Improved Off-policy Reinforcement Learning in Biological Sequence Design

Hyeonah Kim, Minsu Kim, Taeyoung Yun, Sanghyeok Choi
Korea Advanced Institute of Science and Technology (KAIST)
{hyeonah_kim, min-su, 99yty, sanghyeok.choi}@kaist.ac.kr

Emmanuel Bengio
Recursion
{emmanuel.bengio}@recursionpharma.com

Alex Hernández-García
Université de Montréal, Mila - Quebec AI Institute
{hernanga}@mila.quebec

Jinkyoo Park
KAIST
{jinkyoo.park}@kaist.ac.kr

Abstract

Designing biological sequences with desired properties is a significant challenge due to the combinatorially vast search space and the high cost of evaluating each candidate sequence. To address these challenges, reinforcement learning (RL) methods, such as GFlowNets, utilize proxy models for rapid reward evaluation and annotated data for policy training. Although these approaches have shown promise in generating diverse and novel sequences, the limited training data relative to the vast search space often leads to the misspecification of proxy for out-of-distribution inputs. We introduce δ-Conservative Search, a novel off-policy search method for training GFlowNets designed to improve robustness against proxy misspecification. The key idea is to incorporate conservativeness, controlled by parameter δ, to constrain the search to reliable regions. Specifically, we inject noise into high-score offline sequences by randomly masking tokens with a Bernoulli distribution of parameter δ and then denoise masked tokens using the GFlowNet policy. Additionally, δ is adaptively adjusted based on the uncertainty of the proxy model for each data point. This enables the reflection of proxy uncertainty to determine the level of conservativeness. Experimental results demonstrate that our method consistently outperforms existing machine learning methods in discovering high-score sequences across diverse tasks—including DNA, RNA, protein, and peptide design—especially in large-scale scenarios. The code is available at https://github.com/hyeonahkimm/delta_cs.

1 Introduction

Designing biological sequences with desired properties is crucial in therapeutics and biotechnology (Zimmer, 2002; Lorenz et al., 2011; Barrera et al., 2016; Sample et al., 2019; Ogden et al., 2019). However, this task is challenging due to the combinatorially large search space and the expensive and black-box nature of objective functions. Recent advances in deep learning methods for biological sequence design have shown significant promise at overcoming these challenges (Brookes & Listgarten, 2018; Brookes et al., 2019; Angermueller et al., 2020; Jain et al., 2022).

AI for New Drug Modalities at NeurIPS 2024.

Among various approaches, reinforcement learning (RL), which leverages a proxy model as a reward function, has emerged as one of the successful paradigms for automatic biological sequence design (Angermueller et al., 2020). RL methods have the benefit of exploring diverse sequence spaces by generating sequences token-by-token from scratch, enabling the discovery of novel sequences. They employ deep neural networks as inexpensive proxy models to approximate costly oracle objective functions. The proxy model serves as a reward function for training deep RL algorithms, enabling the policy network to generate high-reward biological sequences. The model can be trained in an active learning manner (Gal et al., 2017), iteratively annotating new data by querying the oracle with points generated using the policy; these iterations are called query rounds. There are two approaches for training the policy in this context: on-policy and off-policy.

DyNA PPO (Angermueller et al., 2020), a representative on-policy RL method for biological sequence design, employs Proximal Policy Optimization (PPO; Schulman et al., 2017) within a proxy-based active training loop. While DyNA PPO has demonstrated effectiveness in various biological sequence design tasks, its major limitation is the limited search flexibility inherent to on-policy methods. It cannot effectively leverage off-policy data points, such as data collected from previous rounds.

Conversely, Generative Flow Networks (GFlowNets; Bengio et al., 2021), off-policy RL methods akin to maximum entropy policies (Tiapkin et al., 2024; Deleu et al., 2024), offer diversity-seeking capabilities and flexible exploration strategies. Jain et al. (2022) applied GFlowNets to biological sequence design with additional Bayesian active learning schemes. They leveraged the off-policy nature of GFlowNets by mixing offline datasets with on-policy data during training. This approach provided more stable training compared to DyNA PPO and resulted in better performance.

However, recent studies have consistently reported that GFlowNets perform poorly on long-sequence tasks such as green fluorescent protein (GFP) design (Kim et al., 2023; Surana et al., 2024). We hypothesize that this poor performance stems from the insufficient quality of the proxy model in the early rounds. For example, in the typical benchmark, the proxy is trained with 5,000 sequences, while GFPs have a combinatorial search space of $20^{238} > 10^{309}$. GFlowNets are capable of generating sequences from scratch and producing novel sequences beyond the data points. However, when the quality of the proxy model is unreliable, novel sequences that are out-of-distribution to the proxy model yield unreliable results (Trabucco et al., 2021; Yu et al., 2021). This motivates us to introduce a conservative search strategy to restrict the trajectories to the neighborhood of the existing data point.

Contribution. In this paper, we propose a novel off-policy search method called δ-Conservative Search (δ-CS), which enables a trade-off between sequence novelty and robustness to proxy misspecification by using a conservativeness parameter δ. Specifically, we iteratively train a GFlowNet using δ-CS as follows: (1) we inject noise by independently masking tokens in high-score offline sequences with Bernoulli distribution with parameter δ; (2) the GFlowNet policy sequentially denoises the masked tokens; (3) we use these denoised sequences to train the policy. When $\delta = 1$ (full on-policy), this becomes equivalent to sampling full trajectories, and when $\delta = 0$ (fully conservative), this reduces to only showing offline sequences in off-policy training. We adaptively adjust $\delta(x; \sigma)$ using the proxy model's uncertainty estimates $\sigma(x)$ for each data point x. This approach allows the level of conservativeness to be adaptively adjusted based on the prediction uncertainty of each data point. Figure 1 illustrates the overall procedure of the δ-CS algorithm.

Our extensive experiments demonstrate that δ-CS significantly improves GFlowNets, successfully discovering higher-score sequences compared to existing model-based optimization methods on diverse tasks, including DNA, RNA, protein, and peptide design. This result offers a robust and scalable framework for advancing research and applications in biotechnology and synthetic biology.

2 Problem Formulation

We aim to discover sequences $x \in \mathcal{V}^L$ that exhibit desired properties, where $\mathcal{V}$ denotes the vocabulary, such as amino acids or nucleotides, and L represents the sequence length, which is usually fixed. The desired properties are evaluated by a black-box oracle function $f : \mathcal{V}^L \to \mathbb{R}$, which evaluates the desired property of a given sequence, such as binding affinity or enzymatic activity. Evaluating f is often both expensive and time-consuming since it typically involves wet-lab experiments or high-fidelity simulations.

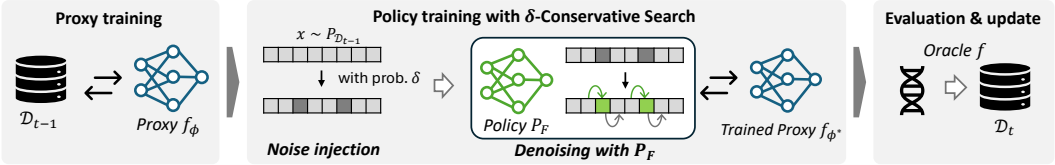


Figure 1: The active learning process for biological sequence design with δ-Conservative Search (δ-CS). Starting with high reward sequences from the offline dataset, we inject token-level noise with probability δ, which determines the conservativeness of the search. Then, the GFlowNet policy denoises the masked sequences. Lastly, the GFlowNet policy is trained with new sequences. After policy training, we query a new batch of sequences and update the dataset for the next round.

Advancements in experimental techniques have enabled the parallel synthesis and evaluation of sequences in batches. Therefore, lab-in-the-loop processes are emerging as practical settings that enable active learning. Following this paradigm, we perform T rounds of batch optimization, where in each round, we have the opportunity to query B batched sequences to the (*assumed*) oracle objective function f. Due to the labor-intensive nature of these experiments, T is typically very small. Following Angermueller et al. (2020) and Jain et al. (2022), we assume the availability of an initial offline dataset $\mathcal{D}_0 = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N_0}$, where $y = f(x)$. The initial number of data points N_0 is typically many orders of magnitude smaller than the size of the search space, as mentioned in the introduction. The goal is to discover, after T rounds, a set of sequences that are novel, diverse, and have high oracle function values.

3 Active Learning for Biological Sequence Design

Following Jain et al. (2022), we formulate an active learning process constrained by a budget of T rounds with query size B, is executed through an iterative procedure consisting of three stages with a novel component of δ-Conservative Search (δ-CS) which will be detailed described in Section 4:

Step A (Proxy Training): We train a proxy model $f_\phi(x)$ using the offline dataset $\mathcal{D}_{t-1}$ at round t.

Step B (Policy Training with δ-CS): We train a generative policy $p(x;\theta)$ using the proxy model $f_\phi(x)$ and the dataset $\mathcal{D}_{t-1}$ with δ-CS.

Step C (Offline Dataset Augmentation with δ-CS): We apply δ-CS to query batched data $\{x_i\}_{i=1}^B$ to the oracle $y_i = f(x_i)$. Then the offline dataset is augmented as: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(x_i, y_i)\}_{i=1}^B$.

