Generative Flow Networks Assisted Biological Sequence Editing

Pouya M. Ghari[†*], **Alex Tseng**[‡], **Gökcen Eraslan**[‡], **Romain Lopez**[‡¶], **Tommaso Biancalani**[‡]

Gabriele Scalia[‡], **Ehsan Hajiramezanali**[‡§]

† University of California Irvine, ‡ Genentech, ¶ Stanford University

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 [16] and offers the potential to rectify genetic mutations in afflicted tissues and cells for therapeutic interventions [7]. The primary objective in biological-sequence editing is to enhance specific biological attributes of a starting seed sequence, while minimizing the number of edits (Figure 1). This reduction in the number of alterations not only augments safety but also facilitates the predictability and precision of modification outcomes.

Several methods have been proposed to address biological-sequence editing, but they have suffered from several limitations. The most traditional approaches 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 [2, 29]. Beyond evolution-based methods, a perturbation-based editing method known as Ledidi has been introduced by [26]. By treating sequence editing as an optimization task, Ledidi learns to perturb specific positions within a given sequence. However, the utilization of these approaches necessitates the evaluation of numerous

[*]Work has been done while interning at Genentech.

[§]Corresponding author: hajiramezanali.ehsan@gene.com.

NeurIPS 2023 Generative AI and Biology Workshop.

candidate edited sequences every iteration. This computational demand can become prohibitively expensive, particularly for lengthy sequences. Additionally, evolution-based methods and Ledidi 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. 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) [5, 6] 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 [13]. 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, 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. Diversity holds significant importance when suggesting novel biological sequences [21], and our stochastic approach empowers GFNSeqEditor to generate a diverse set of edited sequences for each input sequence. 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. More discussion about the related works can be found in Appendix C. 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.

2 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. GFNSeqEditor uses a 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 $\pi(\cdot)$. Preliminaries on GFlowNets can be found in appendix A.

Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, GFNSeqEditor iteratively identifies and edits positions in a seed sequence. 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'`. 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. Using the flow function $F_{\boldsymbol{\theta}}(\cdot)$, given $\hat{\boldsymbol{x}}_{:t-1}$, GFlowNet can evaluate the average reward obtained by appending any possible token to $\hat{\boldsymbol{x}}_{:t-1}$. In this context, each token can be viewed as an action. Let $\hat{\boldsymbol{x}}_{:t-1}+a$ denotes the expanded $\hat{\boldsymbol{x}}_{:t-1}$ 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 $F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)$ the value of action a given $\hat{\boldsymbol{x}}_{:t-1}$ can be evaluated. As discussed in Appendix A, the state flow $F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)$ is proportional to the total reward of all possible sequences that have $\hat{\boldsymbol{x}}_{:t-1} + a$ as their prefix. If the reward resulting from having x_t in the seed sequence is evaluated by $F_{\boldsymbol{\theta}}(\cdot)$ to be relatively small compared to other possible actions, then x_t is considered sub-optimal. In particular, x_t is identified as sub-optimal if we have

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

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.

Algorithm 1 GFNSeqEditor: Sequence Editor using GFlowNet

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

where $0 \leq \delta \leq 1$ is a parameter chosen by the algorithm and $\nu \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable with variance of σ^2. The variance σ^2 is a parameter chosen by the algorithm. From equation 1 it can be inferred that x_t is identified as sub-optimal if its associated out-flow is considerably smaller than the out-flow associated with the best possible action in $\mathbb{A}$. The inclusion of additive noise ν on the right-hand side of equation 1 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:

$$D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma) = \begin{cases} 1 & \text{If the condition in equation 1 is met} \\ 0 & \text{Otherwise} \end{cases}. \tag{2}$$

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:

$$\pi(a|\hat{\boldsymbol{x}}_{:t-1}) = (1-\lambda)\frac{F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a)}{\sum_{a' \in \mathbb{A}} F_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{:t-1} + a')} + \lambda \mathbf{1}_{a=x_t} \tag{3}$$

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, thereby reducing the number of edits. The policy in equation 3 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 3 samples actions with probability proportional to their flow. The second term in the right hand side of equation 3 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 3. In summary, the t-th element in the edited sequence can be written as

$$\hat{x}_t = D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma)\tilde{x}_t + (1 - D(x_t, \hat{\boldsymbol{x}}_{:t-1}; \delta, \sigma))x_t. \tag{4}$$

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 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 the following datasets: TFbinding, AMP, and CRE. The TFbinding and CRE datasets consist of DNA sequences, with lengths of 8 for TFbinding and 200 for CRE, respectively. The AMP dataset comprises protein sequences with variable lengths ranging from 15 to 60 amino acids. Each sequence in the AMP dataset is categorized as either an anti-microbial peptide (AMP) or a non-AMP. Each dataset is partitioned into training, validation, and test samples. Models are trained and validated on the training and validation sets, while algorithm performance is assessed using the test samples. Additional details about the datasets can be found in Appendix B.1. To train models associated with baselines and the proposed GFNSeqEditor, we partition each dataset into a 72%

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
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

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 $F_{\theta}(\cdot)$ employed by 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 B.2. The evaluation of algorithm performance encompasses property improvement (PI), edit percentage (EP), and diversity metrics, with precise definitions provided in Appendix B.1.

