GFlowNet Assisted Biological Sequence Editing

Pouya M. Ghari[*]
University of California Irvine

Alex M. Tseng
Genentech

Gökcen Eraslan
Genentech

Romain Lopez
Genentech, Stanford University

Tommaso Biancalani
Genentech

Gabriele Scalia
Genentech

Ehsan Hajiramezanali[†]
Genentech

Abstract

Editing biological sequences has extensive applications in synthetic biology and medicine, such as designing regulatory elements for nucleic-acid therapeutics and treating genetic disorders. The primary objective in biological-sequence editing is to determine the optimal modifications to a sequence which augment certain biological properties while adhering to a minimal number of alterations to ensure predictability and potentially support safety. In this paper, we propose GFNSeqEditor, a novel biological-sequence editing algorithm which builds on the recently proposed area of generative flow networks (GFlowNets). Our proposed GFNSeqEditor identifies elements within a starting seed sequence that may compromise a desired biological property. Then, using a learned stochastic policy, the algorithm makes edits at these identified locations, offering diverse modifications for each sequence to enhance the desired property. The number of edits can be regulated through specific hyperparameters. We conducted extensive experiments on a range of real-world datasets and biological applications, and our results underscore the superior performance of our proposed algorithm compared to existing state-of-the-art sequence editing methods.

1 Introduction

Editing biological sequences has a multitude of applications in biology, medicine, and biotechnology. For instance, gene editing serves as a tool to elucidate the role of individual gene products in diseases [28] and offers the potential to rectify genetic mutations in afflicted tissues and cells for therapeutic interventions [9]. The primary objective in biological-sequence editing is to enhance specific biological attributes of a starting seed sequence, while minimizing the number of edits. This reduction in the number of alterations not only has the potential to improve safety but also facilitates the predictability and precision of modification outcomes.

Existing methodologies that leverage generative modeling in the context of biological sequences have predominantly concentrated on *de novo* generation of sequences with desired properties [52, 61, 3]. A common feature of these approaches is generating entirely new sequences from scratch. As a result, there is an inherent risk of deviating significantly from naturally occurring sequences, compromising safety (e.g., the risk of designing sequences that might trigger an immune response) and predictability (e.g., obtaining misleading predictions from models that are trained on genomic sequences due to

[*]Work has been done while interning at Genentech
[†]Corresponding author: `hajiramezanali.ehsan@gene.com`.

out-of-distribution effects). Despite the paramount importance of editing biological sequences, there has been a noticeable scarcity of research leveraging generative modeling to address this aspect specifically.

Generative flow networks (GFlowNets) [5, 6], a generative approach recognized for their ability to sequentially generate new objects, have shown remarkable performance in generating novel biological sequences from scratch [19, 30]. Drawing inspiration from the emerging field of GFlowNets, this paper introduces a novel biological-sequence editing algorithm: *GFNSeqEditor*. GFNSeqEditor assesses the potential for significant property enhancement within a given seed sequence by iteratively identifying and subsequently editing specific positions in the input sequence. More precisely, using the trained flow function, GFNSeqEditor first identifies positions in the seed sequence that require editing. Then, it constructs a stochastic policy using the flow function to select a substitution from the available options for the identified positions. Our stochastic approach empowers GFNSeqEditor to generate a diverse set of edited sequences for each input sequence, which, due to the diverse nature of biological targets, is an important consideration in biological sequence design [34, 19].

In summary, this paper makes the following contributions:

- We introduce GFNSeqEditor, a novel sequence-editing method which identifies and edits positions within a given sequence. GFNSeqEditor generates diverse edits for each input sequence based on a stochastic policy.
- We theoretically analyze the properties of the sequences edited through GFNSeqEditor, deriving lower and upper bounds on the property of edited sequences. Additionally, we demonstrate that the lower and upper bounds for the number of edits performed by GFNSeqEditor can be controlled through the adjustment of hyperparameters (Subsection 4.3).
- We conduct experiments across various DNA and protein sequence editing tasks, showcasing GFNSeqEditor's remarkable efficiency in enhancing properties with a reduced number of edits when compared to existing state-of-the-art methods. (Subsection 5.1).
- We highlight the versatility of GFNSeqEditor, which can be employed not only for sequence editing but also alongside biological-sequence generation models to produce novel sequences with improved properties and increased diversity (Subsection 5.2).
- We demonstrate the usage of GFNSeqEditor for sequence length reduction, allowing the creation of new, relatively shorter sequences by combining pairs of long and short sequences (Subsection 5.3).

2 Related Works

De Novo Sequence Design. The generation of biological sequences has been tackled using a diverse range of methods, including reinforcement learning [1], Bayesian optimization [51], deep generative models for search and sampling [18], generative adversarial networks [61], diffusion models [3], model-based optimization approaches [52, 7], adaptive evolutionary strategies [16, 49], likelihood-free inference [55], and surrogate-based black-box optimization [10], and GFlowNet [19]. It is important to note that all these sequence-generation methods generate sequences from scratch. However, *ab initio* generation carries the risk of *deviating too significantly* from naturally occurring sequences, which can compromise safety and predictability. In contrast, our proposed method enhances a target property while maintaining the similarity to seed sequences (e.g., naturally occurring sequences), thus improving predictability and potentially enhancing safety.

Sequence Editing. Traditional approaches commonly employed for biological sequence editing are evolution-based methods, where—over many iterations—a starting "seed" sequence is randomly mutated, retaining only the best sequences (i.e., highest desired property) for the next round [2, 46, 50, 41]. These approaches have several important limitations. First, they require the evaluation of numerous candidate sequences at every iteration. This computational demand can become prohibitively expensive, particularly for lengthy sequences. Additionally, evolution-based methods heavily rely on evaluations provided by a proxy model capable of assessing the properties of unseen sequences; the efficacy of these methods is thus limited by the reliability of the underlying proxy. Moreover, these methods may require repeated rounds of interactions with the lab [41], which can be costly and time-consuming.

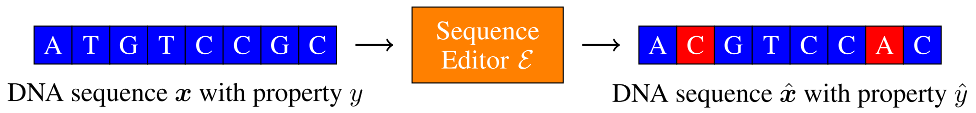
A T G T C C G C
Sequence Editor $\mathcal{E}$
A C G T C C A C
DNA sequence $\boldsymbol{x}$ with property y
DNA sequence $\hat{\boldsymbol{x}}$ with property $\hat{y}$

Figure 1: An example of editing the DNA sequence `'ATGTCCGC'`. The goal is to make a limited number of edits to maximize the property $\hat{y}$. Each token in the sequence in this example is called a *base* and can be any of [`'A'`, `'C'`, `'T'`, `'G'`]. The editor function $\mathcal{E}$ accepts the initial sequence as an input and determines that the second and seventh bases require editing (highlighted in red). Then, $\mathcal{E}$ modifies the bases at these identified locations to improve the property value.

Beyond evolution-based methods, a handful of optimization-based methods have been proposed by [42, 48, 21]. By treating sequence editing as an optimization task, Ledidi [42] learns to perturb specific positions within a given sequence. Utilizing Bayesian optimization, LaMBO [48] generates new sequences by optimizing a batch of starting seed sequences. Building upon the LaMBO framework, MOGFN-AL [21] leverages GFlowNets to generate candidates in each round of Bayesian optimization loop, improving computational efficiency compared to LaMBO. Akin to evolution-based models, these optimization-based methods require the evaluation of unseen sequences. Consequently, their effectiveness is contingent on the quality of the proxy model, which can compromise their performance if the proxy model lacks sufficient generalizability for unseen sequences. Furthermore, both evolution-based and optimization-based methods perform local searches given either a single seed sequence or a batch of seed sequences. Thus, these methods face issues related to low sample efficiency. In contrast, GFNSeqEditor relies on a pre-trained flow function that amortizes the search cost over the learning process, allocating probability mass across the entire space to facilitate exploration and diversity. Furthermore, GFNSeqEditor can be employed for editing without necessitating the evaluation of unseen sequence properties. Theoretical analysis presented in this paper establishes that the bounds of edited sequence rewards, property improvement, and the number of edits can be effectively regulated through GFNSeqEditor hyperparameters. Therefore, GFNSeqEditor offers increased reliability and operational suitability in comparison to counterparts lacking a robust theoretical analysis.

We provide an extensive overview of the related literature, with additional discussion available in the Appendix F.

3 Preliminaries and Problem Statement

Let $\boldsymbol{x}$ be a biological sequence with property y. For example, $\boldsymbol{x}$ may be a DNA sequence, and y may be the likelihood it binds to a particular protein of interest. The present paper considers the problem of searching for edits in $\boldsymbol{x}$ to improve y. To this end, the goal is to learn an editor function $\mathcal{E}(\cdot)$ which accepts a sequence $\boldsymbol{x}$ and outputs the edited sequence $\mathcal{E}(\boldsymbol{x}) = \hat{\boldsymbol{x}}$ with property $\hat{y}$. The editor function $\mathcal{E}(\cdot)$ should maximize $\hat{y}$, while at the same time minimizing the number of edits between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. To achieve this goal, we propose GFNSeqEditor. GFNSeqEditor first identifies positions in a given biological sequence such that editing those positions leads to considerable improvement in the property of the sequence. Then, the learned editor function $\mathcal{E}$ edits these identified locations (Figure 1). GFNSeqEditor uses a trained GFlowNet [5, 6] to identify positions that require editing and subsequently generate edits for those positions. The following Subsections present preliminaries on GFlowNets.