The overall algorithm is described in Algorithm 1. In the following subsections, we describe the details of **Step A** and **Step B**.

3.1 Step A: Proxy Training

Following Jain et al. (2022), we train the proxy model f_ϕ using the dataset $\mathcal{D}_{t-1}$ by minimizing the mean squared error loss:

$$\mathcal{L}(\phi) = \mathbb{E}_{x \sim P_{\mathcal{D}_{t-1}}(x)} \left[\left(f(x) - f_\phi(x) \right)^2 \right], \tag{1}$$

where $\mathcal{D}_t$ is the dataset at active round t, augmented with oracle queries. In the initial round ($t = 1$), we use the given initial dataset $\mathcal{D}_0$. See Appendix A.1 for detailed implementation.

3.2 Step B: Policy training with δ-CS

For policy training, we employ GFlowNets, which aim to produce samples from a generative policy where the probability of generating a sequence x is proportional to its reward, i.e.,

$$p(x;\theta) \propto R(x;\phi) = f_\phi(x) + \kappa\sigma(x). \tag{2}$$

Following Jain et al. (2022), the reward $R(x;\phi)$ is defined as $f_\phi(x) + \kappa\sigma(x)$, which combines the proxy value $f_\phi(x)$ and the uncertainty $\sigma(x)$ in the form of the upper confidence bound (UCB;

Srinivas et al., 2010) acquisition function. This approach prioritizes regions with higher uncertainty, enabling us to query them in the next active round. Here, κ is a mixing hyperparameter.

Policy parameterization. The forward policy P_F generates state transitions sequentially through trajectories $\tau = (s_0 \to \ldots \to s_L = x)$, where $s_0 = ()$ represents the empty sequence, and each state transition involves adding a sequence token. The full sequence $s_L = x$ is obtained after L steps, where L is the length of the sequences. The forward policy $P_F(\tau;\theta)$ is a compositional policy defined as:

$$P_F(\tau;\theta) = \prod_{i=1}^{L} P_F(s_i|s_{i-1};\theta). \tag{3}$$

GFlowNets have a backward policy $P_B(\tau|x)$ that models the probability of backtracking from the terminal state x. The sequence $x = (e_1, \ldots, e_L)$ can be uniquely converted into a state transition trajectory τ, where each intermediate state represents a subsequence. In the case of sequences, there is only a single way to backtrack, so $P_B(\tau|x) = 1$. This makes these types of GFlowNets equivalent to soft off-policy RL algorithms. For example, the trajectory balance (TB) objective of GFlowNets (Malkin et al., 2022) becomes equivalent to path consistency learning (PCL) (Nachum et al., 2017), an entropy-maximizing value-based RL method according to Deleu et al. (2024).

Learning objective and training trajectories. The policy is trained to minimize TB loss: $\mathcal{L}_{\text{TB}}(\tau;\theta)$:

$$\mathcal{L}_{\text{TB}}(\tau;\theta) = \left(\log \frac{Z_\theta P_F(\tau;\theta)}{R(x;\phi)}\right)^2 \tag{4}$$

Usually, GFlowNets training is employed to minimize TB loss with training trajectories τ on full supports, asymptotically guaranteeing optimality for the distribution:

$$p(x;\theta) \propto R(x;\phi).$$

A key challenge in prior works Jain et al. (2022) is that the proxy model $f_\phi(x)$ often produces highly unreliable rewards $R(x;\phi)$ for out-of-distribution inputs. In our approach, we mitigate this by providing off-policy trajectories within more reliable regions by injecting conservativeness into off-policy search. Therefore, **we minimize TB loss with δ-CS**, which offers controllable conservativeness.

4 δ-CS: controllable conservativeness in off-policy search

This section introduces δ-Conservative Search (δ-CS), an off-policy search method that enables controllable exploration through a conservative parameter δ. The parameter δ defines the Bernoulli distribution governing the masking of tokens in a sequence. Our algorithm is iterations of the following:

- Sample high-score offline sequences $x \sim P_{\mathcal{D}_{t-1}}(x)$ from the **rank-based reweighted prior**.
- Inject noise by masking tokens into x using the **noise injection policy** $P_{\text{noise}}(\tilde{x} \mid x, \delta)$.
- Denoise the masked tokens using the **denoising policy** $P_{\text{denoise}}(\hat{x} \mid \tilde{x};\theta)$.

These trajectories are used to update the GFlowNet parameters θ by minimizing the loss function $\mathcal{L}_{\text{TB}}(\tau;\theta)$. For more details on the algorithmic components of δ-CS and its integration with active learning GFlowNets, see Algorithm 1.

Rank-based reweighted prior. First, we sample a reference sequence x from the prior distribution $P_{\mathcal{D}_{t-1}}$. To exploit high-scoring sequences, we employ rank-based prioritization (Tripp et al., 2020).

$$w(x;\mathcal{D}_{t-1},k) \propto \frac{1}{kN + \text{rank}_{f,\mathcal{D}_{t-1}}(x)}.$$

Here, $\text{rank}_{f,\mathcal{D}_{t-1}}(x)$ is a relative rank of the value of $f(x)$ in the dataset $\mathcal{D}_{t-1}$ with a weight-shifting factor k; we fix $k = 0.01$. This assigns greater weight to sequences with higher ranks. Note that this reweighted prior can also be used during proxy training.

Noise injection policy. Let $x = (e_1, e_2, \ldots, e_L)$ denote the original sequence of length L. We define a noise injection policy where each position $i \in \{1, 2, \ldots, L\}$ is independently masked according to a Bernoulli distribution with parameter $\delta \in [0, 1]$, resulting in the masked sequence $\tilde{x} = (\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_L)$. The noise injection policy $P_{\text{noise}}(\tilde{x} \mid x, \delta)$ is defined as:

$$P_{\text{noise}}(\tilde{x} \mid x, \delta) = \prod_{i=1}^{L} \left[\delta \cdot \mathbb{I}\{\tilde{e}_i = \text{[MASK]}\} + (1 - \delta) \cdot \mathbb{I}\{\tilde{e}_i = e_i\}\right],$$

where $\mathbb{I}\{\cdot\}$ is the indicator function.

Denoising policy. We employ the GFlowNet forward policy P_F to sequentially reconstruct the masked sequence $\tilde{x} = (\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_L)$ by predicting tokens from left to right. The probability of denoising next token $\tilde{e}_t$ from previously denoised subsequence $\hat{s}_{t-1}$ is:

$$P_{\text{denoise}}(\hat{e}_t \mid \hat{s}_{t-1}, \tilde{x}; \theta) = \begin{cases} \mathbb{I}\{\hat{e}_t = \tilde{e}_t\}, & \text{if } \tilde{e}_t \neq \text{[MASK]}, \\ P_F(\hat{s}_t = (\hat{s}_{t-1}, \hat{e}_t) \mid \hat{s}_{t-1}; \theta), & \text{if } \tilde{e}_t = \text{[MASK]}. \end{cases}$$

The fully reconstructed sequence $\hat{x} = \hat{s}_L$ is obtained by sampling from:

$$P_{\text{denoise}}(\hat{x} \mid \tilde{x}; \theta) = \prod_{t=1}^{L} P_{\text{denoise}}(\hat{e}_t \mid \hat{s}_{t-1}, \tilde{x}; \theta).$$

By denoising the masked tokens with the GFlowNet policy, which infers each token sequentially from left to right, we generate new sequences $\hat{x}$ that balance novelty and conservativeness through the parameter δ.

4.1 Adjusting conservativeness parameter δ

Determining the conservative parameter δ is a crucial aspect of the algorithm. We propose and study two variants, constant and adaptive δ.

Constant. As a simple approach, we set δ as a constant, selecting it to have the noise policy mask 4–15 tokens per sequence. Despite its simplicity, this choice effectively enhances policy training and leads to the discovery of high-scoring sequences during active rounds; we provide further studies on δ-CS with a constant δ in Appendix B.

Adaptive. Another intuitive approach is to adjust δ based on the uncertainty of the proxy σ on each sequence x, that is $\delta(x; \sigma)$. Specifically, we define a function that assigns lower δ values for highly uncertain samples and vice versa: $\delta(x; \sigma) = \delta_{\text{const}} - \lambda\sigma(x)$. We estimate $\sigma(x)$, the standard deviation of the proxy model $f_\phi(x)$, via MC dropout (Gal & Ghahramani, 2016) or an ensemble method (Lakshminarayanan et al., 2017). λ is a scaling factor and related to the influence of the proxy uncertainty on δ; we set it to satisfy $\lambda\sigma \approx \frac{1}{L}$ based on the observations from the initial round.

In our main experiments, we use adaptive $\delta(x, \sigma)$ as the default setup.

5 Related work

Designing biological sequences using machine learning methods is widely studied. Bayesian optimization methods (Mockus, 2005; Belanger et al., 2019; Zhang et al., 2022) exploit posterior inference over newly acquired data points to update a Bayesian proxy model that can measure useful uncertainty. However, these methods usually suffer from scalability issues due to the complexity of the Gaussian process (GP) kernel (Belanger et al., 2019) or the difficulty of sampling from an intractable posterior (Zhang et al., 2022).

