{t-1} = \texttt{Directed_Mutation}(f_{\widetilde{\theta}_t}, S_{t-1}, \mu)$ and $S_t = \texttt{Crossover_Selection}(f_{\widetilde{\theta}_t}, S'_{t-1})$ at step t, the ascent property (5.1) together with property of the mutation module yields a linear convergence towards $F_t^\star$, i.e., $\mathbb{E}\left[F_t^\star - F_t(S_t) \mid S_{t-1}, \widetilde{\theta}_t\right] \leq \gamma(F_t^\star - F_t(S_{t-1}))$ with a modulus of contraction $\gamma \in (0, 1)$ s.t. $\frac{1}{1-\gamma} = O\left(\frac{\sqrt{d}}{\mu}\right)$. It follows that

$$F_t^\star - F_t(S_t) \leq \gamma\left[F_{t-1}^\star - F_{t-1}(S_{t-1})\right] + \text{error terms} + e_t,$$

where e_t is a martingale difference. Applying the above recursively to H_1, we get $H_1 \leq$

$$\underbrace{\frac{1}{1-\gamma} \cdot \mathbb{E}\left[F_1^\star - F_1(S_0)\right]}_{O(\frac{1}{1-\gamma})} + \underbrace{\mathbb{E}\left[\sum_{k=2}^{T}\gamma^{T-k+1}F_k^\star - \gamma^{T-1}F_1^\star\right]}_{O(\frac{1}{1-\gamma})} + \underbrace{\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{t-1}\gamma^{t-k}\left(F_k(S_k) - F_{k+1}(S_k)\right)\right]}_{\kappa},$$

which is dominated by term κ and $M \cdot \kappa \leq \frac{1}{1-\gamma} \cdot \sum_{t=1}^{T-1}\sum_{i=1}^{M}\left|\langle\widetilde{\theta}_t - \widetilde{\theta}_{t+1}, x_{t,i}\rangle\right| = O\left(\frac{1}{1-\gamma}H_2\right)$.

Step 3: Bounding H_2. H_2 is the accumulated prediction error of $\widetilde{\theta}_t$, which is a classic term to bound in bandit literature and is of $\widetilde{O}\left(d^{1.5}\sqrt{MT}\right)$ by using a batched self-normalization bound. ■

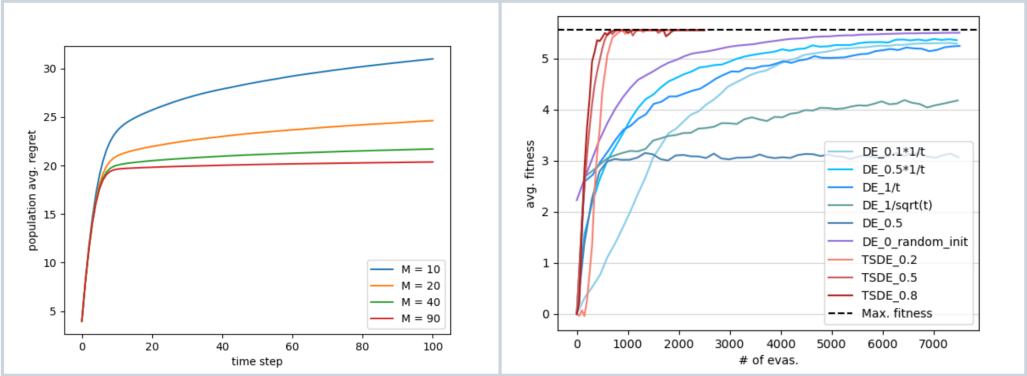

Figure 6.1: **Regret and fitness curves of TS-DE during evolution.** Left: Population-averaged regret with varying population sizes M. Each curve is averaged over 100 trials. Right: Fitness curves of TS-DE with varying values of μ, compared with basic DE with varying mutation rates. (The purple curve plots basic DE without mutation, we modified the initial population to be uniformly distributed in this case to make it non-trivial.)

6 Experiments

6.1 Simulation

We test the TS-DE by simulating the evolution of a population of sequences in $\{0,1\}^d$. We set the initial population to be all zeros, and set $\lambda = 1$, $\sigma = 1$.

