Biological Sequence Editing with Generative Flow Networks

Anonymous authors
Paper under double-blind review

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 safety and predictability. 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 in order to enhance the desired property. Notably, GFNSeqEditor prioritizes edits with a higher likelihood of substantially improving the desired property. Furthermore, 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 (Li et al., 2020) and offers the potential to rectify genetic mutations in afflicted tissues and cells for therapeutic interventions (Cox et al., 2015). 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 augments safety but also facilitates the predictability and precision of modification outcomes.

Existing machine learning methodologies within the domain of biological sequences have predominantly concentrated on generating novel *de novo* sequences with desired properties. These methods employ diverse techniques such as reinforcement learning (Angermueller et al., 2019), generative adversarial networks (Zrimec et al., 2022), diffusion models Avdeyev et al. (2023), model-based optimization approaches (Trabucco et al., 2021) and generative flow networks (Jain et al., 2022). 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. having misleading predictions from models that are trained on genomic sequences due to out-of-distribution). Despite the paramount importance of editing biological sequences, there has been a noticeable scarcity of research dedicated to addressing this specific aspect.

The most traditional approaches for biological sequence editing are evolution-based methods, where—over many iterations—a starting "seed" sequence is randomly mutated, and only the best sequence (i.e., highest desired property) is kept for the next round (Arnold, 1998; Sinai et al., 2020; Taskiran et al., 2022); however, the utilization of these approaches necessitates the evaluation of numerous candidate edited sequences 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 limited by the reliability of the proxy model. Beyond evolution-based methods, a perturbation-based editing method known as Ledidi has been introduced by Schreiber et al. (2020). By treating sequence editing as an optimization task, Ledidi learns to perturb specific positions within a given sequence. Akin to evolution-based models, Ledidi's effectiveness is contingent on the quality of the proxy model, which can compromise Ledidi's performance if the proxy model lacks sufficient generalizability for unseen sequences. Furthermore, both evolution-based methods and Ledidi only perform local searches in sequence space, and as a result they suffer from low sample efficiency.

Generative flow networks (GFlowNets) (Bengio et al., 2021; 2023) are a generative approach known for their capacity to sequentially generate new objects. GFlowNets have demonstrated remarkable performance in the generation of novel biological sequences from scratch (Jain et al., 2022). Drawing inspiration from the emerging field of GFlowNets, this paper introduces a novel biological-sequence editing algorithm: GFNSeqEditor. Leveraging a pre-trained flow function from the GFlowNet (acquired through training on a sequence dataset), GFNSeqEditor assesses the potential for significant property enhancement within a given sequence. GFNSeqEditor iteratively identifies and subsequently edits specific positions in the input sequence to increase the target property. More precisely, using the trained flow function, GFNSeqEditor first identifies positions in the seed sequence which requires editing. GFNSeqEditor then constructs a stochastic policy using the flow function to select a substitution from the available options for the identified positions. Diversity holds significant importance when suggesting novel biological sequences (Mullis et al., 2019), and our stochastic approach empowers GFNSeqEditor to generate a diverse set of edited sequences for each input sequence. This is particularly crucial that the proposed sequences exhibit diversity and cover as much as possible the modes of a goodness function. This approach maximizes the likelihood that, ultimately, at least one of the edited sequences will prove effective.

In contrast to evolution-based methods and Ledidi, GFNSeqEditor does not engage in local searches. Instead, it relies on a pre-trained flow function that amortizes the search cost over the learning process, allocating probability mass across entire space to facilitate exploration and diversity. Unlike existing aforementioned *de novo* sequence generative methods, the proposed GFNSeqEditor distinguishes itself by the ability to create sequences that closely resemble existing natural sequences. More discussion about the related works can be found in Appendix D.

In summary, this paper makes the following contributions: 1) 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. 2) We theoretically analyze the properties of the sequences edited through GFNSeqEditor, deriving a lower bound on the property enhancement. Additionally, we demonstrate that the upper bound for the number of edits performed by GFNSeqEditor can be controlled through the adjustment of hyperparameters (subsection 3.3). 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 4.1). 4) 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 4.2). 5) 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 4.3).

2 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 the sequence property. 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 (Bengio et al., 2021;

2023) to identify positions that require editing and subsequently generate edits for those positions. The following subsections present preliminaries on GFlowNets.