3.1 Generative Flow Networks

Generative Flow Networks (GFlowNets) [5, 6] learn a stochastic policy $\pi(\cdot)$ to sequentially construct a discrete object $\boldsymbol{x}$. Let $\mathcal{X}$ be the space of discrete objects $\boldsymbol{x}$. It is assumed that the space $\mathcal{X}$ is compositional, meaning that an object $\boldsymbol{x}$ can be constructed using a sequence of actions taken from an action set $\mathbb{A}$. At each step t, given a partially constructed object $\boldsymbol{s}_t$, GFlowNet samples an action a_{t+1} from the set $\mathbb{A}$ using the stochastic policy $\pi(\cdot|\boldsymbol{s}_t)$. Then, GFlowNet appends a_{t+1} to $\boldsymbol{s}_t$ to obtain $\boldsymbol{s}_{t+1}$. In this context, $\boldsymbol{s}_t$ can be viewed as the state at step t. The above procedure continues until reaching a terminating state, which yields the fully constructed object $\boldsymbol{x}$. To construct an object $\boldsymbol{x}$, the GFlowNet starts from an initial empty state $\boldsymbol{s}_0$, and applying actions sequentially, all fully constructed objects must end in a special final state $\boldsymbol{s}_f$. Therefore, the trajectory of states to construct an object $\boldsymbol{x}$ can be written as $\tau_{\boldsymbol{x}} = (\boldsymbol{s}_0 \to \boldsymbol{s}_1 \to \cdots \to \boldsymbol{x} \to \boldsymbol{s}_f)$. Let $\mathbb{T}$ be the set of all possible trajectories. Furthermore, let $R(\cdot) : \mathcal{X} \to \mathbb{R}^+$ be a non-negative reward function defined on $\mathcal{X}$. The goal of GFlowNet is to learn a stochastic policy $\pi(\cdot)$ such that $\pi(\boldsymbol{x}) \propto R(\boldsymbol{x})$. This means that the GFlowNet learns a stochastic policy $\pi(\cdot)$ to generate an object $\boldsymbol{x}$ with a probability proportional to its reward.

As described later, to obtain the policy $\pi(\cdot)$, the GFlowNet uses trajectory flow $F : \mathbb{T} \to \mathbb{R}^+$. The trajectory flow $F(\tau)$ assigns a probability mass to the trajectory τ. Then, the *edge flow* from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ is defined as $F(\boldsymbol{s} \to \boldsymbol{s}') = \sum_{\forall \tau : \boldsymbol{s} \to \boldsymbol{s}' \in \tau} F(\tau)$. Moreover, the *state flow* is defined as $F(\boldsymbol{s}) = \sum_{\forall \tau : \boldsymbol{s} \in \tau} F(\tau)$. The trajectory flow $F(\cdot)$ induces a probability measure $P_F(\cdot)$ over completed trajectories that can be expressed as $P_F(\tau) = \frac{F(\tau)}{Z}$ where $Z = \sum_{\forall \tau \in \mathbb{T}} F(\tau)$ represents the total flow. The probability of visiting state $\boldsymbol{s}$ can be written as $P_F(\boldsymbol{s}) = \frac{\sum_{\forall \tau \in \mathbb{T} : \boldsymbol{s} \in \tau} F(\tau)}{Z}$. Then, the forward transition probability from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ can be obtained as $P_F(\boldsymbol{s}'|\boldsymbol{s}) = \frac{F(\boldsymbol{s} \to \boldsymbol{s}')}{F(\boldsymbol{s})}$. The trajectory flow $F(\cdot)$ is called a consistent flow if for any state $\boldsymbol{s}$ it satisfies $\sum_{\forall \boldsymbol{s}' : \boldsymbol{s}' \to \boldsymbol{s}} F(\boldsymbol{s}' \to \boldsymbol{s}) = \sum_{\forall \boldsymbol{s}'' : \boldsymbol{s} \to \boldsymbol{s}''} F(\boldsymbol{s} \to \boldsymbol{s}'')$, which constitutes that the in-flow and out-flow of state $\boldsymbol{s}$ are equal. [5] shows that if $F(\cdot)$ is a consistent flow such that the terminal flow is set as reward (i.e. $F(\boldsymbol{x} \to \boldsymbol{s}_f) = R(\boldsymbol{x})$), the policy $\pi(\cdot)$ defined as $\pi(\boldsymbol{s}'|\boldsymbol{s}) = P_F(\boldsymbol{s}'|\boldsymbol{s})$ satisfies $\pi(\boldsymbol{x}) = \frac{R(\boldsymbol{x})}{Z}$ which means that the policy $\pi(\cdot)$ samples an object $\boldsymbol{x}$ proportional to its reward.

3.2 Training GFlowNet Models

In order to learn the policy $\pi(\cdot)$, a GFlowNet model approximates trajectory flow with a flow function $F_{\boldsymbol{\theta}}(\cdot)$ where $\boldsymbol{\theta}$ includes learnable parameters of the flow function. To learn the flow function that can provide consistency condition, [5] formulates flow-matching loss function as follows:

$$\mathcal{L}_{\text{FM}}(\boldsymbol{s}; \boldsymbol{\theta}) = \left(\log \frac{\sum_{\forall \boldsymbol{s}' : \boldsymbol{s}' \to \boldsymbol{s}} F_{\boldsymbol{\theta}}(\boldsymbol{s}' \to \boldsymbol{s})}{\sum_{\forall \boldsymbol{s}'' : \boldsymbol{s} \to \boldsymbol{s}''} F_{\boldsymbol{\theta}}(\boldsymbol{s} \to \boldsymbol{s}'')} \right)^2 . \tag{1}$$

Moreover, as an alternative objective function, [31] introduces trajectory balance as:

$$\mathcal{L}_{\text{TB}}(\boldsymbol{s}; \boldsymbol{\theta}) = \left(\log \frac{Z_{\boldsymbol{\theta}} \prod_{\boldsymbol{s} \to \boldsymbol{s}'} P_{F_{\boldsymbol{\theta}}}(\boldsymbol{s}'|\boldsymbol{s})}{R(\boldsymbol{x})} \right)^2 \tag{2}$$

where $Z_{\boldsymbol{\theta}}$ is a learnable parameter. The trajectory-balance objective function in (2) can accelerate training GFlowNets and provide robustness to long trajectories. Given a training dataset, optimization techniques such as stochastic gradient descent can be applied to objective functions in (1) and (2) to train the GFlowNet model. We use trajectory balance in this paper due to its well-documented performance. Furthermore, it is worth noting that generating sequences in an autoregressive fashion using GFlowNet involves only one path to generate a particular sequence. In such cases, generating biological sequences with GFlowNet can be viewed as a Soft-Q-Learning [15, 13, 33] and path consistency learning (PCL) [35] problem.

4 Sequence Editing with GFlowNet

To edit a given sequence $\boldsymbol{x}$, we propose identifying *sub-optimal* positions of $\boldsymbol{x}$ such that editing them can lead to considerable improvement in the sequence property. Assume that the flow function $F_{\boldsymbol{\theta}}(\cdot)$ is trained on available offline training data. GFNSeqEditor uses the trained GFlowNet's flow function $F_{\boldsymbol{\theta}}(\cdot)$ to identify sub-optimal positions of $\boldsymbol{x}$, and subsequently replace the sub-optimal parts with newly sampled edits based on the stochastic policy $\pi(\cdot)$.

4.1 Sub-Optimal-Position Identification

This Subsection provides intuition on how GFNSeqEditor uses a pre-trained flow function $F_{\boldsymbol{\theta}}(\cdot)$ to identify sub-optimal positions in a sequence $\boldsymbol{x}$ to edit. Let x_t and $\boldsymbol{x}_{:t}$ denote the t-th element and the first t elements in the sequence $\boldsymbol{x}$, respectively. For example, in the DNA sequence $\boldsymbol{x}$ = 'ATGTCCGC', we have x_2 = 'T' and $\boldsymbol{x}_{:2}$ = 'AT'. GFNSeqEditor constructs edited sequences token by token, and for each position $t + 1$, it examines whether x_{t+1} should be edited or not. Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, given $\boldsymbol{x}_{:t}$, GFlowNet evaluates the average reward obtained by appending any possible token to $\boldsymbol{x}_{:t}$. In this context, each token can be viewed as an action. Let $\boldsymbol{x}_{:t} + a$ denotes the expanded $\boldsymbol{x}_{:t}$ by appending token a. Let $\mathbb{A}$ represent the available action set. For each $a \in \mathbb{A}$, using the state flow $F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + a)$, the value of action a given $\boldsymbol{x}_{:t}$ can be evaluated. As discussed in Section 3, the state flow $F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + a)$ is proportional to the total reward of all possible sequences that have $\boldsymbol{x}_{:t} + a$ as their prefix. Therefore, $F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + a_1) > F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + a_2)$ indicates that taking action a_1 instead of action a_2 can lead to obtaining better candidates for the final sequence. We can leverage this property of the flow function $F_{\boldsymbol{\theta}}(\cdot)$ to examine if x_{t+1} is sub-optimal or not. If the reward resulting from having x_{t+1} in the seed sequence is evaluated by $F_{\boldsymbol{\theta}}(\cdot)$ to be relatively small compared to other possible actions, then x_{t+1} is considered sub-optimal. In particular, x_{t+1} is identified as sub-optimal if:

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + x_{t+1}) < \delta \max_{a \in \mathbb{A}} F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t} + a), \tag{3}$$

where $0 \leq \delta \leq 1$ is a hyperparameter. A larger δ increases the likelihood that the algorithm identifies x_{t+1} as sub-optimal. From (3), it can be inferred that x_{t+1} is identified as sub-optimal if its associated out-flow is considerably smaller than the out-flow associated with the best possible action in $\mathbb{A}$. This means that the flow function $F_{\boldsymbol{\theta}}(\cdot)$ suggests that replacing x_{t+1} with other actions can lead to remarkable improvement in the sequence property.

4.2 Sequence Editing with GFNSeqEditor

Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, GFNSeqEditor iteratively identifies and edits positions in a seed sequence. Subsection 4.1 presented a simple function for determining if a position x_{t+1} in a sequence should be edited to improve the target property value ((3)). Based on this intuition, we now modify (3) to formally define the sub-optimal-position identification function $D(\cdot)$ used by GFNSeqEditor.

