Loss-Guided Auxiliary Agents for Overcoming Mode Collapse in GFlowNets

Idriss Malek[1], **Aya Laajil**[1], **Abhijith Sharma**[1], **Eric Moulines**[1], **Salem Lahlou**[1]

[1]Mohamed Bin Zayed University of Artificial Intelligence, UAE

Abstract

Although Generative Flow Networks (GFlowNets) are designed to capture multiple modes of a reward function, they often suffer from mode collapse in practice, getting trapped in early-discovered modes and requiring prolonged training to find diverse solutions. Existing exploration techniques often rely on heuristic novelty signals. We propose Loss-Guided GFlowNets (LGGFN), a novel approach where an auxiliary GFlowNet's exploration is **directly driven by the main GFlowNet's training loss**. By prioritizing trajectories where the main model exhibits **high loss**, LGGFN focuses sampling on poorly understood regions of the state space. This targeted exploration significantly accelerates the discovery of diverse, high-reward samples. Empirically, across **diverse benchmarks** including grid environments, structured sequence generation, Bayesian structure learning, and biological sequence design, LGGFN consistently **outperforms** baselines in exploration efficiency and sample diversity. For instance, on a challenging sequence generation task, it discovered over 40 times more unique valid modes while simultaneously reducing the exploration error metric by approximately 99% .

Extended version — https://arxiv.org/abs/2505.15251

Introduction

Tasks such as material design and drug discovery are fundamental to advancing essential scientific fields such as medicine (Hughes et al. 2011), energy, engineering and technology (Curtarolo et al. 2013). However, these tasks involve exploring vast combinatorial design spaces, ranging from 10^{60} possible drug-like molecules to potentially more for functional materials, making exhaustive search infeasible. Traditionally, drug and material discovery relied heavily on time-consuming and costly experimental screening and expert-guided design. (Wani et al. 1971; Klayman 1985; Cobb 2010).

To overcome these limitations, machine learning has emerged as a key tool for accelerating discovery by providing proxy models (Schütt et al. 2018; Chen et al. 2019; Jumper et al. 2021) or methods to guide experiments (Häse et al. 2018, 2021). Moreover, it enables efficient exploration of large combinatorial spaces efficiently by leveraging their underlying structure. Reinforcement Learning (RL) (Sutton, Barto et al. 1998) is the most popular learning framework for this purpose, as it enables agents to iteratively interact with an environment to discover the single candidate that maximizes a reward, making it well-suited for tasks where the goal is to identify one optimal solution. However, in a real-world setting, high-scoring candidates may have undesirable side effects or be synthetically inaccessible. Therefore, generating a **diverse set** of high-quality candidates is often more practical than identifying a single optimal solution, increasing the chances that at least one will be viable in practice.

Generative Flow Networks (GFlowNets) (Bengio et al. 2021) aim at learning a forward policy $P_F(a \mid s)$, encoding choices of actions a at partial states s, that sequentially constructs an object x with probability proportional to a given reward $R(x) > 0$. Unlike RL, which prioritizes finding the single best candidate, GFlowNets focus on generating a diverse set of high-reward samples. GFlowNets have found great success in many areas of applications, such as causal discovery (Manta, Hu, and Bengio 2023; Deleu et al. 2022), material discovery (Cipcigan et al. 2024; Nguyen et al. 2023), drug discovery (Shen et al. 2024; Pandey, Subbaraj, and Bengio 2024), biological sequence design (Jain et al. 2022) and editing (Ghari et al. 2024), large language model improvement (Takase et al. 2024; Ho et al. 2024; Hu et al. 2024; Younsi et al. 2025), diffusion models improvement (Zhang et al. 2024), scheduling (Zhang et al. 2023b) and combinatorial optimization problems (Zhang et al. 2023a) . In addition, alternative theories that extend GFlowNets to environments that are continuous (Lahlou et al. 2023a), stochastic (Pan et al. 2023c), and adversarial (Jiralerspong et al. 2024), which further expand the scope of possible applications such as an alternative to denoising diffusion modes (Sendera et al. 2024).

Even if GFlowNets and Reinforcement Learning have two different objectives, they share their need for exploration to achieve their goal. In sparse reward settings, an RL agent that doesn't explore will not be able to improve its "understanding" of the environment and will converge to a sub-optimal policy. In the same manner, even though they were specifically designed to capture multiple modes of a reward function, GFlowNets training strategies that lack exploration might lead the agent to get stuck in a region and make the

underlying distribution incomplete. This behavior is more visible in large sparse-reward environments (Malkin et al. 2023) where the agent might need to go through expansive regions with no or minimal reward.

Previous studies have aimed to enhance GFlowNet training by investigating credit assignment (Malkin et al. 2022; Madan et al. 2023; Pan et al. 2023a). However, akin to RL, the effectiveness of GFlowNets relies on the trajectories used to explore high-reward regions. Due to this similarity, prior work on GFlowNets (Pan et al. 2023b; Rector-Brooks et al. 2023; Lau et al. 2023; Kim et al. 2023; Madan et al. 2025) has drawn insights from the RL literature (Chapelle and Li 2011; Burda et al. 2019; Van Hasselt, Guez, and Silver 2016) to develop new exploration strategies. In particular, Bengio et al. (2023a) briefly cite the idea of using a second GFlowNet trained mostly to match a different reward function that is high when the losses observed by the main GFlowNet are large. Building on this idea, Madan et al. (2025) suggested to implement this idea by using novelty-based intrinsic rewards, whereas Kim et al. (2025) directly uses the loss value of a trajectory.