2.1 Generative Flow Networks

Generative Flow Networks (GFlowNets) 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

$$P_F(\boldsymbol{s}) = \frac{\sum_{\forall \tau \in \mathbb{T} : \boldsymbol{s} \in \tau} F(\tau)}{Z}. \tag{1}$$

Then, the forward transition probability from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ can be obtained as

$$P_F(\boldsymbol{s}'|\boldsymbol{s}) = \frac{F(\boldsymbol{s} \to \boldsymbol{s}')}{F(\boldsymbol{s})}. \tag{2}$$

The trajectory flow $F(\cdot)$ is called a consistent flow if for any state $\boldsymbol{s}$ it satisfies

$$\sum_{\forall \boldsymbol{s}' : \boldsymbol{s}' \to \boldsymbol{s}} F(\boldsymbol{s}' \to \boldsymbol{s}) = \sum_{\forall \boldsymbol{s}'' : \boldsymbol{s} \to \boldsymbol{s}''} F(\boldsymbol{s} \to \boldsymbol{s}''), \tag{3}$$

which constitutes that the in-flow and out-flow of state $\boldsymbol{s}$ are equal. Bengio et al. (2021) 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.

2.2 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. In order to learn the flow

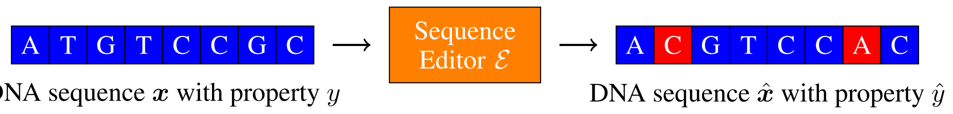
A T G T C C G C
Sequence Editor $\mathcal{E}$
A C G T C C A C
DNA sequence $\boldsymbol{x}$ with property y
DNA sequence $\hat{\boldsymbol{x}}$ with property $\hat{y}$

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 one letter from the alphabet [`'A'`, `'C'`, `'T'`, `'G'`]. The editor function $\mathcal{E}$ accepts the starting sequence and determines that the second and seventh bases require editing (highlighted in red). Then, $\mathcal{E}$ modifies the bases at these identified locations.

function that can provide consistency condition, Bengio et al. (2021) formulates flow-matching loss function as follows:

$$\mathcal{L}_{\text{FM}}(\boldsymbol{s};\boldsymbol{\theta}) = \left(\log \frac{\sum_{\forall \boldsymbol{s}':\boldsymbol{s}'\to\boldsymbol{s}} F_{\boldsymbol{\theta}}(\boldsymbol{s}'\to\boldsymbol{s})}{\sum_{\forall \boldsymbol{s}'':\boldsymbol{s}\to\boldsymbol{s}''} F_{\boldsymbol{\theta}}(\boldsymbol{s}\to\boldsymbol{s}'')}\right)^2. \tag{4}$$

Moreover, as an alternative objective function, Malkin et al. (2022) introduces trajectory balance as:

$$\mathcal{L}_{\text{TB}}(\boldsymbol{s};\boldsymbol{\theta}) = \left(\log \frac{Z_{\boldsymbol{\theta}} \prod_{\boldsymbol{s}\to\boldsymbol{s}'} P_{F_{\boldsymbol{\theta}}}(\boldsymbol{s}'|\boldsymbol{s})}{R(\boldsymbol{x})}\right)^2 \tag{5}$$

where $Z_{\boldsymbol{\theta}}$ is a learnable parameter. The trajectory-balance objective function in equation 5 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 equation 4 and equation 5 to train the GFlowNet model.

3 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 an 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 $\pi(\cdot)$.

3.1 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 if x_{t+1} should be used or not. Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, given $\boldsymbol{x}_{:t}$, GFlowNet would evaluate 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. For instance for the DNA sequence $\boldsymbol{x} =$ `'ATGTCCGC'`, appending token $a =$ `'C'` to $\boldsymbol{x}_{:2}$, we get $\boldsymbol{x}_{:2}+a =$ `'ATC'`. 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 2, 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, if $F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t}+a_1) > F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t}+a_2)$, this 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 we have

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t}+x_{t+1}) < \delta \max_{a\in\mathbb{A}} F_{\boldsymbol{\theta}}(\boldsymbol{x}_{:t}+a) \tag{6}$$

where $0 \leq \delta \leq 1$ is a hyperparameter. Choosing larger δ, it is more probable that the algorithm identifies x_{t+1} as sub-optimal. From equation 6 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.