Offline model-based optimization (MBO) (Kumar & Levine, 2020; Trabucco et al., 2021; Yu et al., 2021; Chen et al., 2022; Kim et al., 2023; Chen et al., 2023a; Yun et al., 2024) also addresses the design of biological sequences using offline datasets only, which can be highly efficient because they do not require oracle queries. These approaches have reported meaningful findings, such as the conservative requirements on proxy models since proxy models tend to give high rewards on unseen samples (Trabucco et al., 2021; Yu et al., 2021; Yuan et al., 2023; Chen et al., 2023b). This supports

our approach of adaptive conservatism in the search process. Surana et al. (2024) recently noted that offline design and existing benchmarks are insufficient to reflect biological reliability, indicating that settings without additional Oracle queries might be too idealistic.

Reinforcement learning methods, such as DyNA PPO (Angermueller et al., 2020) and GFlowNets (Bengio et al., 2021; Jain et al., 2022, 2023; Hernández-García et al., 2024), and sampling with generative models (Brookes & Listgarten, 2018; Brookes et al., 2019; Das et al., 2021; Song & Li, 2023) aim to search the biological sequence space using a sequential decision-making process with a policy, starting from scratch. Similarly, sampling with generative models (Brookes & Listgarten, 2018; Brookes et al., 2019; Song & Li, 2023) searches the sequence space using generative models like VAE (Kingma & Welling, 2014). While these approaches allow for the creation of novel sequences, as sequences are generated from scratch, they are relatively prone to incomplete proxy models, particularly in regions where the proxy is misclassified due to being out-of-distribution.

An alternative line of research is evolutionary search (Arnold, 1998; Bloom & Arnold, 2009; Schreiber et al., 2020; Sinai et al., 2020; Ren et al., 2022; Ghari et al., 2023; Kirjner et al., 2024), a popular method in biological sequence design. Especially Ghari et al. (2023) proposed GFNSeqEditor, which utilizes GFlowNets as prior distribution to edit biological sequences as an evolutionary search. Evolutionary search methods iteratively edit given sequences and constrain the new sequences so as not to deviate too far from the seed sequence; they usually start from the *wild-type*, which occurs in nature. This can be viewed as constrained optimization, where out-of-distribution for the proxy model can lead to unrealistic and low-score biological sequences. Consequently, they do not aim to produce highly novel sequences. Our method can be seen as a hybrid of off-policy RL and evolutionary search, capitalizing on both the high novelty offered by GFlowNets and the high rewards with out-of-distribution robustness provided by constrained search where they are properly balanced by δ. Our experimental comparison with GFNSeqEditor (Ghari et al., 2023) demonstrates this hybridization balanced by δ enables us to discover sequences with greater novelty and higher rewards rather than merely using the GFlowNet policy as editing priors.

6 Experiments

Following the FLEXS benchmark (Sinai et al., 2020),[1] we evaluate our proposed method on various biological sequence design tasks. Furthermore, we analyze the effect of δ-CS by directly comparing with GFN-AL on TF-Bind-8 and an anti-microbial peptide design in Section 6.6. For each experiment, we conduct five independent runs.

Implementation details. For proxy models, we employ a convolutional neural network (CNN) with one-dimensional convolutions (Sinai et al., 2020) with a UCB acquisition function and an ensemble of three network instances to measure the uncertainty. Note that we use the same architecture to implement proxy models for all baselines. For the GFlowNet policy, we use a simple two-layer long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) and train the policy with 1,000 inner-loop updates using a learning rate of 5×10^{-4} with a batch size of 256. However, in TF-Bind-8 and AMP, where we analyze the effectiveness of δ-CS compared to GFN-AL, we directly implement δ-CS on top of the GFN-AL implementation. Lastly, we set δ and λ according to the description in Section 4; specifically, $\delta = 0.5$ for tasks with $L \leq 50$ and $\delta = 0.05$ for long-sequences. More details are provided in Appendix A.2.

Baselines. As our baseline methods, we employ representative exploration algorithms. Further details are provided in Appendix A.3.

- **AdaLead** (Sinai et al., 2020) is a well-implemented model-guided evaluation method with a hill-climbing algorithm.
- **Bayesian optimization** (BO; Snoek et al., 2012) is a classical algorithm for black-box optimization. We employ the BO algorithm with a sparse sampling of the mutation space implemented by Sinai et al. (2020).
- **CMA-ES** (Hansen, 2006) is another well-known evolutionary algorithm that optimizes a continuous relaxation of one-hot vectors encoding sequence with the covariance matrix.
- **CbAS** (Brookes et al., 2019) and **DbAS** (Brookes & Listgarten, 2018) are probabilistic frameworks that use model-based adaptive sampling with a variational autoencoder (VAE; Kingma &

[1] FLEXS (Fitness Landscape EXploration Sandbox) is a widely-used open-source simulation environment for biological sequence design, which is available at https://github.com/samsinai/FLEXS.

Table 1: Maximum rewards achieved by each baseline method across six tasks, with the sequence length (L) for each task specified. The mean and standard deviation from five runs are reported. The highest values for each task are highlighted in **bold**.

	RNA-A ($L = 14$)	RNA-B ($L = 14$)	RNA-C ($L = 14$)	TF-Bind-8 ($L = 8$)	GFP ($L = 238$)	AAV ($L = 90$)
AdaLead	0.968 ± 0.070	0.965 ± 0.033	0.723 ± 0.057	$\mathbf{0.995} \pm \mathbf{0.004}$	3.581 ± 0.004	0.565 ± 0.027
BO	0.722 ± 0.025	0.720 ± 0.032	0.506 ± 0.003	0.977 ± 0.008	3.572 ± 0.000	0.500 ± 0.000
CMA-ES	0.816 ± 0.030	0.850 ± 0.063	0.496 ± 0.041	0.986 ± 0.008	3.572 ± 0.000	0.500 ± 0.000
CbAS	0.678 ± 0.020	0.668 ± 0.021	0.492 ± 0.018	0.988 ± 0.004	3.572 ± 0.000	0.500 ± 0.000
DbAS	0.670 ± 0.041	0.652 ± 0.021	0.495 ± 0.010	0.987 ± 0.004	3.572 ± 0.000	0.500 ± 0.000
DyNA PPO	0.737 ± 0.022	0.730 ± 0.088	0.478 ± 0.015	0.977 ± 0.013	3.572 ± 0.000	0.500 ± 0.000
GFN-AL	1.030 ± 0.024	1.001 ± 0.016	0.774 ± 0.004	0.976 ± 0.002	3.578 ± 0.003	0.560 ± 0.008
GFN-AL + δ-CS	$\mathbf{1.055} \pm \mathbf{0.000}$	$\mathbf{1.014} \pm \mathbf{0.001}$	$\mathbf{0.972} \pm \mathbf{0.043}$	0.981 ± 0.002	$\mathbf{3.592} \pm \mathbf{0.003}$	$\mathbf{0.708} \pm \mathbf{0.010}$

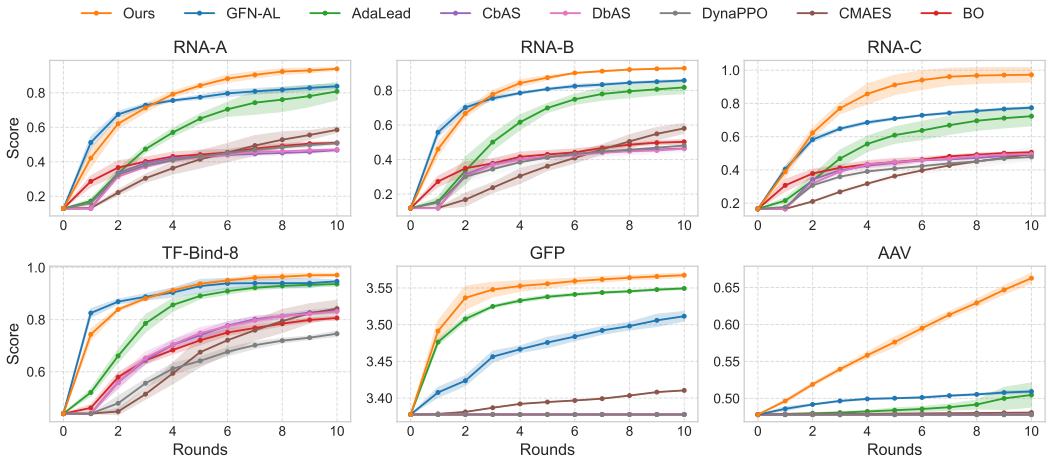


Figure 2: Median scores of Top-128 over active rounds. Ours (GFN-AL + δ-CS) consistently outperforms baseline in RNA, DNA (TF-Bind-8), and protein (GFP and AAV) design tasks.

Welling, 2014). Notably, CbAS restricts the search space with a trust-region search similar to the proposed method.

- **DyNA PPO** (Angermueller et al., 2020) uses proximal policy optimization (PPO; Schulman et al., 2017), an on-policy training method.
- **GFN-AL** (Jain et al., 2022) is our main baseline that uses GFN with Bayesian active learning.

For each task, we conduct 10 active learning rounds starting from the initial dataset $\mathcal{D}_0$. The query batch size is all set as 128 except for the AMP design, whose query size is 1,000. Further details of each task are provided in the following subsections. To evaluate the performance, we measure the maximum, median, and mean scores of Top-K sequences.