Let $\hat{\boldsymbol{x}}_{:t}$ denote the first t elements of the edited sequence. Assume that $x_t \in \mathbb{A}, \forall t$, meaning that x_t is always in the available actions. At each step t of the algorithm, $D(\cdot)$ accepts $\hat{\boldsymbol{x}}_{:t-1}$ and evaluates whether appending x_t (from the seed sequence) to the edited partial sequence $\hat{\boldsymbol{x}}_{:t-1}$ is detrimental to the performance. In particular, modifying (3), the sub-optimal identifier function $D(\cdot)$ checks the following condition:

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} < \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \nu, \tag{4}$$

where $\nu \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable with variance of σ^2. The variance σ^2 is a hyperparameter. The relation between σ and the algorithm performance will be analyzed in Section 4.3. The inclusion of additive noise ν on the right-hand side of (4) introduces a degree of randomness into the process of identifying sub-optimal positions. This, in turn, fosters exploration in the editing process. The sub-optimal-position-identifier function $D(\cdot)$ determines if x_t is sub-optimal as follows:

$$D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = \begin{cases} 1 & \text{If (4) is met} \\ 0 & \text{Otherwise} \end{cases}. \tag{5}$$

If $D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0$, at step t the algorithm appends x_t from the original sequence $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}_{:t-1}$. Otherwise, if $D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1$, the algorithm samples an action a according to the following policy:

$$\pi(a|\hat{\boldsymbol{x}}_{:t-1}) = (1 - \lambda) \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \lambda \mathbf{1}_{a = x_t}, \tag{6}$$

where $0 \leq \lambda < 1$ is a regularization coefficient and $\mathbf{1}_{a=x_t}$ denotes the indicator function and is 1 if $a = x_t$. The regularization parameter λ allows tuning the sampling process to favor the original sequence (a larger λ leads to a smaller number of edits). The policy in (6) constitutes a trade-off between increasing the target property and decreasing the distance between the edited sequence $\hat{\boldsymbol{x}}$ and the original sequence $\boldsymbol{x}$. Let $\tilde{x}_t$ be the action sampled by the policy π in (6). In summary, the t-th element in the edited sequence can be written as:

$$\hat{x}_t = D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) \tilde{x}_t + (1 - D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma)) x_t. \tag{7}$$

Therefore, at each step t, the edited sequence is updated as $\hat{\boldsymbol{x}}_{:t} = \hat{\boldsymbol{x}}_{:t-1} + \hat{x}_t$. The process continues until step T, where $T = |\boldsymbol{x}|$ denotes the length of the original sequence $\boldsymbol{x}$. Note that $\hat{\boldsymbol{x}}_{:0}$ is an empty sequence. Algorithm 1 summarizes GFNSeqEditor.

Algorithm 1 GFNSeqEditor: Sequence Editor using GFlowNet

```
1: Input: Sequence $\boldsymbol{x}$ with length $T$, flow function $F_{\boldsymbol{\theta}}(\cdot)$ and parameters $\delta$, $\lambda$ and $\sigma$.
2: Initialize $\hat{\boldsymbol{x}}_{:0}$ as an empty sequence.
3: for $t = 1, \ldots, T$ do
4:     Check if $x_t$ is sub-optimal by obtaining $D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma)$ according to (5).
5:     if $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1$ then
6:         Sample $\hat{x}_t$ according to policy $\pi(\cdot|\hat{\boldsymbol{x}}_{:t-1})$ in (6).
7:     else
8:         Assign $\hat{x}_t = x_t$.
9:     end if
10: end for
11: Output: Edited sequence $\hat{\boldsymbol{x}}$.
```

4.3 Analysis

This Subsection analyzes the reward and properties of the edited sequence as well as the number of edits performed by GFNSeqEditor. Specifically, the bounds for the reward of edited sequences, property improvement and the number of edits are determined by the algorithm's hyperparameters σ, δ, and λ. The following theorem specifies the lower bound for the reward of edited sequences.

Theorem 4.1. *Let T be the length of the original sequence $\boldsymbol{x}$. The expected reward of the sequence edited by GFNSeqEditor $\hat{\boldsymbol{x}}$ given $\boldsymbol{x}$ is bounded from below as:*

$$\mathbb{E}[R(\hat{\boldsymbol{x}})|\boldsymbol{x}] \geq \left(1 - \Phi(\frac{1-\delta}{\sigma})\right)(1-\lambda)R_{F,T}, \tag{8}$$

where $\Phi(\cdot)$ denotes the cumulative distribution function (CDF) for the normal distribution and $R_{F,T}$ represents the expected reward of a sequence with length T generated using the flow function $F_{\boldsymbol{\theta}}(\cdot)$.

Proof of Theorem 4.1 is deferred to Appendix A. The following Theorem obtains the expected property improvement upper bound of the proposed GFNSeqEditor. The property improvement of a sequence $\boldsymbol{x}$ is defined as PI $= \hat{y} - y$ where $\hat{y}$ denotes the edited sequence property.

Theorem 4.2. *Let $\mathbb{S}^*_{\boldsymbol{x}}$ be the set of all sequences with length T which have larger properties than that of $\boldsymbol{x}$ (i.e., y). Assume that $\mathbb{S}^*_{\boldsymbol{x}}$ is a non-empty set. The expected property improvement by applying GFNSeqEditor on $\boldsymbol{x}$ is bounded from above as*

$$\mathbb{E}[PI|\boldsymbol{x}] \leq \sum_{\boldsymbol{w} \in \mathbb{S}^*_{\boldsymbol{x}}} \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(p_{\boldsymbol{w}} - y) \tag{9}$$

where $p_{\boldsymbol{w}}$ denote the property of the sequence $\boldsymbol{w}$.

The proof of Theorem 4.2 can be found in Appendix B. Theorems 4.1 and 4.2 demonstrate that an increase in δ results in an increase in both the lower bound of reward and the upper bound of property improvement. While a higher value of σ corresponds to larger lower bounds for the reward, an increase in σ diminishes the upper bound of the property improvement. The following theorem obtains the upper bound on the number of edits performed by the proposed GFNSeqEditor.

Theorem 4.3. *The expected distance between the edited sequence $\hat{\boldsymbol{x}}$ by GFNSeqEditor and the original sequence $\boldsymbol{x}$ is bounded from above as:*

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] \leq \left[(1-\lambda)\left(1 - \Phi(-\frac{\delta}{\sigma})\right)\right]T, \tag{10}$$

where $\text{lev}(\cdot,\cdot)$ is the Levenshtein distance between two sequences.

The proof for Theorem 4.3 is available in Appendix C. The following Theorem specifies the lower bound for the number of edits.

Theorem 4.4. *Let there exists $\epsilon > 0$ such that the flow function $F_{\boldsymbol{\theta}}(\cdot)$ satisfies:*

$$\max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \leq 1 - \epsilon, \forall t, \tag{11}$$

meaning that the probability of choosing of each action is always less than $1-\epsilon$. The expected distance between the edited sequence $\hat{\boldsymbol{x}}$ by GFNSeqEditor and the original sequence $\boldsymbol{x}$ is bounded from below as:

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] \geq \left[\epsilon(1-\lambda)\left(1-\Phi(\frac{1-\delta}{\sigma})\right)\right] T. \tag{12}$$

Proof of Theorem 4.4 can be found in Appendix D. Theorems 4.3 and 4.4 show that as δ increases, both the lower and upper bounds of distance increase. In contrast, an increase in λ leads to a decrease in both the lower and upper bounds of distance. Furthermore, Theorem 4.1 demonstrates that a reduction in λ results in a larger lower bound for the reward. Therefore, Theorems 4.1 and 4.3 reveal a trade-off between the expected number of edits and the lower bound for the expected reward. While it is preferable to select hyperparameters δ and λ that reduce the expected number of edits, an increase in the number of edits corresponds to a larger lower bound for the reward.

5 Experiments

We conducted extensive experiments to assess the performance of GFNSeqEditor in comparison to several state-of-the-art baselines across diverse DNA- and protein-sequence editing tasks. We evaluate on TFbinding, AMP, and CRE datasets. TFbinding and CRE datasets consist DNA sequences with lengths of 8 and 200, respectively. The task in both datasets is to edit sequences to increase their binding activities. The vocabulary for both TFbinding and CRE is the four DNA bases, {`A`, `C`, `G`, `T`}. AMP dataset comprises positive samples, representing anti-microbial peptides (AMPs), and negative samples, which are non-AMPs. The vocabulary consists of 20 amino acids. The primary objective is to edit the non-AMP samples in such a way that the edited versions attain the characteristics exhibited by AMP samples. Additional information about the datasets can be found in Appendix E.1.1.

To evaluate the performance of sequence editing methods, we compute the following metrics:

- **Property Improvement (PI):** The PI for a given sequence $\boldsymbol{x}$ with label y is calculated as the average enhancement in property across edits, expressed as PI $= \frac{1}{n_e}\sum_{i=1}^{n_e}(\hat{y}_i - y)$, where n_e is the number of edited sequences associated with the original sequence $\boldsymbol{x}$ and $\hat{y}_i$ denote the property of the i-th edited sequence $\hat{\boldsymbol{x}}_i$. To evaluate the performance of editing methods, for each dataset we leverage an oracle to obtain $\hat{y}_i$ given $\hat{\boldsymbol{x}}_i$. More details about oracles can be found in Appendix E.
- **Edit Percentage (EP):** The average Levenshtein distance between $\boldsymbol{x}$ and edited sequences normalized by the length of $\boldsymbol{x}$ expressed as $\frac{1}{n_e T}\sum_{i=1}^{n_e}\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i)$.
- **Diversity:** For each sequence $\boldsymbol{x}$, the diversity among edited sequences can be obtained as $\frac{2}{n_e(n_e-1)}\sum_{i=1}^{n_e-1}\sum_{j=i+1}^{n_e}\text{lev}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j)$.
- **GMDPI:** The geometric mean of diversity and PI is measured. This metric highlights algorithms that exhibit strong performance in both aspects simultaneously.

We compared GFNSeqEditor to several baselines, including Directed Evolution (DE) [46], Ledidi [42], LaMBO [48], MOGFN-AL [21], GFlowNet-AL [19], and Seq2Seq. To perform Directed Evolution for sequence editing, we select a set of positions uniformly at random within a given sequence and then apply the directed-evolution algorithm to edit these positions. The implementation of the directed-evolution algorithm is the same as that of the AdaLead framework in [46]. Inspired by graph-to-graph translation for molecular optimization in [23], we implemented another editing baseline, which is called Seq2Seq. For the Seq2Seq baseline, we initially partition the dataset into two subsets: i) sequences with lower target-property values, and ii) sequences with relatively higher target-property values. Subsequently, we create pairs of data samples such that each low-property sequence is paired with its closest counterpart from the high-property sequence set, based on Levenshtein distance. A transformer is then trained to map each low-property sequence to its high-property pair. Essentially, the Seq2Seq baseline maps an input sequence to a similar sequence with a higher property value. Furthermore, we adapted GFlowNet-AL for sequence editing, and named it GFlowNet-E in what follows. In this baseline, the initial segment of the sequence serves as the input, allowing the model to generate the subsequent portion of the sequence. For TF-binding, AMP, and CRE datasets, GFlowNet-E takes in the initial 70%, 65%, and 60% of elements, respectively, from the

Table 1: Performance of GFNSeqEditor compared to the baselines in terms of property improvement (PI), edit percentage (EP), diversity, and geometric mean of property improvement and diversity (GMDPI) on TFbinding, AMP, and CRE datasets. EP is selected to be approximately the same for all algorithms (if possible). Higher PI, diversity and GMDPI are preferable.