We compared GFNSeqEditor to several baselines, including Directed Evolution (DE) [2], Ledidi [26], 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 [15], we implemented another editing baseline which is called Seq2Seq. Essentially, Seq2Seq baseline endeavors to map an input sequence to a similar sequence with superior property. More information about baselines can be found in Appendix B.2.

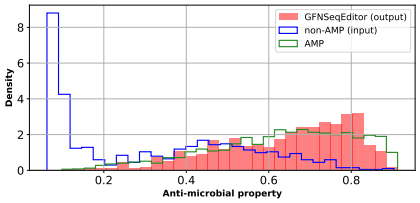

Figure 2: GFNSeqEditor shifts the distribution of non-AMP inputs to the known AMPs.

Table 1 presents the performance of GFNSeqEditor and other baselines on TFbinding, AMP and CRE datasets [3]. 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 3 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 3, 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, which relies on local search techniques for sequence editing. Furthermore, GFNSeqEditor demonstrates a more significant property enhancement compared to Ledidi. Moreover, in Figure 2, 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 2 that GFNSeqEditor shifts the property distribution of input non-AMP sequences towards that of AMP sequences.

4 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. Experimental evaluations using real-world DNA and protein datasets demonstrate that GFNSeqEditor outperforms state-of-the-art sequence-editing baselines in terms of prop-

[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).

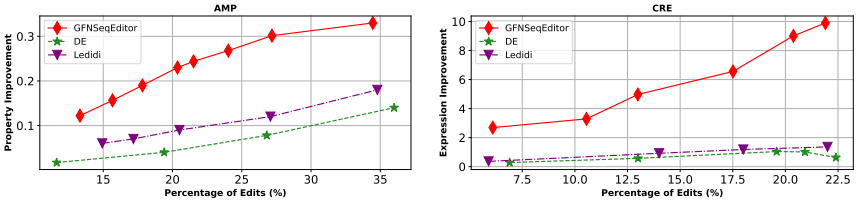

Figure 3: Property improvement of AMP **(left)** and CRE **(right)** with respect to edit percentage.

erty 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] 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.

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

[3] 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.

[4] 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.

[5] 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, pages 27381–27394, 2021.

[6] 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.

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

[8] 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, pages 10156–10165, 2022.

[9] 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*, pages 518–528. PMLR, 2022.

[10] 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*, pages 2023–08, 2023.

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

[12] 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.

[13] 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, pages 9786–9801, Jul 2022.

[14] 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.

[15] 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.

[16] 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.

[17] 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`.

[18] 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*, pages 23467–23483. PMLR, 2023.

[19] 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.

[20] 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.

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

[22] 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.

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

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

[25] 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.

[26] 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`.

[27] 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.

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

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

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

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

[32] 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.

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

[34] 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*, pages 26412–26428. PMLR, 2022.

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

[36] 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.

[37] 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 Preliminaries on GFlowNets

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

$$P_F(\boldsymbol{s}) = \frac{\sum_{\forall \tau \in \mathbb{T} : \boldsymbol{s} \in \tau} F(\tau)}{Z}. \tag{5}$$

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

$$P_F(\boldsymbol{s}'|\boldsymbol{s}) = \frac{F(\boldsymbol{s} \to \boldsymbol{s}')}{F(\boldsymbol{s})}. \tag{6}$$

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

$$\sum_{\forall \boldsymbol{s}' : \boldsymbol{s}' \to \boldsymbol{s}} F(\boldsymbol{s}' \to \boldsymbol{s}) = \sum_{\forall \boldsymbol{s}'' : \boldsymbol{s} \to \boldsymbol{s}''} F(\boldsymbol{s} \to \boldsymbol{s}''), \tag{7}$$

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

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 function that can provide consistency condition, Bengio et al. [5] formulates flow-matching loss function as follows:

$$\mathcal{L}_{\text{FM}}(\boldsymbol{s}; \boldsymbol{\theta}) = \left(\log \frac{\sum_{\forall \boldsymbol{s}' : \boldsymbol{s}' \to \boldsymbol{s}} F_{\boldsymbol{\theta}}(\boldsymbol{s}' \to \boldsymbol{s})}{\sum_{\forall \boldsymbol{s}'' : \boldsymbol{s} \to \boldsymbol{s}''} F_{\boldsymbol{\theta}}(\boldsymbol{s} \to \boldsymbol{s}'')} \right)^2. \tag{8}$$

Moreover, as an alternative objective function, Malkin et al. [19] introduces trajectory balance as:

$$\mathcal{L}_{\text{TB}}(\boldsymbol{s}; \boldsymbol{\theta}) = \left(\log \frac{Z_{\boldsymbol{\theta}} \prod_{\boldsymbol{s} \to \boldsymbol{s}'} P_{F_{\boldsymbol{\theta}}}(\boldsymbol{s}'|\boldsymbol{s})}{R(\boldsymbol{x})} \right)^2 \tag{9}$$

where $Z_{\boldsymbol{\theta}}$ is a learnable parameter. The trajectory-balance objective function in equation 9 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 8 and equation 9 to train the GFlowNet model.