3.2 SEQUENCE EDITING WITH GFNSEQEDITOR

Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, GFNSeqEditor iteratively identifies and edits positions in a seed sequence. Subsection 3.1 presented a simple function for determining if a position x_{t+1} in a sequence should be edited to improve the target property value (equation 6). Based on this intuition, we now modify equation 6 to formally define the sub-optimal-position identification function $D(\cdot)$ used by 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 equation 8.
 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 equation 9.
 7:     else
 8:         Assign $\hat{x}_t = x_t$.
 9:     end if
10: end for
11: Output: Edited sequence $\hat{\boldsymbol{x}}$.
```

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 order to perform exploration in sub-optimal identification, modifying condition in equation 6, the sub-optimal identifier function $D(\cdot)$ checks the following condition:

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} < \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \nu \tag{7}$$

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 3.3 and Appendix E. The inclusion of additive noise ν on the right-hand side of equation 7 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:

$$D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = \begin{cases} 1 & \text{If the condition in equation 7 is met} \\ 0 & \text{Otherwise} \end{cases}. \tag{8}$$

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:

$$\pi(a|\hat{\boldsymbol{x}}_{:t-1}) = (1 - \lambda) \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \lambda \mathbf{1}_{a = x_t} \tag{9}$$

where $0 \leq \lambda < 1$ is a regularization coefficient and $\mathbf{1}_{a=x_t}$ denotes indicator function and is 1 if $a = x_t$. The regularization parameter λ allows tuning the sampling process to favor the original sequence. Choosing larger λ leads to obtaining smaller number of edits. The policy in equation 9 constitutes a trade-off between increasing the target property and decreasing the distance between the edited sequence $\hat{\boldsymbol{x}}$ and the original sequence $\boldsymbol{x}$. Specifically, the first term in the right hand side of equation 9 samples actions with probability proportional to their flow. The second term in the right hand side of equation 9 increases the likelihood of choosing the original x_t to reduce the distance between the edited sequence and the original one. Let $\tilde{x}_t$ be the action sampled by the policy π in equation 9. In summary, the t-th element in the edited sequence can be written as

$$\hat{x}_t = D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma)\tilde{x}_t + (1 - D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma))x_t. \tag{10}$$

Therefore, at each step t, the edited sequence is updated as $\hat{\boldsymbol{x}}_{:t} = \hat{\boldsymbol{x}}_{:t-1} + \hat{x}_t$. This continues until the step T is reached 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 the proposed algorithm GFNSeqEditor.

3.3 Analysis

This subsection analyzes the reward of the edited sequence and the number of edits performed by GFNSeqEditor. Specifically, the bounds for the reward of the edited sequence 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 sequence by GFNSeqEditor.

Theorem 1. *Let T be the length of the original sequence $\boldsymbol{x}$. The expected reward of $\hat{\boldsymbol{x}}$ edited sequence by GFNSeqEditor given $\boldsymbol{x}$ is bounded from below as*

$$\mathbb{E}[R(\hat{\boldsymbol{x}})|\boldsymbol{x}] \geq \left(1 - \Phi(\frac{1-\delta}{\sigma})\right)(1-\lambda)R_{F,T} \tag{11}$$

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 1 is deferred to Appendix A. From Theorem 1, we can deduce that greater values of δ and σ correspond to larger lower bounds for the reward of the edited sequence. Furthermore, Theorem 1 demonstrates that a reduction in λ results in a larger lower bound for the reward. According to equation 9, a lower λ results in a higher editing probability. The following theorem shows the connections between the number of edits and the hyperparameters of GFNSeqEditor.

Theorem 2. *The expected distance between the edited sequence $\hat{\boldsymbol{x}}$ by GFNSeqEditor and the original sequence $\boldsymbol{x}$ is bounded from above as*

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

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

The proof for Theorem 2 is available in Appendix B. Theorem 2 demonstrates that larger values of δ yields a higher upper bound for the expected distance between the edited and original sequences, and conversely, a lower values of λ and σ leads to an increase in this expected distance. Theorems 1 and 2 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 that reduce the expected number of edits, an increase in the number of edits corresponds to a larger lower bound for the reward. More analysis on property improvement upper bound and distance lower bound can be found in Appendix E.

4 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.

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 C.
• **Edit Percentage (EP):** The average Levenshtein distance between $\boldsymbol{x}$ and edited sequences normalized by the length of $\boldsymbol{x}$ expressed as $\frac{1}{n_e T}\sum_{i=1}^{n_e}\operatorname{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}}_i)$.