	TFbinding				AMP				CRE			
Methods	PI	EP(%)	Diversity	GMDPI	PI	EP(%)	Diversity	GMDPI	PI	EP(%)	Diversity	GMDPI
DE	0.12	25.00	3.01	0.60	0.11	33.82	13.67	1.23	0.63	22.93	62.07	6.25
Ledidi	0.06	27.80	1.25	0.27	0.18	34.79	11.65	1.45	1.36	22.13	50.49	8.29
LaMBO	0.05	25.00	3.14	0.40	0.12	34.33	**15.61**	1.36	0.79	23.35	**62.95**	7.05
MOGFN-AL	0.09	25.00	2.66	0.49	0.10	35.26	7.59	0.87	2.45	22.99	10.96	5.18
GFlowNet-E	0.11	28.35	2.10	0.48	0.28	35.68	3.42	0.98	4.24	22.73	37.06	12.53
Seq2Seq	0.03	41.98	-	-	0.21	78.05	-	-	-	-	-	-
GFNSeqEditor	**0.14**	24.27	**3.84**	**0.73**	**0.33**	34.49	14.34	**2.17**	**9.90**	21.90	40.41	**20.00**

input sequence $\boldsymbol{x}$, and generates the remaining elements using the pre-trained flow function. More details on the baselines can be found in Appendix E.1.

To train both the baselines and the proposed GFNSeqEditor, we divide each dataset into training, validation, and test sets with proportions of 72%, 18% and 10%, respectively. The test set serves the purpose of evaluating the performance of the methods in sequence editing tasks. The flow function $F_{\boldsymbol{\theta}}(\cdot)$ utilized by GFNSeqEditor and the GFlowNet-E baseline is an MLP consisting of two hidden layers, each with a dimension of 2048, and $|\mathbb{A}|$ outputs corresponding to actions. Throughout our experiments, we employ the trajectory balance objective to train the flow function. Additional details regarding the training of the flow function can be found in Appendix E.1.

5.1 Sequence Editing

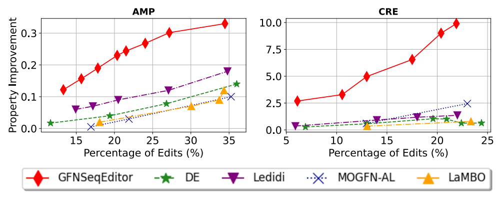

Figure 2: Property improvement of AMP **(left)** and CRE **(right)** with respect to edit percentage.

Table 1 presents the performance of GFNSeqEditor and other baselines on TFbinding, AMP, and CRE datasets[3]. We set GFNSeqEditor and all the baselines except for Seq2Seq to create 10 edited sequences for each input sequence. The Seq2Seq implementation closely resembles a deterministic machine translator and is limited to producing just one edited sequence per input, resulting in a diversity score of zero. Additionally, Figure 2 shows the property improvement achieved by GFNSeqEditor, DE, Ledidi, LaMBO, and MOGFN-AL across a range of edit percentages. As evident from Table 1 and Figure 2, GFNSeqEditor outperforms all baselines, achieving substantial property improvements with a controlled number of edits. This superior performance is attributed to GFNSeqEditor's utilization of a pre-trained flow function from GFlowNet, enabling it to achieve significantly higher property improvements than DE, Ledidi, LaMBO, and MOGFN-AL, which rely on local search techniques by optimizing either a given single sequence or a batch of sequences. Specifically, the flow function $F_{\boldsymbol{\theta}}(\cdot)$ is trained to sample sequences with probability proportional to their reward and, as a result, employing the policy in (6) for editing enables GFNSeqEditor to leverage global information contained in $F_{\boldsymbol{\theta}}(\cdot)$ about the entire space of sequences. Furthermore, GFNSeqEditor achieves larger property improvement than GFlowNet-E. The GFNSeqEditor identifies and edits sub-optimal positions within a seed sequence using (4), while GFlowNet-E only edits the tail of the input seed sequence. This indicates the effectiveness of the sub-optimal position identifier function of GFNSeqEditor.

Ablation study. We further study the property improvement achieved by GFNSeqEditor along with edit percentage across various choices of hyperparameters δ and λ. Figure 3 illustrates that an increase in δ generally corresponds to an increase in both property improvement and edit percentage, whereas, in most cases, an increase in λ results in a decrease in property improvement and edit percentage. Furthermore, in Figure 7 in Appendix E.3, we illustrate the impact of changing σ on property improvement and edit diversity for GFNSeqEditor. This figure highlights that increasing σ results in decreased property improvement and enhanced diversity. These results corroborate the theoretical analyses outlined in Theorems 4.1, 4.2 and 4.3 in Section 4.3.

[3]Seq2Seq relies on identifying pairs of similar sequences for training. However, we were unable to identify similar pairs for CRE, possibly because of the limited number of training samples relative to the lengthy nature of the sequences (i.e., sequences with a length of 200).

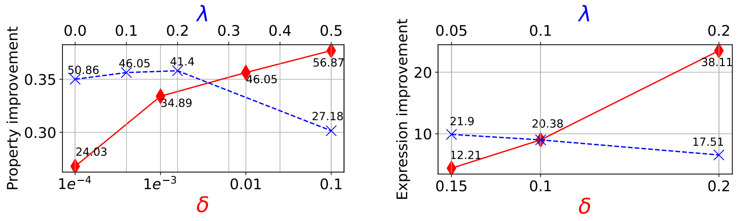

Figure 3: Studying the effect of heyperparameters δ and λ on the performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets. The marker values are edit percentages.

5.2 Assisting Sequence Generation

In addition to editing sequences, we investigate the ability of GFNSeqEditor to be used alongside a sequence generative model to enhance the generation of novel sequences. This highlights the versatility of the proposed GFNSeqEditor. In this Subsection, we utilize a pre-trained diffusion model (DM) for sequence generation, with further details available in Appendix E.2. The sequences generated by the DM are passed to GFNSeqEditor to improve their target property. Given that GFNSeqEditor utilizes a trained GFlowNet model, this combination of a DM and GFNSeqEditor can be regarded as an *ensemble approach*, effectively leveraging both the DM and the GFlowNet for sequence generation. Table 2 presents the property and diversity metrics for sequences generated by the DM, the GFlowNet, and the combined DM+GFNSeqEditor across AMP and CRE datasets, with each method generating $1,000$ sequences. As observed from Table 2, GFlowNet excels at producing sequences with higher property values compared to the DM, while the DM exhibits greater sequence diversity than the GFlowNet. Sequences generated by DM+GFNSeqEditor maintain similar property levels to the GFlowNet on its own, while their diversity is in line with that of the DM. This highlights the effectiveness of DM+GFNSeqEditor in harnessing the benefits of both the GFlowNet and the DM.

Table 2: Performance of DM, GFlowNet and combination of DM with GFNSeqEditor for generating novel sequences.

	AMP		CRE	
Algorithms	Property	Diversity	Property	Diversity
DM	0.66	23.86	1.75	107.38
GFlowNet	0.74	17.86	28.20	83.88
DM+GFNSeqEditor	0.73	23.78	26.42	103.10

Moreover, we show the CDF of the property for sequences generated by the DM, the GFlowNet, and DM+GFNSeqEditor in Figure 4. As shown, the CDF of DM+GFNSeqEditor aligns with both DM and GFlowNet. Specifically, for AMP dataset, DM+GFNSeqEditor generates more sequences with higher properties than 0.78 compared to GFlowNet, while reducing the number of low-property generated sequences compared to DM alone. In the case of CRE dataset, the results in Figure 4 indicate that as δ increases, the CDF of DM+GFNSeqEditor becomes more akin to that of GFlowNet. This is expected, as an increase in δ leads to a greater number of edits.

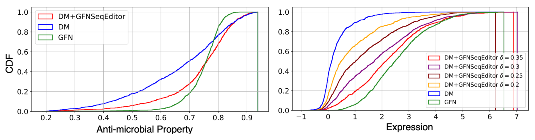

Figure 4: CDF of generated sequence properties for AMP **(left)** and CRE **(right)**. A right-shifted curve indicates that the model is generating more sequences that are high in the target property.

5.3 Sequence Combination

GFNSeqEditor possesses the capability to combine multiple sequences, yielding a novel sequence that closely resembles its *parent* sequences. This capability proves invaluable in several applications. For example, when it is important to shorten relatively lengthy sequences while retaining desired properties (see, e.g., [54, 59]). GFNSeqEditor accomplishes this by combining a longer sequence with a shorter one. The resultant sequence maintains high similarity with the longer one to retain its desired properties, while also resembling a realistic, relatively shorter sequence to ensure safety and predictability. Algorithm 2 in Appendix E.5 describes using GFNSeqEditor to combine two sequences with the goal of shortening the longer one.

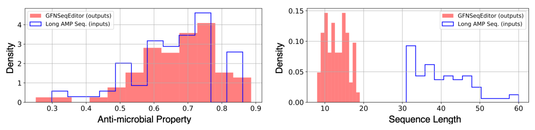

Figure 5: GFNSeqEditor effectively reduces the length of AMP sequence inputs (**right**) while keeping their properties intact (**left**).

We evaluate GFNSeqEditor's performance in combining pairs of long and short sequences using the AMP dataset as a test case. In this context, a *long sequence* is defined as one with a length exceeding

30, while a *short sequence* has a length shorter than 20. Each initial pair consists of a long AMP sequence and its closest short sequence with an AMP property exceeding 0.7. Table 5 and Figure 5 in Appendix E.5 present the results of sequence combination for sequence length reduction. As indicated in Table 5, GFNSeqEditor not only enhances the properties of the initial long sequences, but also significantly shortens them by more than 63%. Additionally, the sequences generated by GFNSeqEditor resemble both the initial long and short sequences, with an average Levenshtein similarity of approximately 65% to long sequences and 55% to short sequences.