Regret and convergence results. Figure 6.1 shows the regret curves and learning curves of TS-DE, with comparison to basic DE. In the left panel of Figure 6.1, we plot the population-averaged Bayesian regret of TS-DE with various values of M, where $d = 10$, $T = 100$ and $\mu = 0.8$. These results confirm our sublinear regret bounds. In the right panel of Figure 6.1, we tested TS-DE using various mutation rates, and compared them with a basic DE approach [5]. The comparison shows that TS-DE converges significantly faster, while the convergence of DE is much slower and very sensitive to mutation scheduling.

Visualizing the evolution of a population. We visualize the evolution trajectory of population S_t in one run of TS-DE, with $d = 40$, $M = 20$ and $\mu = 0.1$. In the left panel of Fig.6.2, we visualize the evolving high-dimensional population S_t by mapping them to 2D (via PCA and KDE density contour plot). In the right panel of Fig.6.2, we plot the fitness distribution of each S_t. These plots illustrate how TS-DE balances the exploration-exploitation trade-off: It guides S_t to "diversify" initially and then quickly approach and concentrate around a maximal solution.

6.2 Real-world experiment validation

Having demonstrated our approach with simulations, we use real-world experiments to showcase the validity and generalizability of our method. The TS-DE method is adapted to work with real-world motif features (continuous-valued instead of binary), linear model and multiple rounds of wet-lab experiments for optimizing a CRISPR design sequence. Our approach together with high-throughput experiment identified a high-performing sequence with 30+ fold improvement in efficiency. Notably, the optimized CRISPR designs generated by our DE approach was experimentally validated in [26] and demonstrated the real-world utility of our method. This technology is used for ex-vivo high-throughput single-cell barcoding with applications in genomics and drug discovery.

We postpone more details about this real-world validation to Appendix B.1 and Figure B.1.

[5] The basic DE approach does not employ any function estimate. It does random mutation with a predefined mutation rate and random crossover recombination. It evaluates every candidate sequence and uses the noisy feedback in replace of $f_{\tilde{\theta}}$ for selection.

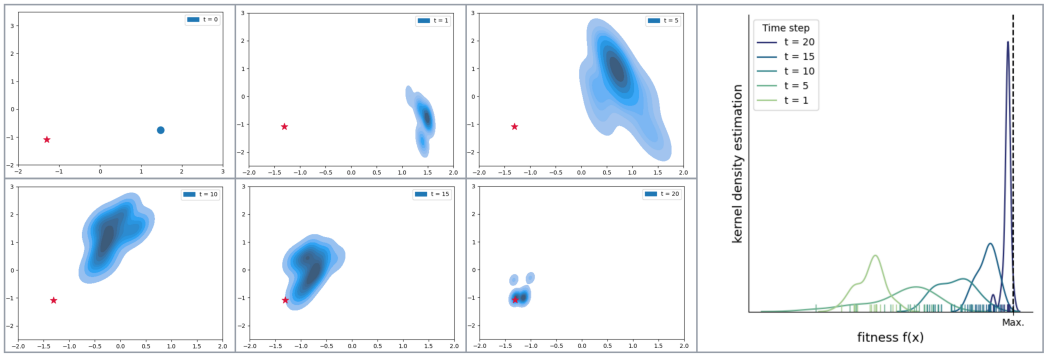

Figure 6.2: **Evolving population of TS-DE and fitness levels.** Left panels: Visualization of population evolution projected in 2D shown, taken at 6 snapshots. Right panel: The population's fitness distribution shifts towards optimal during evolution. ⋆ denotes the optimal solution.

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] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

[3] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

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

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

[6] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[7] 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.

[8] 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.

[9] 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.

[10] Joseph M Cunningham, Grigoriy Koytiger, Peter K Sorger, and Mohammed AlQuraishi. Biophysical prediction of protein–peptide interactions and signaling networks using machine learning. *Nature methods*, 17(2):175–183, 2020.

[11] Benjamin Doerr and Marvin Künnemann. Optimizing linear functions with the (1+ λ) evolutionary algorithm—different asymptotic runtimes for different instances. *Theoretical Computer Science*, 561:3–23, 2015.

[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] Jennifer A Doudna and Emmanuelle Charpentier. The new frontier of genome engineering with crispr-cas9. *Science*, 346(6213):1258096, 2014.