• **Diversity:** For each sequence $\boldsymbol{x}$, the diversity among edited sequences can be obtained as $\frac{2}{n_e(n_e-1)}\sum_{i=1}^{n_e-1}\sum_{j=i+1}^{n_e}\operatorname{lev}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j)$.

We compared GFNSeqEditor to several baselines, including Directed Evolution (DE) (Sinai et al., 2020), Ledidi (Schreiber et al., 2020), GFlowNet (Jain et al., 2022), 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. Inspired by graph-to-graph translation for molecular optimization in Jin et al. (2019), 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

Table 1: Performance of GFNSeqEditor compared to the baselines in terms of property improvement (PI), edit percentage (EP) and diversity on TFbinding, AMP, and CRE datasets. Higher PI with a lower EP is preferable.

	TFbinding			AMP			CRE		
Algorithms	PI	EP(%)	Diversity	PI	EP(%)	Diversity	PI	EP(%)	Diversity
DE	0.12	25.00	3.01	0.11	33.82	13.67	0.63	22.93	62.07
Ledidi	0.06	27.80	1.25	0.18	34.79	11.65	1.36	22.13	50.49
GFlowNet-E	0.11	28.35	2.10	0.28	35.68	3.42	4.24	22.73	37.06
Seq2Seq	0.03	41.98	-	0.21	78.05	-	-	-	-
GFNSeqEditor	**0.14**	**24.27**	**3.84**	**0.33**	**34.49**	**14.34**	**9.90**	**21.90**	40.41

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 translator model is then trained to map each low-property sequence to its high-property pair. Essentially, Seq2Seq baseline endeavors to map an input sequence to a similar sequence with superior property. Furthermore, we adapted GFLowNet-AL for sequence editing and named it GflowNet-E. 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, GFlowNet-E takes in the initial 70%, 65% and 60% of elements respectively from the input sequence $\boldsymbol{x}$, and generates the remaining elements using the pre-trained flow function. Detailed information about the implementation of the baselines can be found in Appendix C.1.

To train models associated with baselines and the proposed GFNSeqEditor, we partition each dataset into a 72% training set and an 18% validation set. The remaining 10% constitutes the test set, employed to evaluate the performance of methods in sequence editing tasks. The trained flow function $F_{\boldsymbol{\theta}}(\cdot)$ employed by GFlowNet-E and the proposed GFNSeqEditor, is an MLP comprising 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 for training the flow function. Detailed information about training the flow function can be found in Appendix C.1.

4.1 Sequence Editing

Table 1 presents the performance of GFNSeqEditor and other baselines on TFbinding, AMP and CRE datasets [1]. We set GFNSeqEditor and all baselines except for Seq2Seq to create 10 edited sequences for each input sequence. However, our 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 provides a visualization of property improvement achieved by GFNSeqEditor, DE, and Ledidi across a range of edit percentages. As evident from Table 1 and Figure 2, GFNSeqEditor surpasses all baselines in terms of achieving substantial property improvements with a minimal number of edits when compared to the other methods. This superior performance is attributed to GFNSeqEditor's utilization of a pre-trained flow function from GFlowNet, enabling it to attain notably higher property improvements than DE and Ledidi, which relies on local search techniques. 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 equation 9 for editing enables GFNSeqEditor to involve global information contained in $F_{\boldsymbol{\theta}}(\cdot)$ about the entire space of sequences. However, both DE and Ledidi perform local search such that at each iteration they perturb the edited sequence obtained from the previous iteration and then they evaluate their perturbed sequences using the proxy model to update the edited sequence. Furthermore, GFNSeqEditor achieves larger property improvement than GFlowNet-E. The GFNSeqEditor identifies and edits sub-optimal positions within a seed sequence using equation 7 while GFlowNet-E only edits the tail of the input seed sequence. This indicates the effectiveness of sub-optimal position identification of GFNSeqEditor. In addition to sequence editing, the proposed GFNSeqEditor is able to generate new sequences. The performance of GFNSeqEditor in generating new sequences is studied in Appendix C.4.

[1]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).

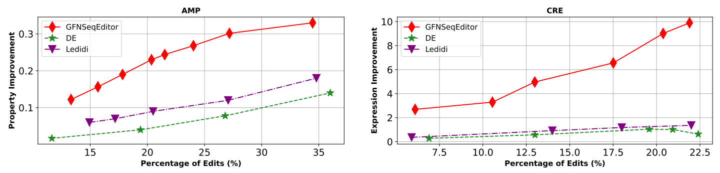

Figure 2: Property improvement of AMP **(left)** and CRE **(right)** with respect to edit percentage.

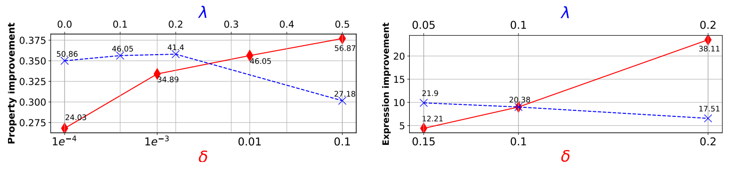

Figure 3: Studying the effect of heyperparameters δ and λ on the performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets. The marker values are edit percentages.

Furthermore, in Figure 3, we present the property improvement achieved by GFNSeqEditor along with edit percentage across various choices of hyperparameters δ and λ. The figure 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 4, 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 1 and 2 in section 3.3 as well as Theorem 3 in Appendix E.

4.2 Assisting Sequence Generation