6 Conclusions

This paper introduces GFNSeqEditor, a generative model for sequence editing built upon GFlowNet. Given an input seed sequence, GFNSeqEditor identifies and edits positions within the input sequence to enhance its property. This paper also offers a theoretical analysis of the properties of edited sequences and the amount of edits performed by GFNSeqEditor. Experimental evaluations using real-world DNA and protein datasets demonstrate that GFNSeqEditor outperforms state-of-the-art baselines in terms of property enhancement while maintaining a similar amount of edits. Nevertheless, akin to many machine learning algorithms, GFNSeqEditor does have its limitations. It relies on a well-trained GFlowNet model, necessitating the availability of a high-quality trained GFlowNet for optimal performance.

References

[1] Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*, 2019.

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

[3] Avdeyev, P., Shi, C., Tan, Y., Dudnyk, K., and Zhou, J. Dirichlet diffusion score model for biological sequence generation. *arXiv preprint arXiv:2305.10699*, 2023.

[4] Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016.

[5] Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27381–27394, 2021.

[6] Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[7] Chen, C. S., Zhang, Y., Liu, X., and Coates, M. Bidirectional learning for offline model-based biological sequence design. In *Proceedings of the International Conference on Machine Learning*, 2023.

[8] Chen, Y. and Mauch, L. Order-preserving GFlownets. In *International Conference on Learning Representations*, 2024.

[9] Cox, D. B. T., Platt, R. J., and Zhang, F. Therapeutic genome editing: prospects and challenges. *Nature medicine*, 21(2):121–131, 2015.

[10] Dadkhahi, H., Rios, J., Shanmugam, K., and Das, P. Fourier representations for black-box optimization over categorical variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10156–10165, 2022.

[11] Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pp. 518–528. PMLR, 2022.

[12] Gosai, S. J., Castro, R. I., Fuentes, N., Butts, J. C., Kales, S., Noche, R. R., Mouri, K., Sabeti, P. C., Reilly, S. K., and Tewhey, R. Machine-guided design of synthetic cell type-specific cis-regulatory elements. *bioRxiv*, pp. 2023–08, 2023.

[13] Grau-Moya, J., Leibfried, F., and Vrancx, P. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations*, 2019.

[14] Guo, S., Chu, J., Zhu, L., and Li, T. Dynamic backtracking in gflownet: Enhancing decision steps with reward-dependent adjustment mechanisms. *arXiv preprint arXiv:2404.05576*, 2024.

[15] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the International Conference on Machine Learning*, pp. 1352–1361, 2017.

[16] Hansen, N. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

[17] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. URL `https://github.com/hojonathanho/diffusion`.

[18] Hoffman, S. C., Chenthamarakshan, V., Wadhawan, K., Chen, P.-Y., and Das, P. Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, 4(1): 21–31, 2022.

[19] Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F. P., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. Biological sequence design with GFlowNets. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 9786–9801, Jul 2022.

[20] Jain, M., Deleu, T., Hartford, J., Liu, C.-H., Hernandez-Garcia, A., and Bengio, Y. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023.

[21] Jain, M., Raparthy, S. C., Hernandez-Garcia, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. Multi-objective gflownets. In *Proceedings of the International Conference on Machine Learning*, 2023.

[22] Jang, H., Kim, M., and Ahn, S. Learning energy decompositions for partial inference in GFlownets. In *International Conference on Learning Representations*, 2024.

[23] Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*, 2019.

[24] Kim, H., Kim, M., Choi, S., and Park, J. Genetic-guided gflownets: Advancing in practical molecular optimization benchmark. *arXiv preprint arXiv:2402.05961*, 2024.

[25] Kim, M., Ko, J., Yun, T., Zhang, D., Pan, L., Kim, W. C., Park, J., Bengio, E., and Bengio, Y. Learning to scale logits for temperature-conditional GFlowNets. In *Proceedings of the International Conference on Machine Learning*, volume 235, pp. 24248–24270, Jul 2024.

[26] Kim, M., Yun, T., Bengio, E., Zhang, D., Bengio, Y., Ahn, S., and Park, J. Local search GFlownets. In *International Conference on Learning Representations*, 2024.

[27] Koziarski, M., Abukalam, M., Shah, V., Vaillancourt, L., Schuetz, D. A., Jain, M., van der Sloot, A. M., Bourgey, M., Marinier, A., and Bengio, Y. Towards DNA-encoded library generation with GFlownets. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024.

[28] Li, H., Yang, Y., Hong, W., Huang, M., Wu, M., and Zhao, X. Applications of genome editing technology in the targeted therapy of human diseases: mechanisms, advances and prospects. *Signal transduction and targeted therapy*, 5(1):1, 2020.

[29] Li, W., Li, Y., Li, Z., HAO, J., and Pang, Y. DAG matters! GFlownets enhanced explainer for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=jgmuRzM-sb6`.

[30] Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.

[31] Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in GFlownets. In *Advances in Neural Information Processing Systems*, 2022.

[32] Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E. J., Everett, K. E., Zhang, D., and Bengio, Y. GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Mohammadpour, S., Bengio, E., Frejinger, E., and Bacon, P.-L. Maximum entropy GFlowNets with soft Q-learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 238, pp. 2593–2601, May 2024.

[34] Mullis, M. M., Rambo, I. M., Baker, B. J., and Reese, B. K. Diversity, ecology, and prevalence of antimicrobials in nature. *Frontiers in microbiology*, pp. 2518, 2019.

[35] Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 2772–2782, 2017.

[36] Nishikawa-Toomey, M., Deleu, T., Subramanian, J., Bengio, Y., and Charlin, L. Bayesian learning of causal structure and mechanisms with gflownets and variational bayes. *arXiv preprint arXiv:2211.02763*, 2022.

[37] Niu, P., Wu, S., Fan, M., and Qian, X. GFlowNet training by policy gradients. In *Proceedings of the International Conference on Machine Learning*, volume 235, pp. 38344–38380, Jul 2024.

[38] Pan, L., Zhang, D., Courville, A., Huang, L., and Bengio, Y. Generative augmented flow networks. *arXiv preprint arXiv:2210.03308*, 2022.

[39] Pan, L., Zhang, D., Jain, M., Huang, L., and Bengio, Y. Stochastic generative flow networks. *arXiv preprint arXiv:2302.09465*, 2023.

[40] Pirtskhalava, M., Amstrong, A. A., Grigolava, M., Chubinidze, M., Alimbarashvili, E., Vishnepolsky, B., Gabrielian, A., Rosenthal, A., Hurt, D. E., and Tartakovsky, M. DBAASP v3: database of antimicrobial/cytotoxic activity and structure of peptides as a resource for development of new therapeutics. *Nucleic Acids Research*, 49(D1):D288–D297, 11 2020.

[41] Ren, Z., Li, J., Ding, F., Zhou, Y., Ma, J., and Peng, J. Proximal exploration for model-guided protein sequence design. In *Proceedings of the International Conference on Machine Learning*, volume 162, pp. 18520–18536, Jul 2022.

[42] Schreiber, J., Lu, Y. Y., and Noble, W. S. Ledidi: Designing genomic edits that induce functional activity. *bioRxiv*, 2020. doi: 10.1101/2020.05.21.109686. URL `https://www.biorxiv.org/content/early/2020/05/25/2020.05.21.109686`.

[43] Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. Towards understanding and improving gflownet training. *arXiv preprint arXiv:2305.07170*, 2023.

[44] Sidorczuk, K., Gagat, P., Pietluch, F., Kała, J., Rafacz, D., Bąkała, L., Słowik, J., Kolenda, R., Rödiger, S., Fingerhut, L. C. H. W., Cooke, I. R., Mackiewicz, P., and Burdukiewicz, M. Benchmarks in antimicrobial peptide prediction are biased due to the selection of negative data. *Briefings in Bioinformatics*, 23(5):bbac343, Sep 2022.

[45] Silva, T., Carvalho, L. M., Souza, A. H., Kaski, S., and Mesquita, D. Embarrassingly parallel GFlowNets. In *Proceedings of the International Conference on Machine Learning*, volume 235, pp. 45406–45431, Jul 2024.

[46] Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., and Kelsic, E. D. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

[47] Song, Y., Sohl-Dickstein, J., Brain, G., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *arXiv preprint arXiv:2011.13456*, 2021.

[48] Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, volume 162, pp. 20459–20478, Jul 2022.

[49] Swersky, K., Rubanova, Y., Dohan, D., and Murphy, K. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, pp. 769–778. PMLR, 2020.

[50] Taskiran, I. I., Spanier, K. I., Christiaens, V., Mauduit, D., and Aerts, S. Cell type directed design of synthetic enhancers. *bioRxiv*, pp. 2022–07, 2022.

[51] Terayama, K., Sumita, M., Tamura, R., and Tsuda, K. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

[52] Trabucco, B., Kumar, A., Geng, X., and Levine, S. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

[53] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[54] Xu, X., Chemparathy, A., Zeng, L., Kempton, H. R., Shang, S., Nakamura, M., and Qi, L. S. Engineered miniature crispr-cas system for mammalian genome regulation and editing. *Molecular Cell*, 81(20):4333–4345.e4, 2021.

[55] Zhang, D., Fu, J., Bengio, Y., and Courville, A. Unifying likelihood-free inference with black-box optimization and beyond. *arXiv preprint arXiv:2110.03372*, 2021.

[56] Zhang, D., Chen, R. T., Malkin, N., and Bengio, Y. Unifying generative models with gflownets. *arXiv preprint arXiv:2209.02606*, 2022.

[57] Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pp. 26412–26428. PMLR, 2022.

[58] Zhang, D., Pan, L., Chen, R. T., Courville, A., and Bengio, Y. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*, 2023.

[59] Zhao, F., Zhang, T., Sun, X., Zhang, X., Chen, L., Wang, H., Li, J., Fan, P., Lai, L., Sui, T., et al. A strategy for cas13 miniaturization based on the structure and alphafold. *Nature Communications*, 14(1):5545, 2023.

[60] Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. A variational perspective on generative flow networks. *arXiv preprint arXiv:2210.07992*, 2022.

[61] Zrimec, J., Fu, X., Muhammad, A. S., Skrekas, C., Jauniskis, V., Speicher, N. K., Börlin, C. S., Verendel, V., Chehreghani, M. H., Dubhashi, D., et al. Controlling gene expression with deep generative design of regulatory dna. *Nature communications*, 13(1):5099, 2022.