B Supplementary Experimental Results and Details

This appendix provides a comprehensive overview of the experimental setup in Section 3 and presents additional supplementary experimental results.

B.1 Datasets and Evaluation Metrics

- **TFbinding:** The dataset is taken from Barrera et al. [4] 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 [25], is curated following the approach outlined by Jain et al. [13]. 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)[10] 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.

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 $\text{PI} = \frac{1}{n_e}\sum_{i=1}^{n_e}(\hat{y}_i - y)$ 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 B.
- **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 $\frac{2}{n_e(n_e-1)}\sum_{i=1}^{n_e-1}\sum_{j=i+1}^{n_e}\text{lev}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j)$.

B.2 Implementation Details

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. [1], Jain et al. [13], 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 [13] for AMP dataset. Furthermore, for CRE dataset we leverage the Malinois model [10] which is a deep convolutional neural network (CNN) for cell type-informed CRE activity prediction of any arbitrary sequence.

In order to implement DE and Ledidi baselines, there should be a proxy model to enable baselines to evaluate their candidate edits. 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

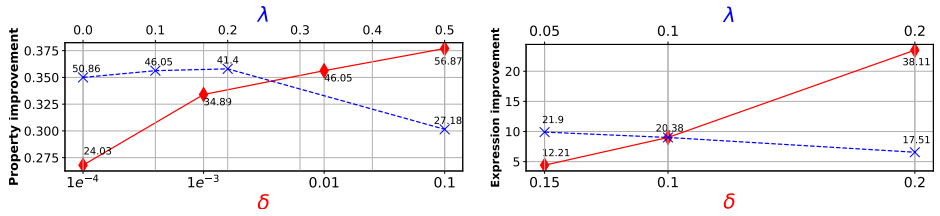

Figure 4: Studying the effect of heyperparameters δ and λ on the performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets. The marker values are edit percentages.

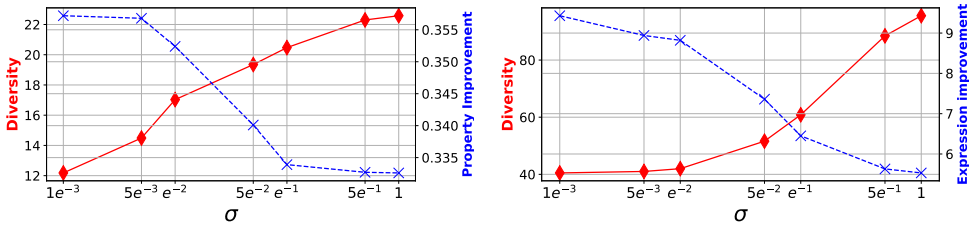

Figure 5: Studying the effect of heyperparameter σ on the diversity and performance of GFNSeqEditor over AMP **(left)** and CRE **(right)** datasets.

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 $2,000$. 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 translator model is then trained to map each low-property sequence to its high-property pair.

We trained an active learning based GFlowNet model following the setting in Jain et al. [13]. 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. 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. [13].

B.3 Supplementary Results

Furthermore, in Figure 4, 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 5, 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.

B.4 Masking

It's worth noting that GFNSeqEditor is capable of performing edits even when certain portions of the input sequence are masked and cannot be modified. Table 2 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 2, GFNSeqEditor achieves significantly greater property improvement than Ledidi while utilizing a lower edit percentage.

Table 2: 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

C Related Works

Generative Flow Networks. GFlowNets, initially proposed by Bengio et al. [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 [20, 14, 36, 33].

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 [9, 22], and graph explainability [17]. Recognizing its significance, several studies have emerged to enhance the learning efficiency of GFlowNets [6, 20, 18, 27] since the introduction of the flow matching learning objective by Bengio et al. [5]. Moreover, GFlowNets have demonstrated adaptability in being jointly trained with energy and reward functions [34]. Pan et al. [23] 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 [24] and rewards [35].

Sequence Generation. The generation of biological sequences (a separate problem from sequence *editing*) has been tackled using a diverse range of methods, including reinforcement learning [1], Bayesian optimization [30], deep generative models for search and sampling [12], generative adversarial networks [37], diffusion models [3], optimization with deep model-based approaches [31], adaptive evolutionary strategies [11, 28], likelihood-free inference [32], and surrogate-based black-box optimization [8], and GFlowNet [13].

Bengio et al. [6] 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.

GFlowNet-AL. Inspired by Bayesian Optimization, Jain et al. [13] proposed a new active learning algorithm based on GFlowNets to design novel biological sequences. GFlowNet-AL [13] 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 sequences.

D 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.