6.1 RNA sequence design

Task setup. The goal is to design an RNA sequence that binds to the target with the lowest binding energy, which is measured by ViennaRNA (Lorenz et al., 2011). The length (L) of RNA is set to 14, with 4 tokens. In this paper, we have three RNA binding tasks, RNA-A, RNA-B, and RNA-C, whose initial datasets consist of 5,000 randomly generated sequences with certain thresholds; we adopt the offline dataset provided in Kim et al. (2023). We use $\delta = 0.5$ and $\lambda = 5$, according to the guidelines in Section 4.

Results. As shown in Figure 2 and Table 1, our method outperforms all baseline approaches. The curve in Figure 2 increases significantly faster than the other methods, indicating that δ-CS effectively trains the policy and generates appropriate queries in each active round. More results are provided in Appendix C.1.

Table 2: Results on AMP with different acquisition functions (UCB, EI). The mean and standard deviation from five runs are reported. Improved results with δ-CS over GFN-AL are marked in **bold**.

	Max	Mean	Diversity	Novelty
COMs	0.930 ± 0.001	0.920 ± 0.000	0.000 ± 0.000	11.869 ± 0.298
DyNA PPO	0.953 ± 0.005	0.941 ± 0.012	15.186 ± 5.109	16.556 ± 3.653
GFN-AL (UCB)	0.936 ± 0.004	0.919 ± 0.005	**28.504 ± 2.691**	19.220 ± 1.369
GFN-AL + δ-CS (UCB)	**0.948 ± 0.015**	**0.938 ± 0.016**	25.379 ± 3.735	**23.551 ± 1.290**
GFN-AL (EI)	0.950 ± 0.002	0.940 ± 0.003	15.576 ± 7.896	21.810 ± 4.165
GFN-AL + δ-CS (EI)	**0.962 ± 0.003**	**0.958 ± 0.004**	**16.631 ± 2.135**	**24.946 ± 4.246**

6.2 DNA sequence design

Task setup. In this task, we aim to generate diverse and novel DNA sequences that maximize the binding affinity to the target transcription factor. The length (L) of the sequence is fixed with 8. The initial dataset $\mathcal{D}_0$ is the bottom 50% in terms of the reward, which results in $32,898$ samples, with the maximum score of 0.439. Though this has been widely used in many studies (Sinai et al., 2020; Jain et al., 2022; Trabucco et al., 2022; Kim et al., 2023), the TF-Bind-8 is easy to optimize, especially due to its size (Sinai et al., 2020). Similar to RNA, we use $\delta = 0.5$ and $\lambda = 5$.

Results. As shown in Table 1, Adalead achieves the highest maximum performance, while δ-CS still outperforms GFN-AL. We believe that Adalead's greedy evolutionary search capability was crucial in the small search space of TF-bind-8. However, in Figure 2, δ-CS demonstrates the best median performance compared to the other baselines; see the mean, diversity, and novelty in Appendix C.2.

6.3 Protein sequence design

We consider two protein sequence design tasks: the green fluorescent protein (GFP; Sarkisyan et al., 2016) and additive adeno-associated virus (AAV; Ogden et al., 2019).

GFP. The objective is to identify protein sequences with high log-fluorescence intensity values.[2] The vocabulary is defined as 20 standard amino acids, i.e., $|\mathcal{V}| = 20$, and the sequence length L is 238; thus, we set δ as 0.05 and λ as 0.1, according to our guideline. The initial datasets are generated by randomly mutating the provided wild-type sequence for each task while filtering out sequences that have higher scores than the wild-type; we obtain the initial dataset with $|\mathcal{D}_0| = 10\ 200$ with a maximum score value of 3.572.

AAV. The aim is to discover sequences that lead to higher gene therapeutic efficiency. The sequences are composed of the 20 standard amino acids with a length of 90, resulting in the search space of 20^{90}. In the same way as in GFP, we collect an initial dataset of 15,307 sequences with a maximum score of 0.500. We use $\delta = 0.05$ and $\lambda = 1$.

Results. Table 1 shows the results of all methods in protein sequence design tasks. Given the combinatorially vast design space with sequence lengths $L = 238$ and 90, most baselines fail to discover new sequences whose score is higher than the maximum of the dataset. In contrast, as depicted in Figure 2, our method finds high-score sequences beyond the dataset, even with a single active round. This underscores the superiority of our search strategy in practical biological sequence design tasks. Full results are provided in Appendix C.3.

6.4 Anti-microbial peptide design

Task setup. The goal is to generate protein sequences with anti-microbial properties (AMP). The vocabulary size $|\mathcal{V}| = 20$, and the sequence length (L) varies across sequences, and we consider sequences of length 50 or lower. For the AMP task, we consider a much larger query batch size for each active round because they can be efficiently synthesized and evaluated (Jain et al., 2022). We set δ as 0.5 with $\lambda = 1$.

Results. The results in Table 3 illustrate that ours consistently gives improved performance over GFN-AL regardless of acquisition function. According to the work from Jain et al. (2022), we

[2]The score is evaluated by ML oracle models. FLEXS uses TAPE (Rao et al., 2019) for evaluation, while Design Bench Transformer (Trabucco et al., 2022) is employed in GFN-AL.

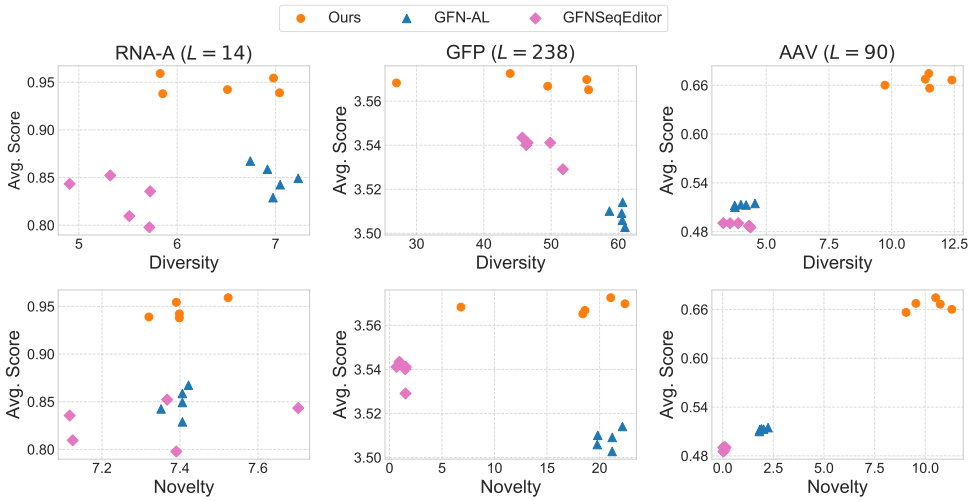


Figure 3: Average score and diversity/novelty with five independent runs. Our method (GFN-AL + δ-CS) consistently approaches Pareto frontier performance. We set $\delta = 0.5$ for short sequences ($L \leq 50$) and set $\delta = 0.05$ for long length sequences ($L > 50$).

also adopt conservative model-based optimization method, (COMs; Trabucco et al., 2021) and on-policy reinforcement learning, DyNA PPO (Angermueller et al., 2020) as baselines. Our method demonstrated significantly higher performance in terms of mean, diversity, and novelty compared to the baselines.

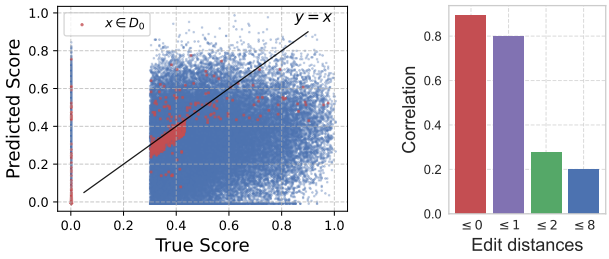
6.5 Achieving Pareto frontier with balancing capability of δ-CS

In this analysis, we demonstrate that δ-CS achieves a balanced search using δ, producing Pareto frontiers or comparable results to the baseline methods: GFN-AL (Jain et al., 2022) and GFNSeqEditor (Ghari et al., 2023). Notably, GFN-AL can be seen as a variant of our method with $\delta = 1$, which fully utilizes the entire trajectory search. This approach is expected to yield high novelty and diversity, but it is also prone to generating high rewards due to the increased risk of out-of-distribution samples affecting the proxy model. GFNSeqEditor, on the other hand, leverages GFlowNets as a prior, editing from a wild-type sequence. It is designed to deliver reliable performance and be more robust to out-of-distribution issues by constraining the search to sequences similar to the wild type. However, unlike δ-CS, GFNSeqEditor does not utilize such obtained samples for training GFlowNets in full trajectory level; GFNSeqEditor is expected to have lower diversity and novelty compared to GFN-AL and δ-CS.