A Proof of Theorem 4.1

Let $\boldsymbol{z}$ denotes a sequence with length T generated from scratch using the policy $\pi_F(\cdot)$ as

$$\pi_F(a|\boldsymbol{z}_{:t}) = \frac{F_{\boldsymbol{\theta}}(\boldsymbol{z}_{:t}+a)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\boldsymbol{z}_{:t}+a')}. \tag{13}$$

The expected reward of $\boldsymbol{z}$ can be obtained as

$$R_{F,T} = \mathbb{E}[\boldsymbol{z}] = \sum_{\boldsymbol{w}\in\mathbb{T}_T} \Pr[\boldsymbol{z}=\boldsymbol{w}]R(\boldsymbol{w}) = \sum_{\boldsymbol{w}\in\mathbb{T}_T}\prod_{t=1}^{T}\pi_F(w_t|\boldsymbol{w}_{:t-1})R(\boldsymbol{w}) \tag{14}$$

where $\mathbb{T}_T$ denotes the set of sequences with length T that can be generated by $F_{\boldsymbol{\theta}}(\cdot)$. The probability that the GFNSeqEditor outputs an arbitrary sequence $\boldsymbol{w}\in\mathbb{T}_T$ given $\boldsymbol{x}$ can be expressed as

$$\Pr[\hat{\boldsymbol{x}}=\boldsymbol{w}|\boldsymbol{x}] = \prod_{t=1}^{T}\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1},\boldsymbol{x}]. \tag{15}$$

The probability $\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1},\boldsymbol{x}]$ can be obtained as

$$\begin{aligned}\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1},\boldsymbol{x}] &= \Pr[D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=1]\pi(w_t|\boldsymbol{w}_{:t-1}) + \Pr[D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=0]\mathbf{1}_{w_t=x_t}\\ &\geq \Pr[D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=1]\pi(w_t|\boldsymbol{w}_{:t-1})\end{aligned} \tag{16}$$

where $\pi(\cdot)$ defined in (6). According to (6), it can be written that

$$\pi(w_t|\boldsymbol{w}_{:t-1}) \geq (1-\lambda)\frac{F_{\boldsymbol{\theta}}(\boldsymbol{w}_{:t})}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\boldsymbol{w}_{:t-1}+a')} = (1-\lambda)\pi_F(w_t|\boldsymbol{w}_{:t-1}). \tag{17}$$

Furthermore, according to (4) and (5), we have $D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=1$ if

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+x_t)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} - \delta\max_{a\in\mathbb{A}}\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} < \nu. \tag{18}$$

In addition, it can be inferred that

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+x_t)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} - \delta\max_{a\in\mathbb{A}}\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} \leq 1-\delta. \tag{19}$$

Therefore, it can be concluded that if $\nu > 1-\delta$, it is guaranteed that $D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=1$. Since ν follows a Gaussian distribution with a variance of σ^2 we have $\nu > 1-\delta$ with probability $1-\Phi(\frac{1-\delta}{\sigma})$. Hence, it can be written that

$$\Pr[D(\hat{\boldsymbol{x}}_{:t-1};\delta,\sigma)=1] \geq 1-\Phi(\frac{1-\delta}{\sigma}). \tag{20}$$

Combining (20) and (17) with (16), we get

$$\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1},\boldsymbol{x}] \geq (1-\lambda)\pi_F(w_t|\boldsymbol{w}_{:t-1})\left(1-\Phi(\frac{1-\delta}{\sigma})\right). \tag{21}$$

Moreover, combining (21) with (15), we obtain

$$\Pr[\hat{\boldsymbol{x}}=\boldsymbol{w}|\boldsymbol{x}] \geq \prod_{t=1}^{T}(1-\lambda)\pi_F(w_t|\boldsymbol{w}_{:t-1})\left(1-\Phi(\frac{1-\delta}{\sigma})\right). \tag{22}$$

Using (22) for the expected reward of $\hat{\boldsymbol{x}}$ given $\boldsymbol{x}$ we can write

$$\begin{aligned}\mathbb{E}[R(\hat{\boldsymbol{x}})|\boldsymbol{x}] &= \sum_{\boldsymbol{w}\in\mathbb{T}_T}\Pr[\hat{\boldsymbol{x}}=\boldsymbol{w}|\boldsymbol{x}]R(\boldsymbol{w})\\ &\geq \sum_{\boldsymbol{w}\in\mathbb{T}_T}\prod_{t=1}^{T}(1-\lambda)\pi_F(w_t|\boldsymbol{w}_{:t-1})\left(1-\Phi(\frac{1-\delta}{\sigma})\right)R(\boldsymbol{w}).\end{aligned} \tag{23}$$

Combining (23) with (14), we get

$$\mathbb{E}[R(\hat{\boldsymbol{x}})|\boldsymbol{x}] \geq (1-\lambda)\left(1-\Phi(\frac{1-\delta}{\sigma})\right)R_{F,T} \tag{24}$$

which proves (8). Moreover, the upper bound of property improvement by the proposed GFNSeqEditor is analyzed in Appendix B.

B Proof of Theorem 4.2

The expected property improvement of GFNSeqEditor can be obtained as

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] = \sum_{\boldsymbol{w}\in\mathbb{T}_T} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{25}$$

Since $\mathbb{T}_T$ can be split into two sets $\mathbb{S}^*_{\boldsymbol{x}}$ and $\mathbb{T}_T \setminus \mathbb{S}^*_{\boldsymbol{x}}$, the expected property improvement of GFNSeqEditor can be obtained as

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] = \sum_{\boldsymbol{w}\in\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y) + \sum_{\boldsymbol{w}\in\mathbb{T}_T\setminus\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{26}$$

If $\boldsymbol{w} \in \mathbb{T}_T \setminus \mathbb{S}^*_{\boldsymbol{x}}$, then $p_{\boldsymbol{w}} \leq y$. Therefore, the expected property improvement of GFNSeqEditor can be bounded from above as

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] \leq \sum_{\boldsymbol{w}\in\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{27}$$

The probability that the GFNSeqEditor outputs $\boldsymbol{w} \in \mathbb{S}^*_{\boldsymbol{x}}$ can be expressed as

$$\Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}] = \prod_{t=1}^{T} \Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}]. \tag{28}$$

The probability $\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}]$ can be obtained as

$$\begin{aligned}\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}] =& \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1]\pi(w_t|\boldsymbol{w}_{:t-1}) \\ &+ \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0]\mathbf{1}_{x_t=w_t}.\end{aligned} \tag{29}$$

If $x_t \neq w_t$, according to (37) and considering the fact that $\pi(w_t|\boldsymbol{w}_{:t-1}) \leq 1$, the probability in (29) can be bounded from above as

$$\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{30}$$

Otherwise if $x_t = w_t$, it can be written that $\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}] \leq 1$. Since any $\boldsymbol{w} \in \mathbb{S}^*_{\boldsymbol{x}}$ should be different from $\boldsymbol{x}$ in at least one position, combining (28) with (30) we can conclude that

$$\Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{31}$$

Combining (27) with (31) proves the Theorem.

C Proof of Theorem 4.3

We obtain the upper bound for the expected distance between edited sequence $\hat{\boldsymbol{x}}$ and the original sequence $\boldsymbol{x}$. Since both $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ have the same length T, the distance $\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})$ can be interpreted as the number of elements different in these two sequences. Therefore, in order to obtain $\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})$, it is sufficient to find the number of times that $x_t \neq \hat{x}_t, \forall t : 1 \leq t \leq T$. If $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0$, then $\hat{x}_t = x_t$. Furthermore, if $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1$, then according to (6), we have $\hat{x}_t = x_t$ with probability

$$(1 - \lambda)\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \lambda. \tag{32}$$

Therefore, the probability $\Pr[\hat{x}_t \neq x_t]$ can be obtained as

$$\Pr[\hat{x}_t \neq x_t] = \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1 - \lambda)\left(1 - \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')}\right). \tag{33}$$

Since $F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t) \geq 0$, the probability $\Pr[\hat{x}_t \neq x_t]$ can be bounded as

$$\Pr[\hat{x}_t \neq x_t] \leq \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1 - \lambda). \tag{34}$$

Moreover, if we have

$$\nu \leq \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} - \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \tag{35}$$

then $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0$. Furthermore, the right hand side of (35) can be bounded from below as

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} - \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \geq -\delta. \tag{36}$$

Therefore, if $\nu \leq -\delta$, it is ensured that $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0$. The probability that $\nu \leq -\delta$ is $\Phi(-\frac{\delta}{\sigma})$. Hence, we can conclude that

$$\Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1] = 1 - \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{37}$$

Combining (37) with (34), we arrive at

$$\Pr[\hat{x}_t \neq x_t] \leq \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(1 - \lambda). \tag{38}$$

Moreover, since both $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ have the same length T, the expected Levenshtein distance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ can be obtained as

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] = \sum_{t=1}^{T} \Pr[\hat{x}_t \neq x_t]. \tag{39}$$

Thus, combining (39) with (38), we can write that

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] \leq \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(1 - \lambda)T \tag{40}$$

which proves the Theorem.

D Proof of Theorem 4.4

According to (33) and the assumption in (11), it can be written that

$$\Pr[\hat{x}_t \neq x_t] \geq \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1 - \lambda)\epsilon. \tag{41}$$

Combining (41) with (20), we get

$$\Pr[\hat{x}_t \neq x_t] \geq \epsilon(1 - \lambda)\left(1 - \Phi(\frac{1 - \delta}{\sigma})\right). \tag{42}$$

Summing (42) over all elements in the sequence proves the theorem.

E Supplementary Experimental Results and Details

This appendix provides a comprehensive overview of the experimental setup in Section 5 and presents additional supplementary experimental results.

E.1 Implementation Details

All training and inferences, including GFNSeqEditor, have been conducted using a single Nvidia Quadro P6000.