GFNSeqEditor can complement generative models to enhance the generation of novel sequences. In this subsection, we incorporate a diffusion model (DM) for sequence generation, with further details available in Appendix C.2. The sequences generated unconditionally 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 as 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. Moreover, we show the CDF of properties for sequences generated by the DM, the GFlowNet, and DM+GFNSeqEditor in Figure 5. 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 5 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 performed by GFNSeqEditor.

4.3 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 applications where shortening relatively lengthy sequences is advantageous while retaining desired properties (see e.g. Xu et al. (2021); Zhao et al. (2023)). GFNSeqEditor accomplishes this by merging the longer sequence with a shorter one. The resultant sequence maintains similarities 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 C.5 describes using GFNSeqEditor to combine two sequences to shorten the longer one. We evaluate GFNSeqEditor's performance in combining

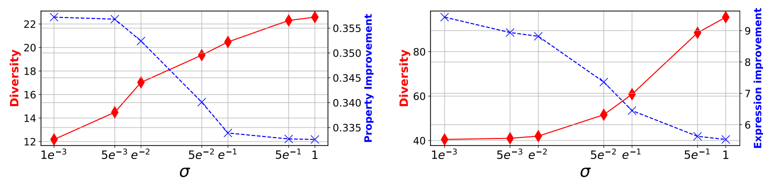

Figure 4: Studying the effect of hyperparameter σ on the diversity and performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets.

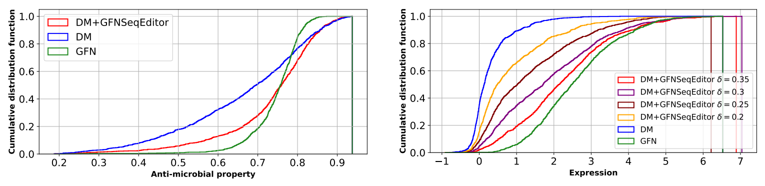

Figure 5: CDF of generated sequence properties for AMP **(left)** and CRE **(right)**. A right-shifted curve indicates that the model is generating more sequences that are high in the target property.

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 pair consists of a long AMP sequence and the closest short sequence to the long one, chosen from among all short sequences, with an AMP property exceeding 0.7. Table 3 and Figure 7 in Appendix C.5 present the results of sequence combination for the purpose of reducing the length of long sequences. As indicated in Table 3, 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 exhibit a resemblance to both long and short sequences, with a Levenshtein similarity of approximately 65% to long sequences and 55% to short sequences.

Table 3: 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(%)
GFNSeqEditor	0.65	0.67	35.96	44.65	63.23

5 Conclusions

This paper introduces GFNSeqEditor, a sequence-editing method built upon GFlowNet. Given an input 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 sequence-editing baselines in terms of property enhancement while maintaining a similar amount of edits. Moreover, the empirical findings highlight the versatility of GFNSeqEditor, showcasing its applications beyond single-sequence editing. Furthermore, GFNSeqEditor can effectively complement other generative models to generate sequences with improved properties and increased diversity. It can also be employed to combine two sequences into a new one with desired properties. 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.

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

References

Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*, 2019.

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

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. *arXiv preprint arXiv:2305.10699*, 2023.

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

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27381–27394, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

David Benjamin Turitz Cox, Randall Jeffrey Platt, and Feng Zhang. Therapeutic genome editing: prospects and challenges. *Nature medicine*, 21(2):121–131, 2015.

Hamid Dadkhahi, Jesus Rios, Karthikeyan Shanmugam, and Payel Das. Fourier representations for black-box optimization over categorical variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10156–10165, 2022.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pp. 518–528. PMLR, 2022.

Sager J Gosai, Rodrigo I Castro, Natalia Fuentes, John C Butts, Susan Kales, Ramil R Noche, Kousuke Mouri, Pardis C Sabeti, Steven K Reilly, and Ryan Tewhey. Machine-guided design of synthetic cell type-specific cis-regulatory elements. *bioRxiv*, pp. 2023–08, 2023.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. URL `https://github.com/hojonathanho/diffusion`.

Samuel C Hoffman, Vijil Chenthamarakshan, Kahini Wadhawan, Pin-Yu Chen, and Payel Das. Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, 4(1):21–31, 2022.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 9786–9801, Jul 2022.

Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*, 2019.

Hongyi Li, Yang Yang, Weiqi Hong, Mengyuan Huang, Min Wu, and Xia Zhao. 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.

Wenqian Li, Yinchuan Li, Zhigang Li, Jianye HAO, and Yan Pang. 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`.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.

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

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

Megan M Mullis, Ian M Rambo, Brett J Baker, and Brandi Kiel Reese. Diversity, ecology, and prevalence of antimicrobials in nature. *Frontiers in microbiology*, pp. 2518, 2019.

Mizu Nishikawa-Toomey, Tristan Deleu, Jithendaraa Subramanian, Yoshua Bengio, and Laurent Charlin. Bayesian learning of causal structure and mechanisms with gflownets and variational bayes. *arXiv preprint arXiv:2211.02763*, 2022.

Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative augmented flow networks. *arXiv preprint arXiv:2210.03308*, 2022.