As shown in Fig. 3, GFN-AL generally produces higher diversity and novelty in the RNA and GFP tasks compared to GFNSeqEditor. However, GFNSeqEditor performs better in terms of reward on the large-scale GFP task, whereas GFN-AL struggles due to the lack of a constrained search procedure in such a large combinatorial space. In contrast, δ-CS achieves Pareto-optimal performance compared to both methods, clearly outperforming GFNSeqEditor across six tasks, with higher rewards, diversity, and novelty. For the RNA and GFP tasks, we achieve higher scores than GFN-AL while maintaining similar novelty but slightly lower diversity. In the AAV task, δ-CS shows a distinct Pareto improvement. These results demonstrate that δ-CS provides a beneficial balance by combining conservative search with amortized inference on full trajectories using off-policy GFlowNets training, effectively capturing the strengths of both GFN-AL and GFNSeqEditor.

6.6 Study on proxy failure and conservativeness effect

Task: Hard TF-Bind-8. By modifying the initial dataset distribution and the landscape, we can make a harder version of TF-Bind-8. Specifically, we collect the initial dataset near a certain sequence (considered as a wild-type) while ensuring that the initial sequences have lower scores than the given sequence, which is 0.431. The size of $\mathcal{D}_0$ is 1,024. Furthermore, we modify the landscape to give 0 rewards for sequences with scores lower than 0.3. These features are often observed in pro-



(a) Proxy vs. true scores at $t = 1$ (b) $\rho(f, f_\phi)$

Figure 4: Proxy failure on Hard TF-Bind-8. (a) shows the proxy values (i.e., reward) and true score on the whole data point at the initial round. In (b), the correlation between f and f_ϕ is much higher when the data points are close to the initial dataset ('≤ 0' and '≤ 8' correspond to the initial dataset and the whole sequence space, respectively).

tein design tasks, where the search space is extremely large, e.g., 20^{238} previously–with a limited real-world dataset, and the score often falls to 0.

Proxy failure. As shown in Figure 2, δ-CS gives slightly better performance than GFN-AL, but only marginally. This is because the TF-Bind-8 task is relatively easy to optimize, leading to similar results across methods. To more clearly assess the effectiveness of δ-CS, we conduct several studies on the harder TF-Bind-8 task, which is more difficult to optimize. Figure 4a illustrates the proxy values and true scores for all $x \in \mathcal{X}$ in the first round. While the proxy provides accurate predictions for the initial data points $x \in \mathcal{D}_0$ (represented by the red dots), it produces unreliable predictions for points outside $\mathcal{D}_0$. This supports our hypothesis that the proxy model performs poorly on out-of-distribution data.

Effect of δ conservativeness. Figure 4b illustrates that the correlation between the oracle f and the proxy f_ϕ significantly decreases as data points move farther from the observed dataset. This strongly motivates the use of δ-CS, which constrains the search bounds using δ. By limiting the search to within these constrained edit distances, δ-CS enhances the correlation with the oracle.

Studies on δ. We study on the choice of δ and effectiveness of adaptive $\delta(x, \sigma)$, in Appendix B.2.

7 Conclusions

In this paper, we introduced a novel off-policy sampling method for GFlowNets, called δ-CS, which provides controllable conservativeness through the use of a δ parameter. Additionally, we proposed an adaptive conservativeness approach by adjusting δ for each data point based on prediction uncertainty. We demonstrated the effectiveness of δ-CS in active learning GFlowNets, achieving strong performance across various biological sequence design tasks, consistently outperforming existing baselines.

References

Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, et al. Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science*, 351 (6280):1450–1454, 2016.

David Belanger, Suhani Vora, Zelda Mariet, Ramya Deshpande, David Dohan, Christof Angermueller, Kevin Murphy, Olivier Chapelle, and Lucy Colwell. Biological sequence design using

batched Bayesian optimization. In *NeurIPS 2019 Workshop on Machine Learning and the Physical Sciences*, 2019.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jesse D Bloom and Frances H Arnold. In the light of directed evolution: pathways of adaptive protein evolution. *Proceedings of the National Academy of Sciences*, 106:9995–10000, 2009.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

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

Can Chen, Yingxueff Zhang, Jie Fu, Xue (Steve) Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Can Chen, Yingxue Zhang, Xue Liu, and Mark Coates. Bidirectional learning for offline model-based biological sequence design. In *International Conference on Machine Learning (ICML)*, 2023a.

Can (Sam) Chen, Christopher Beckham, Zixuan Liu, Xue (Steve) Liu, and Chris Pal. Parallel-mentoring for offline model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 76619–76636, 2023b.

Payel Das, Tom Sercu, Kahini Wadhawan, Inkit Padhi, Sebastian Gehrmann, Flaviu Cipcigan, Vijil Chenthamarakshan, Hendrik Strobelt, Cicero Dos Santos, Pin-Yu Chen, et al. Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations. *Nature Biomedical Engineering*, 5(6):613–623, 2021.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. In *Uncertainty in Artificial Intelligence (UAI)*, 2024.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Pouya M. Ghari, Alex Tseng, Gökcen Eraslan, Romain Lopez, Tommaso Biancalani, Gabriele Scalia, and Ehsan Hajiramezanali. Generative flow networks assisted biological sequence editing. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023. URL https://openreview.net/forum?id=9BQ3l8OVru.

Nikolaus Hansen. The CMA evolution strategy: a comparing review. *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, pp. 75–102, 2006.

Alex Hernández-García, Nikita Saxena, Moksh Jain, Cheng-Hao Liu, and Yoshua Bengio. Multi-fidelity active learning with GFlowNets. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=dLaazW9zuF. Expert Certification.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with GFlowNets. In *International Conference on Machine Learning (ICML)*, 2022.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International Conference on Machine Learning (ICML)*, 2023.

Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Diederik P Kingma. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi S. Jaakkola, Regina Barzilay, and Ila R Fiete. Improving protein optimization with smoothed fitness landscapes. In *International Conference on Learning Representations (ICLR)*, 2024.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in neural information processing systems (NeurIPS)*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA package 2.0. *Algorithms for molecular biology*, 6:1–14, 2011.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jonas Mockus. The bayesian approach to global optimization. In *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31–September 4, 1981*, pp. 473–481. Springer, 2005.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Pierce J Ogden, Eric D Kelsic, Sam Sinai, and George M Church. Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, 366(6469): 1139–1143, 2019.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with TAPE. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Paul J Sample, Ban Wang, David W Reid, Vlad Presnyak, Iain J McFadyen, David R Morris, and Georg Seelig. Human 5 UTR design and variant effect prediction from a massively parallel translation assay. *Nature Biotechnology*, 37(7):803–809, 2019.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Jacob Schreiber, Yang Young Lu, and William Stafford Noble. Ledidi: Designing genomic edits that induce functional activity. *BioRxiv*, pp. 2020–05, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Zhenqiao Song and Lei Li. Importance weighted expectation-maximization for protein sequence design. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

Shikha Surana, Nathan Grinsztajn, Timothy Atkinson, Paul Duckworth, and Thomas D Barrett. Overconfident oracles: Limitations of in silico sequence design benchmarking. In *ICML 2024 AI for Science Workshop*, 2024. URL https://openreview.net/forum?id=fPBCnJKXUb.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry Vetrov. Generative flow networks as entropy-regularized RL. *Artificial Intelligence and Statistics (AISTATS)*, 2024.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning (ICML)*, 2021.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-Bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning (ICML)*, 2022.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin. RoMA: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ye Yuan, Can (Sam) Chen, Zixuan Liu, Willie Neiswanger, and Xue (Steve) Liu. Importance-aware co-teaching for offline model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 55718–55733, 2023.

Taeyoung Yun, Sujin Yun, Jaewoo Lee, and Jinkyoo Park. Guided trajectory generation with diffusion models for offline model-based optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Dinghuai Zhang, Jie Fu, Yoshua Bengio, and Aaron Courville. Unifying likelihood-free inference with black-box sequence design and beyond. In *International Conference on Learning Representations (ICLR)*, 2022.

Marc Zimmer. Green fluorescent protein (GFP): applications, structure, and related photophysical behavior. *Chemical reviews*, 102(3):759–782, 2002.