E.1.1 Datasets

Detailed information about the datasets can be found below:

- **TFbinding:** The dataset is taken from [4] and contains all possible DNA sequences with length 8. The vocabulary is the four DNA bases, $\{$`A`, `C`, `G`, `T`$\}$. The goal is to edit a given DNA sequence to increase its binding activity with certain DNA-binding proteins called transcription factors. Higher binding activity is preferable. For train, test and validation purposes 50% of the dataset is set aside. The task entails editing a test dataset consisting of 10% of samples while the remaining data is utilized for training and validation.
- **AMP:** The dataset, acquired from DBAASP [40], is curated following the approach outlined by [19]. Peptides (i.e. short proteins) within a sequence-length range of 12 to 60 amino acids are specifically chosen. The dataset comprises a total of $6,438$ positive samples, representing anti-microbial peptides (AMPs), and 9,522 negative samples, which are non-AMPs. The vocabulary consists of 20 amino acids. The primary objective is to edit the non-AMP samples in such a way that the edited versions attain the characteristics exhibited by AMP samples. The task primarily centers on editing a subset comprising 10% of the non-AMP samples, designated for use as test samples, with the remaining samples allocated for training and validation purposes.
- **CRE:** The dataset contains putative human cis-regulatory elements (CRE) which are regulatory DNA sequences modulating gene expression. CREs were profiled via massively parallel reporter assays (MPRAs)[12] where the activity is measured as the expression of the reporter gene. For our analysis, we randomly extract $10,000$ DNA sequences, each with a length of 200 base pairs, utilizing a vocabulary of the four bases. The overarching objective is to edit the DNA sequences to increase the reporter gene's expression specifically within the K562 cell line, which represents erythroid precursors in leukemia. The task involves editing a subset of $1,000$ test samples, while the rest are allocated for training and validation purposes.

E.1.2 Oracles

To evaluate the performance of each sequence editing method in terms of property improvement, it is required to obtain the properties of edited sequences. To this end, we employ an oracle for each dataset. The TFbinding dataset contains all possible $65,792$ DNA sequences with length of 8. Therefore, by looking into the dataset the true label of each edited sequence can be found. Following [1, 19], the AMP dataset is split into two parts: D_1 and D_2. The oracle for the AMP dataset is a set of trained models on partition D_2 as a simulation of wet-lab experiments. We employed oracles trained by [19] for AMP dataset. It is worth noting that the performance of predictive models on AMP datasets can be influenced by negative sampling in the dataset [44]. Furthermore, for CRE dataset we leverage the Malinois model [12] which is a deep convolutional neural network (CNN) for cell type-informed CRE activity prediction of any arbitrary sequence.

E.1.3 Baselines Implementation

DE and Ledidi. In order to implement DE and Ledidi baselines, there should be a proxy model to enable baselines to evaluate their candidate edits at each iteration of these algorithms. For each dataset, we train a proxy model on the training split of each dataset. For the TFBinding dataset, we configure a three-layer MLP with hidden dimensions of 64. In the case of AMP, we opt for a four-layer MLP, also with hidden dimensions of 64. Finally, for CRE, we utilize a four-layer MLP with hidden dimensions set to 2048. Across all models, the learning rate is consistently set to 10^{-4}, ReLU serves as the activation function, and we set the number of epochs as $2,000$. To implement the DE baseline, we randomly select edit locations based on the desired edit percentages. At each selected location, we apply an edit by choosing the action that maximizes the proxy model's property prediction.

LaMBO. We utilize the official implementation of LaMBO[4]. Test sequences[5] serve as the candidate pool, and all candidate samples in the pool are weighted similarly to maintain consistency with other sequence editing baselines. To ensure a fair comparison, we employ the same proxy as GFNSeqEditor to calculate the property scores. Across all three datasets, we use `mlm` as the encoder objective, `ei` as the acquisition function, and DKL SVGP regression as the surrogate. To generate 10 edits per sample, we configure `num-gens=10`, and `window-size` is adjusted for each dataset to ensure the edited

[4]https://github.com/samuelstanton/lambo.

[5]For the AMP dataset, we have removed the samples with a length lower than `window-size`.

percentages closely match the desired values. Hence, we set it to 2, 14, and 23 for TFBinding, AMP, and CRE, respectively. Additionally, for all datasets, we set `pref-cond=False`, `pref-alpha=1`, and `beta-sched=1`.

MOGFN-AL. We utilize the official implementation of MOGFN-AL[6], where sequence tasks are based on the LaMBO implementation. Similar to LaMBO, test sequences serve as the candidate pool, weighted equally. The same proxy as GFNSeqEditor and LaMBO is employed to calculate property scores. Like LaMBO, we utilize `mlm` as the encoder objective. Training and validation batch sizes are set to 16 and 64, respectively, with hyperparameters for the conditional transformer as follows: `num-hid=128`, `num-layers=3`, and `num-head=8`. For each dataset, we adjust the `max-len` parameter to closely match the desired edited percentage. Thus, for TFBinding, AMP, and CRE, we set them as 6, 30, and 50, respectively. The reward function is modified to match that of GFNSeqEditor. To ensure fair comparisons and adapt LaMBO and MOGFN-AL for sequence editing, we do not use active learning settings for either baseline.

Seq2Seq. In order to implement Seq2Seq baseline we use a standard transformer [53] as the translator to map an input sequence to an output sequence with superior property. We paired samples in each dataset such that each pair contains a sequence with lower property and a similar sequence with higher property which is the most similar to the sequence with lower property in the dataset. The transformer is trained to map the low property sequence to the high property sequence in each pair. The transformer is trained using the standard configurations in Pytorch transformer module tutorial. Both the embedding dimension of the transformer and the dimension of the 2 layer feedforward network model in the transformer encoder are set to 200. The number of heads in multihead attention layer is 2 and the drop-out rate is 0.2. We employ the CrossEntropyLoss function in conjunction with the stochastic gradient descent optimizer. The initial learning rate is set at 5.0 and follows a StepLR schedule.

E.1.4 GFlowNet Training

Both the baseline GFlowNet-E and the proposed GFNSeqEditor use the same trained GFlowNet model. We trained an active learning based GFlowNet model following the setting in [19]. In active learning setting, at each round of active learning $t \times K$ candidates generated by GFlowNet are sampled and then top K samples based on scores given by a proxy are chosen to be added to the offline dataset. Here offline dataset refers to an initial labeled dataset. To train the GFlowNet, we employed the same proxy models as those used by other baseline methods. For all datasets, we set the number of active learning rounds to 1, with t equal to 5 and K equal to 100. We parameterize the flow using a MLP comprising two hidden layers, each with a dimension of 2048, and $|\mathbb{A}|$ outputs corresponding to individual actions. Throughout our experiments, we employ the trajectory balance objective for training. Adam optimizer with $(\beta_0, \beta_1) = (0.9, 0.999)$ is utilized during the training process. The learning rate for $logZ$ in trajectory balance loss is set to 10^{-3} for all the experiments. The number of training steps for TFbinding, AMP and CRE are 5000, 10^6 and 10^4, respectively. The remaining hyperparameters were configured in accordance with the settings established in [19].

E.2 Diffusion Model Training

We trained our diffusion models on the full sequence datasets of AMP sequences or CRE sequences. The sequences were one-hot encoded, yielding 20-vectors for protein sequences and 4-vectors for DNA sequences.

We employed the "variance-preserving stochastic differential equation" (VP-SDE) [47]. We used a variance schedule of $\beta(t) = 0.9t + 0.1$. We set our time horizon $T = 1$ (i.e. $t \in [0, 1)$). This amounts to adding Gaussian noise over continuous time.

For our discrete-time diffusion model, we defined a discrete-time Gaussian noising process, following [17]. We defined $\beta_t = (1 \times 10^{-4}) + (1 \times 10^{-5})t$. We set our time horizon $T = 1000$ (i.e. $t \in [0, 1000]$).

Our denoising network was based on a transformer architecture. The time embedding was computed as $[\sin(2\pi \frac{t}{T} z), \cos(2\pi \frac{t}{T} z)]$, where z is a 30-vector of Gaussian-distributed parameters that are not trainable. The time embeddings were passed through two dense layers with a sigmoid in between,

[6]https://github.com/MJ10/mogfn-al.

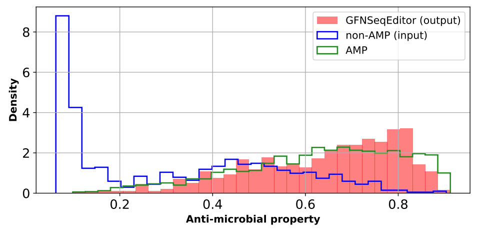

Figure 6: GFNSeqEditor shifts the distribution of non-AMP inputs to the known AMPs.

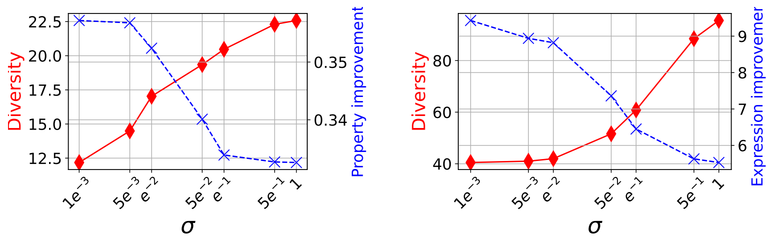

Figure 7: Studying the effect of hyperparameter σ on the diversity and performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets.

mapping to a 256-vector of time representations. For any input in a batch, it was concatenated with the time embedding and a sinusoidal positional embedding (defined in [53]) of dimension 64. This concatenation was passed to a linear layer to map it to 128 dimensions. This was then passed to a standard transformer encoder of 3 layers, 8 attention heads, and with a hidden dimension of 128 and an MLP dimension of 64. The result was then passed to a linear layer which projected back to the input dimension.

We trained our diffusion models with a learning rate of 0.001, for 100 epochs. We noted that the loss had converged for all models at that point. We also employed empirical loss weighting, where the loss of each input in a batch is divided by the L2 norm of the true Stein score.

We trained our diffusion models on a single Nvidia Quadro P6000.

When generating samples from a continuous-time diffusion model, we used the predictor-corrector algorithm defined in [47], using 1000 time steps from T to 0. We then rounded all outputs to the nearest integer to recover the one-hot encoded sample.

E.3 Supplementary Results for Sequence Editing

In Figure 6, we illustrate the distribution of input non-AMP sequences, the sequences edited by GFNSeqEditor, and the AMP samples from the AMP dataset. It is evident from Figure 6 that GFNSeqEditor shifts the property distribution of input non-AMP sequences towards that of AMP sequences. Moreover, Figure 7 illustrates the impact of changing σ on property improvement and edit diversity for GFNSeqEditor. As can be seen increasing σ results in decreased property improvement and enhanced diversity.