Ling Pan, Dinghuai Zhang, Moksh Jain, Longbo Huang, and Yoshua Bengio. Stochastic generative flow networks. *arXiv preprint arXiv:2302.09465*, 2023.

Malak Pirtskhalava, Anthony A Amstrong, Maia Grigolava, Mindia Chubinidze, Evgenia Alimbarashvili, Boris Vishnepolsky, Andrei Gabrielian, Alex Rosenthal, Darrell E Hurt, and Michael Tartakovsky. 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.

Jacob Schreiber, Yang Young Lu, and William Stafford Noble. 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`.

Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. *arXiv preprint arXiv:2305.07170*, 2023.

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

Yang Song, Jascha Sohl-Dickstein, Google Brain, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *arXiv preprint arXiv:2011.13456*, 2021.

Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, pp. 769–778. PMLR, 2020.

Ibrahim Ihsan Taskiran, Katina I Spanier, Valerie Christiaens, David Mauduit, and Stein Aerts. Cell type directed design of synthetic enhancers. *bioRxiv*, pp. 2022–07, 2022.

Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Xiaoshu Xu, Augustine Chemparathy, Leiping Zeng, Hannah R. Kempton, Stephen Shang, Muneaki Nakamura, and Lei S. Qi. Engineered miniature crispr-cas system for mammalian genome regulation and editing. *Molecular Cell*, 81(20):4333–4345.e4, 2021.

Dinghuai Zhang, Jie Fu, Yoshua Bengio, and Aaron Courville. Unifying likelihood-free inference with black-box optimization and beyond. *arXiv preprint arXiv:2110.03372*, 2021.

Dinghuai Zhang, Ricky TQ Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with gflownets. *arXiv preprint arXiv:2209.02606*, 2022a.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pp. 26412–26428. PMLR, 2022b.

Dinghuai Zhang, Ling Pan, Ricky TQ Chen, Aaron Courville, and Yoshua Bengio. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*, 2023.

Feiyu Zhao, Tao Zhang, Xiaodi Sun, Xiyun Zhang, Letong Chen, Hejun Wang, Jinze Li, Peng Fan, Liangxue Lai, Tingting Sui, et al. A strategy for cas13 miniaturization based on the structure and alphafold. *Nature Communications*, 14(1):5545, 2023.

Heiko Zimmermann, Fredrik Lindsten, Jan-Willem van de Meent, and Christian A Naesseth. A variational perspective on generative flow networks. *arXiv preprint arXiv:2210.07992*, 2022.

Jan Zrimec, Xiaozhi Fu, Azam Sheikh Muhammad, Christos Skrekas, Vykintas Jauniskis, Nora K Speicher, Christoph S Börlin, Vilhelm Verendel, Morteza Haghir Chehreghani, Devdatt Dubhashi, et al. Controlling gene expression with deep generative design of regulatory dna. *Nature communications*, 13(1):5099, 2022.

A Proof of Theorem 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 equation 9. According to equation 9, 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 equation 7 and equation 8, 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 equation 20 and equation 17 with equation 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 equation 21 with equation 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 equation 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 equation 23 with equation 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 equation 11. Moreover, the upper bound of property improvement by the proposed GFNSeqEditor is analyzed in Appendix E.

B Proof of Theorem 2

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 equation 9, we have $\hat{x}_t = x_t$ with probability

$$(1-\lambda)\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \lambda. \tag{25}$$

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

$$\Pr[\hat{x}_t \neq x_t] = \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1-\lambda)\left(1 - \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')}\right). \tag{26}$$

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

$$\Pr[\hat{x}_t \neq x_t] \leq \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1-\lambda). \tag{27}$$

Moreover, if we have

$$\nu \leq \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} - \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \tag{28}$$

then $D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0$. Furthermore, the right hand side of equation 28 can be bounded from below as

$$\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + x_t)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} - \delta \max_{a \in \mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \geq -\delta. \tag{29}$$

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

$$\Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1] = 1 - \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{30}$$

Combining equation 30 with equation 27, we arrive at

$$\Pr[\hat{x}_t \neq x_t] \leq \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(1-\lambda). \tag{31}$$

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

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] = \sum_{t=1}^{T} \Pr[\hat{x}_t \neq x_t]. \tag{32}$$

Thus, combining equation 32 with equation 31, we can write that

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] \leq \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(1-\lambda)T \tag{33}$$

which proves the Theorem.

C Supplementary Experimental Results and Details