[14] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+ 1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[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] Christian Gießen and Carsten Witt. Optimal mutation rates for the (1+ λ) ea on onemax. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 1147–1154, 2016.

[20] Uffe Haagerup. The best constants in the khintchine inequality. *Studia Mathematica*, 70: 231–283, 1981.

[21] Nima Hamidi and Mohsen Bayati. On worst-case regret of linear thompson sampling. *arXiv preprint arXiv:2006.06790*, 2020.

[22] Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose Blanchet, Peter W Glynn, and Yinyu Ye. Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321*, 2020.

[23] Botao Hao, Tor Lattimore, and Mengdi Wang. High-dimensional sparse linear bandits. *Advances in Neural Information Processing Systems*, 33:10753–10763, 2020.

[24] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

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

[26] Nicholas W Hughes, Yuanhao Qu, Jiaqi Zhang, Weijing Tang, Justin Pierce, Chengkun Wang, Aditi Agrawal, Maurizio Morri, Norma Neff, Monte M Winslow, et al. Machine-learning-optimized cas12a barcoding enables the recovery of single-cell lineages and transcriptional profiles. *Molecular Cell*, 82(16):3103–3118, 2022.

[27] Jens Jägersküpper. A blend of markov-chain and drift analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 41–51. Springer, 2008.

[28] Jens Jägersküpper. Combining markov-chain analysis and drift analysis. *Algorithmica*, 59(3): 409–424, 2011.

[29] Thomas Jansen and Ingo Wegener. Real royal road functions—where crossover provably is essential. *Discrete applied mathematics*, 149(1-3):111–125, 2005.

[30] Thomas Jansen, Ingo Wegener, et al. The analysis of evolutionary algorithms–a proof that crossover really can help. *Algorithmica*, 34(1):47–66, 2002.

[31] Cem Kalkanlı and Ayfer Özgür. An improved regret bound for thompson sampling in the gaussian linear bandit setting. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2783–2788. IEEE, 2020.

[32] Timo Kötzing, Dirk Sudholt, and Madeleine Theile. How crossover helps in pseudo-boolean optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 989–996, 2011.

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

[34] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[35] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64 (4):623–642, 2012.

[36] 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.

[37] Dan Ofer and Michal Linial. Profet: Feature engineering captures high-level protein functions. *Bioinformatics*, 31(21):3429–3436, 2015.

[38] Dan Ofer, Nadav Brandes, and Michal Linial. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 19:1750–1758, 2021.

[39] Pietro S Oliveto and Carsten Witt. Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science*, 605:21–41, 2015.

[40] Pietro S Oliveto, Dirk Sudholt, and Carsten Witt. A tight lower bound on the expected runtime of standard steady state genetic algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 1323–1331, 2020.

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

[42] 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.

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

[44] Ophir Shalem, Neville E Sanjana, and Feng Zhang. High-throughput functional genomics using crispr–cas9. *Nature Reviews Genetics*, 16(5):299–311, 2015.

[45] 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.

[46] Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric D Kelsic. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

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

[48] Jérôme Tubiana, Simona Cocco, and Rémi Monasson. Learning protein constitutive motifs from sequence data. *Elife*, 8:e39397, 2019.

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

[50] 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.

[51] Richard A Watson and Thomas Jansen. A building-block royal road where crossover is provably essential. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1452–1459, 2007.

[52] Robert H Whittaker. Evolution and measurement of species diversity. *Taxon*, 21(2-3):213–251, 1972.

[53] 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.

[54] Carsten Witt. Runtime analysis of the (μ+ 1) ea on simple pseudo-boolean functions. *Evolutionary Computation*, 14(1):65–86, 2006.

[55] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing*, 22(2):294–318, 2013.

[56] Bruce J Wittmann, Kadina E Johnston, Zachary Wu, and Frances H Arnold. Advances in machine learning for directed evolution. *Current opinion in structural biology*, 69:11–18, 2021.

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

[58] Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pages 10746–10756. PMLR, 2020.

[59] Zhi-Hua Zhou, Yang Yu, and Chao Qian. *Evolutionary learning: Advances in theories and algorithms*. Springer, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]