It is worth noting that GFNSeqEditor is capable of performing edits even when certain portions of the input sequence are masked and cannot be modified. Table 3 showcases the performance of GFNSeqEditor compared to Ledidi on the CRE dataset, with the first 100 elements of the input sequences masked. As depicted in Table 3, GFNSeqEditor achieves significantly greater property improvement than Ledidi while utilizing a lower edit percentage.

E.4 Supplementary Results for Sequence Generation

This Subsection compares the performance of GFNSeqEditor in sequence generation task with that of GFlowNet and diffusion model (DM) on CRE dataset. We relax the hyperparameters to allow a higher amount of edits and we set $\delta = 0.4$, $\lambda = 0.1$ and $\sigma = 0.001$ for GFNSeqEditor. The results are presented in Table 4. GFlowNet and DM generate 1000 sequences. GFNSeqEditor also generates 1000 sequences by editing each of 1000 samples in the test dataset. As can be seen, GFNSeqEditor

Table 3: Performance of GFNSeqEditor and Ledidi with 100 elements of each sequence masked for editing for CRE dataset.

Algorithms	PI	EP(%)	Diversity	PI	EP(%)	Diversity
Ledidi	0.52	18.69	38.34	0.26	14.39	37.45
GFNSeqEditor	**4.79**	**17.89**	32.30	**4.05**	**14.19**	25.52

Table 4: Performance of GFNSeqEditor, GFlowNet and DM on generating new sequences for CRE dataset.

Algorithms	Property	Diversity	Distance(%)
DM	1.75	107.38	63.59
GFlowNet	28.20	83.88	54.41
GFNSeqEditor	**29.25**	87.32	**47.34**

achieves higher property than both GFlowNet and DM. It is useful to note that the experimental study by [19] have shown that GFlowNet outperforms state-of-the-art sequence design methods. For each sequence generated by GFlowNet and DM, the distance to the test set is measured as the distance between the generated sequence and its closest counterpart in the test set. On average, the distance between sequences generated by GFlowNet and the test set is 54.34%, while for DM, it is 63.59%. GFNSeqEditor achieves superior performance by editing, on average, 47.34% of a sequence in the test dataset. The distance between test set and generated sequences by GFlowNet and DM cannot be controlled. As it is studied in Figures 3 and 7, the amounts of edits performed by GFNSeqEditor can be controlled by hyperparameters δ, λ and σ.

E.5 Supplementary Discussion and Results for Sequence Combination

Algorithm 2 presents the GFNSeqEditor for combining two sequences in order to obtain a new sequence whose length is the length of shorter sequence. In Figure 5, we depict the distributions of input sequence lengths and properties, alongside the lengths and properties of the outputs generated by GFNSeqEditor. This scenario pertains to the combination of a long AMP sequence with a short AMP sequence, as detailed in Subsection 5.3. As depicted in Figure 5, the edited sequences produced by GFNSeqEditor exhibit property distributions akin to those of the lengthy input AMP sequences. Simultaneously, these edited sequences are considerably shorter than the original long input sequences. This highlights GFNSeqEditor's effectiveness in shortening lengthy AMP sequences while preserving their inherent properties.

Furthermore, Table 6 provides results for combining pairs of AMP sequences as well as pairs consisting of an AMP sequence and a non-AMP sequence. In both cases, GFNSeqEditor generates a sequence with a length matching that of the longer sequence. When combining two AMP sequences, GFNSeqEditor produces new sequences with higher properties than their parent sequences, maintaining an average resemblance of over 60% to each parent. Additionally, GFNSeqEditor can be applied to combine a non-AMP sequence with an AMP sequence, offering the advantage of rendering the edited non-AMP sequence more akin to a genuine AMP sequence. The results in Table 6 demonstrate that GFNSeqEditor substantially enhances the properties of non-AMP sequences, surpassing the properties of their AMP parents. Furthermore, on average, 35% of the edited sequences bear a resemblance to their AMP parents.

F Supplementary Related Works

GFlowNets, initially proposed by [5], were introduced as a reinforcement-learning (RL) algorithm designed to expand upon maximum-entropy RL, effectively handling scenarios with multiple paths leading to a common state. However, recent studies have redefined and generalized its scope, describing it as a general framework for amortized inference with neural networks [32, 20, 60, 56].

There has been a recent surge of interest in employing GFlowNets across various domains. Noteworthy examples include its utilization in molecule discovery [5], Bayesian structure learning [11, 36], and

Algorithm 2 GFNSeqEditor for combining two sequences to shorten the length of longer sequence.

```
1: Input: $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with lengths $T_1$ and $T_2$, flow function $F_{\boldsymbol{\theta}}(\cdot)$ and parameters $\delta$, $\lambda$ and $\sigma$.
2: Initialize $\hat{\boldsymbol{x}}_{:0}$ as an empty sequence and $T_{\min} = \min\{T_1, T_2\}$.
3: for $t = 1, \ldots, T_{\min}$ do
4:     Assign $x_t = \arg\max_{\{x_{1,t}, x_{2,t}\}}\{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_{1,t}), F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_{2,t})\}$.
5:     Check if $x_t$ is sub-optimal by obtaining $D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma)$ according to (5).
6:     if $D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1$ then
7:         Sample $\hat{x}_t$ according to policy $\pi(\cdot|\hat{\boldsymbol{x}}_{:t-1})$ as follows:
8:         if $T_1 > T_2$ then
9:             $\pi(a|\hat{\boldsymbol{x}}_{:t-1}) = (1-\lambda)\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} + \lambda \mathbf{1}_{a=x_{1,t}}$.
10:        else
11:            $\pi(a|\hat{\boldsymbol{x}}_{:t-1}) = (1-\lambda)\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1}+a')} + \lambda \mathbf{1}_{a=x_{2,t}}$.
12:        end if
13:    else
14:        Assign $\hat{x}_t = x_t$.
15:    end if
16: end for
17: Output: Edited sequence $\hat{\boldsymbol{x}}$.
```

Table 5: Performance of GFNSeqEditor for sequence reduction on AMP dataset in terms of variation in property, edit percentage of long sequences (EPLS), edit percentage of short sequences (EPSS), and percentage of length reduction in the long sequences.

Input Property	Output Property	EPLS(%)	EPSS(%)	Sequence Reduction(%)
0.65	0.67	35.96	44.65	63.23

graph explainability [29]. Recognizing its significance, several studies have emerged to enhance the learning efficiency of GFlowNets [6, 32, 30, 43] since the introduction of the flow matching learning objective by [5]. Moreover, GFlowNets have demonstrated adaptability in being jointly trained with energy and reward functions [57]. [38] introduce intrinsic exploration rewards into GFlowNets, addressing exploration challenges within sparse reward tasks. A couple of recent studies try to extend GFlowNets to stochastic environments, accommodating stochasticity in transition dynamics [39] and rewards [58]. Several novel GFlowNet training methodologies have been recently proposed in [22, 26, 37, 14]. The application of GFlowNets when a predefined reward function is not accessible is explored in [8]. Distributed training of GFlowNets is discussed in [45]. Accelerating GFlowNet training is investigated in [25]. Moreover, [27] employs GFlowNets for designing DNA-encoded libraries. To reduce the need for expensive reward evaluations, [24] proposes a new GFlowNet-based method for molecular optimization.

The aforementioned works have primarily focused on theoretical developments of GFlowNet and its application in molecular generation, without directly addressing the challenges associated with sequence design or editing. In a departure from this trend, and inspired by Bayesian Optimization, [19] proposed a new active learning algorithm based on GFlowNets, i.e. GFlowNet-AL to design novel biological sequences. GFlowNet-AL [19] utilizes the epistemic uncertainty of the surrogate model within its reward function, guiding the GFlowNet towards the optimization of promising yet less-explored regions within the state space. This approach fosters the generation of a diverse set of *de novo* sequences from scratch and token-by-token. Unlike GFNSeqEditor, it lacks the capability to edit input seed sequences and combine multiple sequences. This distinction underscores the unique contribution of GFNSeqEditor in addressing the sequence editing problem, positioning it as a valuable addition to the existing literature on GFLowNet.

G Societal Impact

Biological sequence optimization and design hold transformative potential for biotechnology and health, offering enhanced therapeutic solutions and a vast range of applications. Techniques that

Table 6: Performance of GFNSeqEditor for sequence combination over AMP dataset in terms of property improvements of first (PI-S_1) and second (PI-S_2) sequences, edit percentages of first (EP-S_1) and second (EP-S_2) sequences, and diversity.

Seq_1	Seq_2	PI-S_1	PI-S_2	EP-S_1(%)	EP-S_2(%)	Diversity
AMP	AMP	0.05	0.06	36.10	39.47	7.29
non-AMP	AMP	0.41	0.04	41.41	65.39	12.77

enable refining sequences can lead to advancements like elucidating the role of individual gene products, rectifying genetic mutations in afflicted tissues, and optimizing properties of peptides, antibodies, and nucleic-acid therapeutics. However, the dual-edged nature of such breakthroughs must be acknowledged, as the same research might be misappropriated for unintended purposes. Our method can be instrumental in refining diagnostic procedures and uncovering the genetic basis of diseases, which promises a deeper grasp of genetic factors in diseases. Yet, we must approach with caution, as these advancements may unintentionally amplify health disparities for marginalized communities. As researchers, we emphasize the significance of weighing the potential societal benefits against unintended consequences while remaining optimistic about our work's predominant inclination towards beneficial outcomes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction explain that the paper study biological sequence editing using GFlowNet. The abstract and introduction summarize contributions and main results of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In conclusion Section 6 and Appendix G, the paper discusses its limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper provides the full set of assumptions and complete, correct proofs for all theoretical results. Each theorem is proven, with the proofs detailed in the appendices. The main text appropriately references these sections, ensuring that readers can easily locate and verify the proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Section 5 and Appendix E.1, we provide details about implementation and training of baselines and the proposed algorithm.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data is already public, as we stated in the experiments. We will release the code as well upon company approval.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides detailed information about all the training and test details in Section 5 and Appendix E.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper presents Figures 4, 5, and 6, which show the distribution of results generated by the proposed algorithm, illustrating the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix E.1, we specify that all training and inferences, including GFNSeqEditor, have been conducted using a single Nvidia Quadro P6000.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that the research conducted in this paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The societal impact discussion is provided in Appendix G.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We believe that the paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: In Section 5 and Appendix E.1, we properly cited the original papers that produced the code packages or datasets we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.