This appendix provides a comprehensive overview of the experimental setup in Section 4 and presents additional supplementary experimental results.

C.1 Implementation Details

C.1.1 Datasets

Detailed information about the datasets can be foud below:

- **TFbinding:** The dataset is taken from Barrera et al. (2016) and contains all possible DNA sequences with length 8. The vocabulary is the four DNA bases, $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. 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 (Pirtskhalava et al., 2020), is curated following the approach outlined by Jain et al. (2022). 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)(Gosai et al., 2023) 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.

C.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 Angermueller et al. (2019); Jain et al. (2022), 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 (Jain et al., 2022) for AMP dataset. Furthermore, for CRE dataset we leverage the Malinois model (Gosai et al., 2023) which is a deep convolutional neural network (CNN) for cell type-informed CRE activity prediction of any arbitrary sequence.

C.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 $2,000$.

Seq2Seq. In order to implement Seq2Seq baseline we use a standard transformer (Vaswani et al., 2017) 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.

C.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 Jain et al. (2022). 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 Jain et al. (2022).

C.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) (Song et al., 2021). 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 Ho et al. (2020). 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, 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 Vaswani et al. (2017)) 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 Song et al. (2021), using 1000 time steps from T to 0. We then rounded all outputs to the nearest integer to recover the one-hot encoded sample.

Table 4: 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 5: 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**

C.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.

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 4 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 4, GFNSeqEditor achieves significantly greater property improvement than Ledidi while utilizing a lower edit percentage.

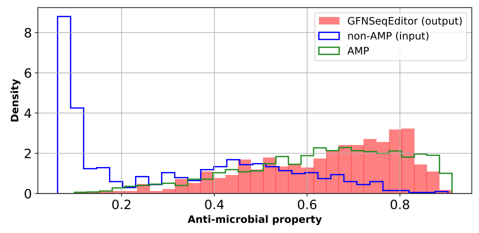

Figure 6: GFNSeqEditor shifts the distribution of non-AMP inputs to the known AMPs.

C.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 5. 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 achieves higher property than both GFlowNet and DM. It is useful to note that the experimental study by Jain et al. (2022) 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 4, the amounts of edits performed by GFNSeqEditor can be controlled by hyperparameters δ, λ and σ.

C.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 7, 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

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 equation 8.
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 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

short AMP sequence, as detailed in subsection 4.3. As depicted in Figure 7, 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.

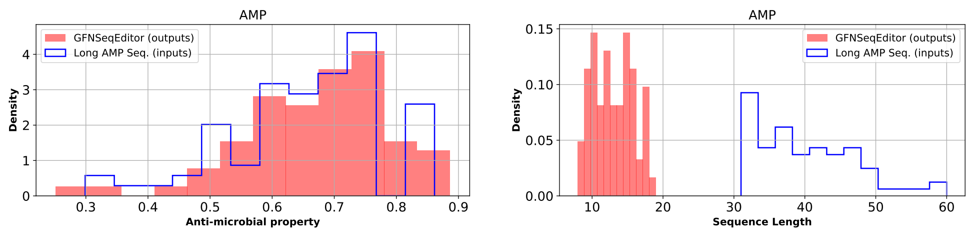

Figure 7: GFNSeqEditor effectively reduces the length of AMP sequence inputs (**right**) while keeping their properties intact (**left**).

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.

D Related Works

Generative Flow Networks. GFlowNets, initially proposed by Bengio et al. (2021), 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 (Malkin et al., 2023; Jain et al., 2023; Zimmermann et al., 2022; Zhang et al., 2022a).

There has been a recent surge of interest in employing GFlowNets across various domains. Noteworthy examples include its utilization in molecule discovery (Bengio et al., 2021), Bayesian structure learning (Deleu et al., 2022; Nishikawa-Toomey et al., 2022), and graph explainability (Li et al., 2023). Recognizing its significance, several studies have emerged to enhance the learning efficiency of GFlowNets (Bengio et al., 2023; Malkin et al., 2023; Madan et al., 2023; Shen et al., 2023) since the introduction of the flow matching learning objective by Bengio et al. (2021). Moreover, GFlowNets have demonstrated adaptability in being jointly trained with energy and reward functions (Zhang et al., 2022b). Pan et al. (2022) 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 (Pan et al., 2023) and rewards (Zhang et al., 2023).

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, Jain et al. (2022) proposed a new active learning algorithm based on GFlowNets, i.e. GFlowNet-AL to design novel biological sequences. GFlowNet-AL (Jain et al., 2022) 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.

Sequence Generation. The generation of biological sequences has been tackled using a diverse range of methods, including reinforcement learning (Angermueller et al., 2019), Bayesian optimization (Terayama et al., 2021), deep generative models for search and sampling (Hoffman et al., 2022), generative adversarial networks (Zrimec et al., 2022), diffusion models Avdeyev et al. (2023), optimization with deep model-based approaches (Trabucco et al., 2021), adaptive evolutionary strategies (Hansen, 2006; Swersky et al., 2020), likelihood-free inference (Zhang et al., 2021), and surrogate-based black-box optimization (Dadkhahi et al., 2022), and GFlowNet (Jain et al., 2022).