Algorithm 1 Active Learning GFlowNets with δ-CS

Input: Oracle f, initial dataset $\mathcal{D}_0$, active rounds T, query size B, training batch size $2 \times M$.
procedure δ-CS ($\mathcal{D}_{t-1}, M, \delta$) ▷ δ-CS subroutine
  sample high reward data $x_1, \ldots, x_M$ with **rank-based reweighed prior** $P_{\mathcal{D}_{t-1}}(\cdot)$.
  obtain masked data $\tilde{x}_1, \ldots, \tilde{x}_M$ with **noise injection policy** $P_{\text{noise}}(\tilde{\cdot}|\cdot, \delta)$ from $x_1, \ldots, x_M$.
  obtain denoised $\hat{x}_1, \ldots, \hat{x}_M$ with **denoising policy** $P_{\text{denoise}}(\hat{\cdot} \mid \tilde{\cdot}; \theta)$ from $\tilde{x}_1, \ldots, \tilde{x}_M$.
  return $\hat{x}_1, \ldots, \hat{x}_M$.
end procedure
for $t = 1$ to T **do** ▷ Active learning with T rounds
  while proxy training iterations **do** ▷ **Step A:** Proxy training
    train proxy $f_\phi(x)$ with current round dataset $\mathcal{D}_{t-1}$:

$$\mathcal{L}(\phi) = \mathbb{E}_{x \sim P_{\mathcal{D}_{t-1}}(x)} \left[(f(x) - f_\phi(x))^2 \right].$$

  end while
  while policy training iterations **do** ▷ **Step B:** Policy training
    obtain off-policy trajectories $\hat{\tau}_1, \ldots, \hat{\tau}_M$ from $\hat{x}_1, \ldots, \hat{x}_M$ given by δ-CS ($\mathcal{D}_{t-1}, M, \delta$).
    obtain offline trajectory $\tau_1, \ldots, \tau_M$ from $x_1, \ldots, x_M \sim P_{\mathcal{D}_{t-1}}(\tau)$.
    train θ with TB loss over $\hat{\tau}_1, \ldots, \hat{\tau}_M$ and $\tau_1, \ldots, \tau_M$

$$\frac{1}{2M} \sum_{i=1}^{M} \left(\log \frac{Z_\theta P_F(\tau_i; \theta)}{R(x_i; \phi)} \right)^2 + \frac{1}{2M} \sum_{i=1}^{M} \left(\log \frac{Z_\theta P_F(\hat{\tau}_i; \theta)}{R(\hat{x}_i; \phi)} \right)^2.$$

  end while
  obtain query samples $\hat{x}_1, \ldots, \hat{x}_B$ from δ-CS ($\mathcal{D}_{t-1}$, B, δ).
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\hat{x}_i, f(\hat{x}_i))\}_{i=1}^{B}$. ▷ **Step C:** Dataset augmentation with oracle f query
end for

A Implementation detail

A.1 Proxy training

For training proxy models, we follow the procedure of (Jain et al., 2022). We use Adam (Kingma, 2015) optimizer with learning rate 1×10^{-5} and batch size of 256. To prevent over-fitting, we use early stopping using the 10% of the dataset as a validation set and terminate the training procedure if validation loss does not improve for five consecutive iterations.

A.2 Policy training

As described in Section 6, we employ a two-layer long short-term memory (LSTM; Hochreiter & Schmidhuber, 1997) with 512 hidden dimensions. The policy is trained with a learning rate of 5×10^{-4} with a batch size of 256. The learning rate of Z is set as 10^{-3}. The coefficient κ in Equation (2) is set as 0.1 for TF-Bind-8 and AMP with MC dropout, according to Jain et al. (2022), and 1.0 for RNA and protein design with Ensemble following Ren et al. (2022).

A.3 Implementation details of baselines

We adopt open-source code from FLEXS benchmark (Sinai et al., 2020).

- **AdaLead** (Sinai et al., 2020): We use a default settings of hyperparmeters for AdaLead. Specifically, we use a recombination rate of 0.2, mutation rate of $1/L$, where L is sequence length, and threshold $\tau = 0.05$.
- **DbAS** (Brookes & Listgarten, 2018): We implement DbAS with variational autoencoder (VAE; Kingma & Welling, 2014) as the generator. The input is a one-hot encoding vector, and the output latent dimension is 2. In each cycle, DbAS starts by training the VAE with the top 20% sequences in terms of the score.
- **CbAS** (Brookes et al., 2019): Similar to DbAS, we implement CbAS with VAE. The main difference from DbAS is that we select top 20% sequences with the weights $p(\boldsymbol{x}|\boldsymbol{z}, \theta^{(0)})/q(\boldsymbol{x}|\boldsymbol{z}, \phi^{(t)})$,

where $p(\cdot; \theta^{(0)})$ is trained with the ground-truth samples and $q(\cdot; \phi^{(t)})$ is trained on the generated sequences over t training rounds.

- **DyNA PPO** (Angermueller et al., 2020): We closely follow the algorithm presented in (Angermueller et al., 2020). For a fair comparison, we use CNN ensembles to parameterize the proxy model instead of suggested architectures.
- **CMA-ES** (Hansen, 2006): We implement a covariance matrix adaptation evolution strategy (CMA-ES) for sequence generation. As the generated samples from CMA-ES are continuous, we convert the continuous vectors into one-hot representation by computing the argmax at each sequence position.
- **BO** (Snoek et al., 2012): We use classical GP-BO algorithm for all tasks. For Gaussian Process Regressor (GPR), we use a default setting from the `sklearn` library. For the acquisition function, they use Thompson sampling (Russo et al., 2018).

Furthermore, we employ GFN-AL and GFNSeqEditor. We adopt the original implementation and setup for TF-Bind-8 and AMP. For newly added tasks, we report better results among the original MLP policy and the LSTM policy. Note that GFP in FLEXS is different from the one employed in GFA-AL; we treat this as a new task based on the observation in the work from Surana et al. (2024).

- **GFN-AL** (Jain et al., 2022): We strictly follow hyperparameters of the original code in they conduct experiments on TF-Bind-8 and AMP. The proxy is parameterized using an MLP with two layers of 2,048 hidden. For the policy, a 2-layer MLP with 2,048 hidden dimensions is used, but we also test it with a 2-layer LSTM policy.
- **GFNSeqEditor** (Ghari et al., 2023): We implemented the editing procedure on top of the GFN-AL. Note that GFNSeqEditor does not utilize the proxy model, so the GFlowNets policy is trained using offline data only with the same policy training procedure of GFN-AL. GFNSeqEditor can also implicitly control the edit percentage with its hyperparameters, which are set $\delta = 0.01, \sigma = 0.0001, \lambda = 0.1$ in this study. Note that δ is not the conservativeness parameter.

Table 3: Results on AMP with different acquisition functions (UCB, EI). The mean and standard deviation from five runs are reported. Improved results with δ-CS over GFN-AL are marked in **bold**.

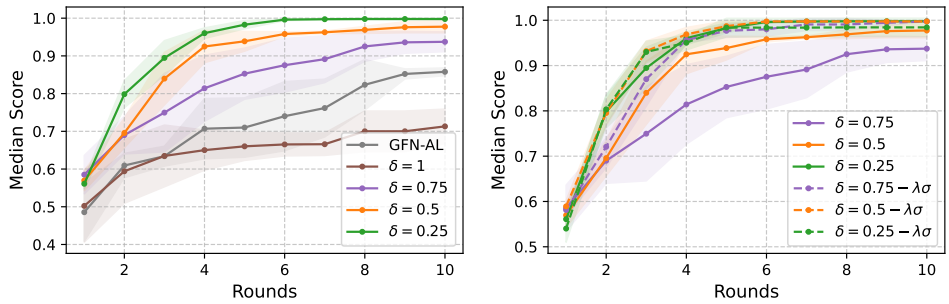
	Max	Mean	Diversity	Novelty
COMs	0.930 ± 0.001	0.920 ± 0.000	0.000 ± 0.000	11.869 ± 0.298
DyNA PPO	0.953 ± 0.005	0.941 ± 0.012	15.186 ± 5.109	16.556 ± 3.653
GFN-AL (UCB)	0.936 ± 0.004	0.919 ± 0.005	**28.504 ± 2.691**	19.220 ± 1.369
GFN-AL + δ-CS (UCB)	**0.948 ± 0.015**	**0.938 ± 0.016**	25.379 ± 3.735	**23.551 ± 1.290**
GFN-AL (EI)	0.950 ± 0.002	0.940 ± 0.003	15.576 ± 7.896	21.810 ± 4.165
GFN-AL + δ-CS (EI)	**0.962 ± 0.003**	**0.958 ± 0.004**	**16.631 ± 2.135**	**24.946 ± 4.246**

B Further studies

B.1 Anti-microbial peptide design

Task setup. The goal is to generate protein sequences with anti-microbial properties (AMP). The vocabulary size $|\mathcal{V}| = 20$, and the sequence length (L) varies across sequences, and we consider sequences of length 50 or lower. For the AMP task, we consider a much larger query batch size for each active round because they can be efficiently synthesized and evaluated (Jain et al., 2022). We set δ as 0.5 with $\lambda = 1$.

B.2 Studies on effect of δ on Hard TF-Bind-8



(a) Median scores with various fixed δ

(b) Effect of adaptive control

Figure 5: Median score over rounds on Hard TF-Bind-8.

To verify its effectiveness and give intuition about how to set δ, we conduct experiments with various δ in Hard TF-Bind-8. The results show that δ-CS with $\delta < 1$ can significantly outperform GFN-AL by searching for data points that correlate better with the oracle. In the Hard TF-Bind-8 task, a more conservative search with $\delta = 0.25$ is beneficial since the proxy is unreliable in the early rounds. In particular, ours with $\delta = 1$ means the full on-policy search (no conservativenss). The performance differences between $\delta = 1$ and GFN-AL came from ϵ-noisy behavior policy, which selects random actions with a probability of 0.001, in GFN-AL. Furthermore, using adaptive $\delta(x, \sigma)$ mostly gives the improved scores as depicted in Figure 5b.

We further examine the effect of adaptive δ adjustment on RNA in the following section.

B.3 Studies on the effect of adaptive δ on RNA-A

We examine the effects of proxy uncertainty-based δ. In RNA, the average proxy standard deviation $\bar{\sigma}$ at the initial round is observed as 0.005 to 0.012. Therefore, we set $\lambda = 5$ to roughly make $\lambda\bar{\sigma} \approx 1/L$, where $L = 14$. As illustrated in Figure 6, $\delta(x; \sigma)$ consistently gives the higher score. However, the constant $\delta = 5$ still outperforms all baselines, exhibiting the robustness of δ-CS.

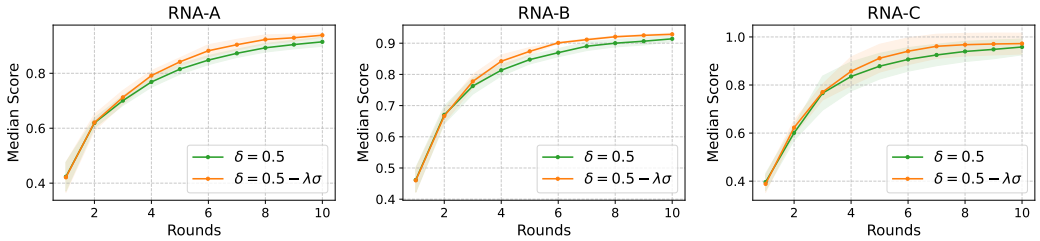


Figure 6: Effect of adaptive control on RNA

B.4 Balancing capability of δ on RNA

Similar to Section 6.5, we also verify the balancing capability of δ-CS on RNA-B and RNA-C. The δ is set as 0.5.

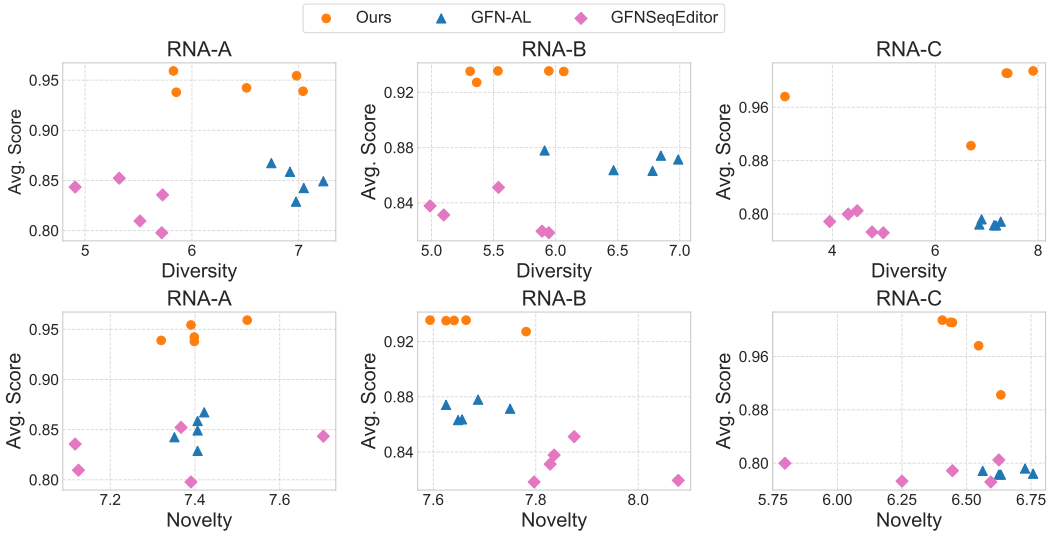


Figure 7: Average score and diversity/novelty with five independent runs on RNA design tasks.

B.5 Studies on rank-based reweighted sampling in proxy training

The rank-based reweighing also can be used in proxy training, i.e., $x \sim P_{\mathcal{D}_{t-1}}(x; k)$, where k is a reweighting factor and fixed as 0.01 in this work. The results show that rank-based reweighted proxy training improves performance mostly. However, the gap is small, and δ-CS still works well even without reweighting.

Table 4: Ablation studies of rank-based reweighted proxy training

		Max	Median	Mean	Diversity	Novelty
RNA-A	with rank-based	1.055 ± 0.000	0.939 ± 0.008	0.947 ± 0.009	6.442 ± 0.525	7.406 ± 0.066
	without rank-based	1.049 ± 0.010	0.936 ± 0.016	0.944 ± 0.015	5.782 ± 0.697	7.397 ± 0.098
RNA-B	with rank-based	1.014 ± 0.001	0.929 ± 0.004	0.934 ± 0.003	5.644 ± 0.307	7.661 ± 0.064
	without rank-based	1.009 ± 0.008	0.932 ± 0.012	0.938 ± 0.012	6.252 ± 0.291	7.673 ± 0.033
RNA-C	with rank-based	1.094 ± 0.045	0.972 ± 0.043	0.983 ± 0.043	6.493 ± 1.751	6.494 ± 0.084
	without rank-based	1.097 ± 0.022	0.958 ± 0.029	0.965 ± 0.031	5.472 ± 1.921	6.464 ± 0.192

C Full results of main results

C.1 Full results of RNA sequence design

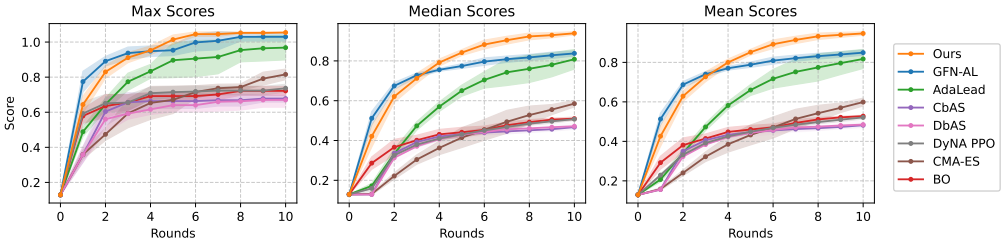


Figure 8: The max, median, and mean curve over rounds in RNA-A

Table 5: The results of RNA-A after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	0.968 ± 0.070	0.808 ± 0.049	0.817 ± 0.048	3.518 ± 0.446	6.888 ± 0.426
BO	0.722 ± 0.025	0.510 ± 0.008	0.528 ± 0.004	$\mathbf{9.531} \pm \mathbf{0.062}$	5.842 ± 0.083
CMA-ES	0.816 ± 0.030	0.585 ± 0.016	0.599 ± 0.020	5.747 ± 0.110	6.373 ± 0.159
CbAS	0.678 ± 0.020	0.467 ± 0.009	0.481 ± 0.008	9.457 ± 0.189	5.428 ± 0.078
DbAS	0.670 ± 0.041	0.472 ± 0.016	0.485 ± 0.015	9.483 ± 0.100	5.450 ± 0.132
DyNA PPO	0.737 ± 0.022	0.507 ± 0.007	0.521 ± 0.009	8.889 ± 0.034	5.828 ± 0.095
GFN-AL	1.030 ± 0.024	0.838 ± 0.013	0.849 ± 0.013	6.983 ± 0.159	7.398 ± 0.024
GFN-AL + δ-CS	$\mathbf{1.055} \pm \mathbf{0.000}$	$\mathbf{0.939} \pm \mathbf{0.008}$	$\mathbf{0.947} \pm \mathbf{0.009}$	6.442 ± 0.525	$\mathbf{7.406} \pm \mathbf{0.066}$

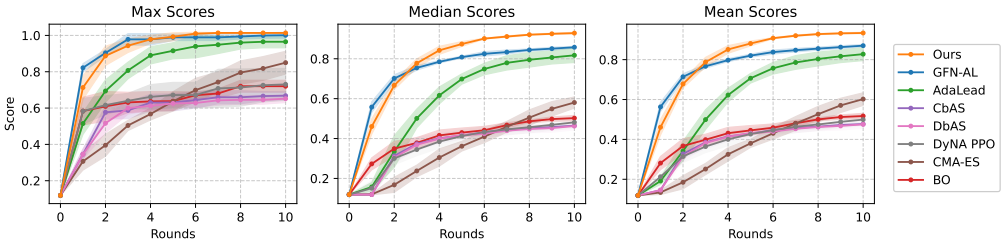


Figure 9: The max, median, and mean curve over rounds in RNA-B

Table 6: The results of RNA-B after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	0.965 ± 0.033	0.817 ± 0.036	0.828 ± 0.032	3.334 ± 0.423	7.441 ± 0.135
BO	0.720 ± 0.032	0.502 ± 0.013	0.517 ± 0.014	$\mathbf{9.495} \pm \mathbf{0.103}$	5.903 ± 0.116
CMA-ES	0.850 ± 0.063	0.581 ± 0.028	0.602 ± 0.032	5.568 ± 0.365	6.480 ± 0.200
CbAS	0.668 ± 0.021	0.465 ± 0.005	0.477 ± 0.004	9.234 ± 0.356	5.523 ± 0.083
DbAS	0.652 ± 0.021	0.463 ± 0.019	0.475 ± 0.019	9.019 ± 0.648	5.537 ± 0.150
DyNA PPO	0.730 ± 0.088	0.481 ± 0.028	0.499 ± 0.029	8.978 ± 0.196	5.839 ± 0.198
GFN-AL	1.001 ± 0.016	0.858 ± 0.004	0.870 ± 0.006	6.599 ± 0.384	$\mathbf{7.673} \pm \mathbf{0.043}$
GFN-AL + δ-CS	$\mathbf{1.014} \pm \mathbf{0.001}$	$\mathbf{0.929} \pm \mathbf{0.004}$	$\mathbf{0.934} \pm \mathbf{0.003}$	5.644 ± 0.307	7.661 ± 0.064