Bengio et al. (2023) demonstrated that GFlowNet offers improvements over existing sequence-generation methods by amortizing the search cost over the learning process, allocating probability mass across the entire sequence space to facilitate exploration and diversity, enabling the use of imperfect data, and efficiently scaling with data through the exploitation of structural patterns in function approximation. It is important to note that all these sequence-generation methods—including GFLowNet—generate sequences from scratch. However, *ab initio* generation carries the risk of *deviating too significantly* from naturally occurring genomic sequences, which can compromise safety and predictability. In contrast, our proposed method tends to enhance the target property of sequences while preserving their similarity to naturally occurring sequences, ensuring safety and predictability. Consequently, biological sequence design may not be suitable for the purpose of biological sequence editing.

E Supplementary Analysis

This section provides supplementary analysis to uncover additional insights into the impact of parameters on algorithm performance, complementing the findings presented in Theorems 1 and 2. The following Theorem obtains the expected property improvement upper bound of the proposed GFNSeqEditor.

Theorem 3. *Let $\mathbb{S}^*_{\boldsymbol{x}}$ be the set of all sequences in $\mathbb{T}_T$ 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*

$$\mathbb{E}[PI|\boldsymbol{x}] \leq \sum_{\boldsymbol{w}\in\mathbb{S}^*_{\boldsymbol{x}}} \left(1 - \Phi(-\frac{\delta}{\sigma})\right)(p_{\boldsymbol{w}} - y) \tag{34}$$

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

Proof. The expected property improvement of GFNSeqEditor can be obtained as

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] = \sum_{\boldsymbol{w}\in\mathbb{T}_T} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{35}$$

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

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] = \sum_{\boldsymbol{w}\in\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y) + \sum_{\boldsymbol{w}\in\mathbb{T}_T\setminus\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{36}$$

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

$$\mathbb{E}[\text{PI}|\boldsymbol{x}] \leq \sum_{\boldsymbol{w}\in\mathbb{S}^*_{\boldsymbol{x}}} \Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}](p_{\boldsymbol{w}} - y). \tag{37}$$

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

$$\Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}] = \prod_{t=1}^{T} \Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}]. \tag{38}$$

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

$$\begin{aligned}\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}] = &\Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1]\pi(w_t|\boldsymbol{w}_{:t-1}) \\ &+ \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 0]\mathbf{1}_{x_t = w_t}.\end{aligned} \tag{39}$$

If $x_t \neq w_t$, according to equation 30 and considering the fact that $\pi(w_t|\boldsymbol{w}_{:t-1}) \leq 1$, the probability in equation 39 can be bounded from above as

$$\Pr[\hat{x}_t = w_t|\hat{\boldsymbol{x}}_{:t-1}, \boldsymbol{x}] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{40}$$

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 equation 38 with equation 40 we can conclude that

$$\Pr[\hat{\boldsymbol{x}} = \boldsymbol{w}|\boldsymbol{x}] \leq 1 - \Phi(-\frac{\delta}{\sigma}). \tag{41}$$

Combining equation 37 with equation 41 proves the Theorem. □

Theorem 3 demonstrates that an increase in δ results in an increase in the upper bound of property improvement, whereas an increase in σ diminishes the upper bound of property improvement. The following theorem obtains the lower bound on the number edits performed by the proposed GFNSeqEditor.

Theorem 4. *Let there exists $\epsilon > 0$ such that the flow function $F_{\boldsymbol{\theta}}(\cdot)$ satisfies*

$$\max_{a\in\mathbb{A}} \frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a'\in\mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} \leq 1 - \epsilon, \forall t \tag{42}$$

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

$$\mathbb{E}[\text{lev}(\boldsymbol{x}, \hat{\boldsymbol{x}})] \geq \left[\epsilon(1 - \lambda)\left(1 - \Phi(\frac{1-\delta}{\sigma})\right)\right] T. \tag{43}$$

Proof. According to equation 26 and the assumption in equation 42, it can be written that

$$\Pr[\hat{x}_t \neq x_t] \geq \Pr[D(\hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = 1](1 - \lambda)\epsilon. \tag{44}$$

Combining equation 44 with equation 20, we get

$$\Pr[\hat{x}_t \neq x_t] \geq \epsilon(1 - \lambda)\left(1 - \Phi(\frac{1-\delta}{\sigma})\right). \tag{45}$$

Summing equation 45 over all elements in the sequence proves the theorem. □

Theorem 4 shows that as δ and σ increase, the lower bound of distance increases. In contrast, increase in λ leads to decrease in lower bound of distance.

F 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 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.