C.2 Full results of TF-Bind-8

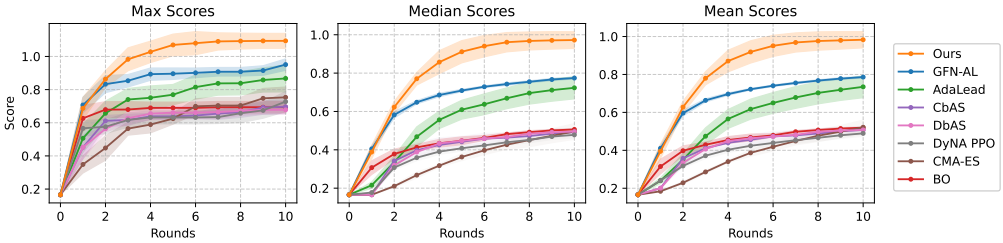


Figure 10: The max, median, and mean curve over rounds in RNA-C

Table 7: The results of RNA-C after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	0.867 ± 0.081	0.723 ± 0.057	0.735 ± 0.057	3.893 ± 0.444	5.856 ± 0.515
BO	0.694 ± 0.034	0.506 ± 0.003	0.519 ± 0.003	9.714 ± 0.054	5.430 ± 0.043
CMA-ES	0.753 ± 0.062	0.496 ± 0.041	0.521 ± 0.037	5.581 ± 0.399	5.019 ± 0.294
CbAS	0.696 ± 0.041	0.492 ± 0.018	0.507 ± 0.017	$\mathbf{9.518 \pm 0.310}$	5.033 ± 0.086
DbAS	0.678 ± 0.025	0.495 ± 0.010	0.508 ± 0.011	9.249 ± 0.414	5.128 ± 0.153
DyNA PPO	0.728 ± 0.060	0.478 ± 0.015	0.489 ± 0.015	9.246 ± 0.086	5.306 ± 0.124
GNF-AL	0.951 ± 0.034	0.774 ± 0.004	0.786 ± 0.004	7.072 ± 0.163	$\mathbf{6.661 \pm 0.071}$
GFN-AL + δ-CS	$\mathbf{1.094 \pm 0.045}$	$\mathbf{0.972 \pm 0.043}$	$\mathbf{0.983 \pm 0.043}$	6.493 ± 1.751	6.494 ± 0.084

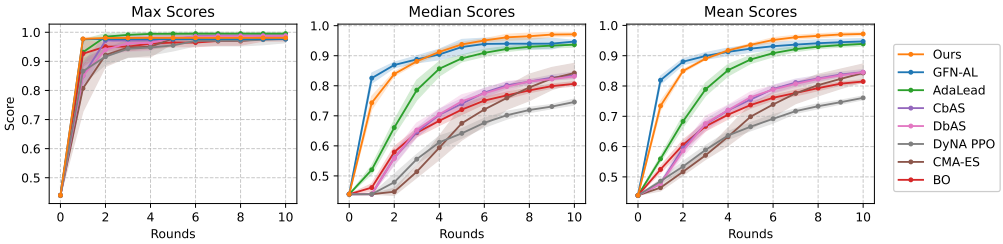


Figure 11: The max, median, and mean curve over rounds in TF-Bind-8

Table 8: The results of TF-Bind-8 after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	$\mathbf{0.995 \pm 0.004}$	0.937 ± 0.008	0.939 ± 0.007	3.506 ± 0.267	1.194 ± 0.035
BO	0.977 ± 0.008	0.806 ± 0.007	0.815 ± 0.005	4.824 ± 0.074	1.144 ± 0.029
CMA-ES	0.986 ± 0.008	0.843 ± 0.032	0.843 ± 0.030	3.617 ± 0.321	1.130 ± 0.083
CbAS	0.988 ± 0.004	0.835 ± 0.011	0.845 ± 0.009	4.662 ± 0.079	1.134 ± 0.021
DbAS	0.987 ± 0.004	0.831 ± 0.005	0.845 ± 0.005	4.694 ± 0.056	1.141 ± 0.047
DyNA PPO	0.977 ± 0.013	0.746 ± 0.010	0.761 ± 0.006	4.430 ± 0.030	1.120 ± 0.021
GFN-AL	0.976 ± 0.002	0.947 ± 0.004	0.947 ± 0.009	3.158 ± 0.166	$\mathbf{2.409 \pm 0.071}$
GFN-AL + δ-CS	0.981 ± 0.002	$\mathbf{0.971 \pm 0.006}$	$\mathbf{0.972 \pm 0.005}$	$\mathbf{1.277 \pm 0.182}$	2.237 ± 0.356

C.3 Full results of Protein design

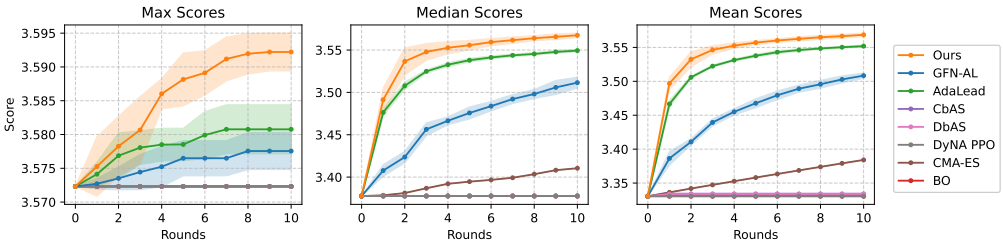


Figure 12: The max, median, and mean curve over rounds in GFP

Table 9: The results of GFP after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	3.581 ± 0.004	3.549 ± 0.002	3.552 ± 0.002	47.237 ± 1.213	1.467 ± 0.094
BO	3.572 ± 0.000	3.378 ± 0.000	3.331 ± 0.000	**62.955 ± 0.000**	0.000 ± 0.000
CMA-ES	3.572 ± 0.000	3.410 ± 0.000	3.384 ± 0.000	58.299 ± 0.000	0.000 ± 0.000
CbAS	3.572 ± 0.000	3.378 ± 0.000	3.334 ± 0.002	62.926 ± 0.139	0.009 ± 0.012
DbAS	3.572 ± 0.000	3.378 ± 0.000	3.334 ± 0.002	62.926 ± 0.139	0.009 ± 0.012
DyNA PPO	3.572 ± 0.000	3.378 ± 0.000	3.331 ± 0.000	**62.955 ± 0.000**	0.000 ± 0.000
GFN-AL	3.578 ± 0.003	3.511 ± 0.006	3.508 ± 0.004	60.278 ± 0.819	**20.837 ± 0.916**
GFN-AL + δ-CS	**3.592 ± 0.003**	**3.567 ± 0.003**	**3.569 ± 0.003**	46.255 ± 10.534	17.459 ± 5.538

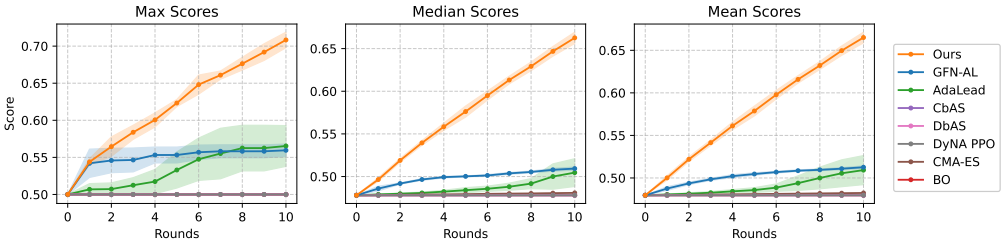


Figure 13: The max, median, and mean curve over rounds in AAV

Table 10: The results of AAV after ten rounds.

	Max	Median	Mean	Diversity	Novelty
AdaLead	0.565 ± 0.027	0.505 ± 0.016	0.509 ± 0.017	5.693 ± 0.946	2.133 ± 1.266
BO	0.500 ± 0.000	0.478 ± 0.000	0.480 ± 0.000	4.536 ± 0.000	0.000 ± 0.000
CMA-ES	0.500 ± 0.000	0.481 ± 0.000	0.482 ± 0.000	4.148 ± 0.000	0.000 ± 0.000
CbAS	0.500 ± 0.000	0.478 ± 0.000	0.480 ± 0.000	4.545 ± 0.018	0.002 ± 0.003
DbAS	0.500 ± 0.000	0.478 ± 0.000	0.480 ± 0.000	4.545 ± 0.018	0.002 ± 0.003
DyNA PPO	0.500 ± 0.000	0.478 ± 0.000	0.480 ± 0.000	4.536 ± 0.000	0.000 ± 0.000
GFN-AL	0.560 ± 0.008	0.509 ± 0.002	0.513 ± 0.002	4.044 ± 0.303	1.966 ± 0.157
GFN-AL + δ-CS	**0.708 ± 0.010**	**0.663 ± 0.007**	**0.665 ± 0.006**	**11.296 ± 0.865**	**10.233 ± 0.822**