In this work, we introduce a new general training strategy for GFlowNets that can be used with any training objective. Instead of using an artificial novelty-based intrinsic reward to judge how informative a trajectory is, we directly use the loss value. To avoid ruining the objective probability distribution, we use an auxiliary agent that will sample never-seen trajectories. The Background section introduces GFlowNets, while the related works section in the appendix reviews existing exploration strategies employed in their training. Our methodology section presents the motivation for our approach, provides detailed descriptions of the proposed method, and examines the benefits of utilizing loss signals compared to conventional novelty-based intrinsic rewards. The Experiments section then shows that our method matches or exceeds the previous state-of-the-art across diverse experimental settings, including **hypergrid environments** of varying sizes and sparsity levels, **structured bit sequence generation**, **Bayesian network structure learning**, and **a novel multi-objective mRNA sequence design environment**, achieving improvements ranging from 10% in **Bayesian structure learning** to 99% reduction in exploration error in **sequence generation** where **prior methods failed entirely**.

The method shows **particular strength in sparse reward settings** where traditional on-policy training completely fails, as demonstrated by our motivating example where discovering a distant mode requires navigating through exponentially many low-reward states.

Background

Readers unfamiliar with GFlowNets are encouraged to refer to the first section of the Appendix[1]*, which provides additional context and introduces some key ideas to understand this framework.*

We consider a directed acyclic graph $(\mathcal{S}, \mathcal{A})$, where nodes $\mathcal{S}$ represent states and edges $\mathcal{A}$ represent actions. If $(s, s') \in \mathcal{A}$, we say that s is a parent of s' and that s' is a child of s. There is a unique state s_0 with no parents that we name source state and a unique state s_f with no children that we name sink state. We refer to the parents of the sink state as terminating states $\mathcal{X} = \{s \in \mathcal{S} \mid (s, s_f) \in \mathcal{A})\}$. A complete trajectory $\tau = (s_0, s_1, ..., s_n, s_{n+1} = s_f)$ is a sequence of states that start with the source state and ends with the sink state.

We also consider a reward function $R : \mathcal{X} \to \mathbf{R}$ such as $\forall x \in \mathcal{X}, R(x) > 0$. By convention, we can also consider that the reward for non terminating states is zero $\forall s \in \mathcal{S} \setminus \mathcal{X}, R(s) = 0$. The goal of the GFlowNet is to learn to sample terminating states proportionally to the reward function $P_T(x) \propto R(x)$. To do this, GFlowNets start from the initial state (for example, empty molecule in drug discovery setting, initial coordinates in hyper-grid environment) and iteratively sample actions to move to the following states.

Formally, we consider a non-negative flow function $F : \mathcal{A} \to \mathbf{R}$. The goal of the GFlowNet framework is to learn such a flow function that satisfies the following flow matching (FM) and reward matching constraints, for all $s' \neq s_0, s_f$ and for all $x \in \mathcal{X}$, respectively:

$$\sum_{(s,s')\in\mathcal{A}} F(s \to s') = \sum_{(s',s'')\in\mathcal{A}} F(s' \to s''), \quad (1)$$

$$F(x \to s_f) = R(x). \quad (2)$$

We extend the definition of the flow function to states:

$$\forall s \in \mathcal{S}, \quad F(s) = \sum_{(s,s')\in\mathcal{A}} F(s \to s'). \quad (3)$$

We also define the forward and backward policies as follow:

$$P_F(s' \mid s) = \frac{F(s \to s')}{F(s)}, \;\; P_B(s \mid s') = \frac{F(s \to s')}{F(s')}. \quad (4)$$

In this paper, we use the trajectory balance (TB) objective to train GFlowNets. Introduced in Malkin et al. (2022), the trajectory-decomposable loss only parametrizes the forward P_F^θ and backward P_B^θ policies, and adds a learnable scalar Z_θ that represents the unknown partition function. The loss for each trajectory τ:

$$\mathcal{L}_{TB}(\tau;\theta) = \left(\log\left(\frac{Z_\theta \prod_{t=1}^{n+1} P_F^\theta(s_t|s_{t-1})}{R(s_n)\prod_{t=1}^{n} P_B^\theta(s_{t-1}|s_t)}\right)\right)^2, \quad (5)$$

is minimized by stochastic gradient descent, using trajectories sampled from a given behavior policy. Common choices of the behavior policy include the learned policy P_F^θ and artificially modified versions of it (Bengio et al. 2023b).

Methodology

In this section, we describe our methodology for improving the training efficiency and exploration capabilities of GFlowNets. We begin by illustrating a minimal environment that highlights the limitations of on-policy GFlowNet training in discovering high-reward modes. Building on this insight, **we introduce a novel training framework that leverages an auxiliary GFlowNet guided by the main GFlowNet's loss to promote exploration in underrepresented regions of the state space**. Finally, we discuss the

[1]Appendices are available at: https://arxiv.org/abs/2505.15251.

practical benefits of this loss-guided auxiliary mechanism, particularly in settings where neural network generalization and efficient credit assignment are critical.

Motivation

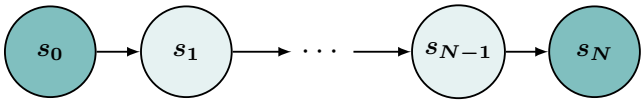

Figure 1: Directed chain of $N+1$ states. Extremal nodes s_0 and s_N have high reward, while all intermediate states have low reward.

To understand why GFlowNets can fail to learn the desired distribution, we consider a simple illustrative environment, depicted in Figure 1. The state space is defined as $\mathcal{S} = \{s_0, s_1, \ldots, s_N\} \cup s_f$, and the action space is $\mathcal{A} = \{\texttt{advance}, \texttt{exit}\}$. For every $i \leq N$, the action `exit` leads from s_i to the terminal state s_f, and for every $i < N$, the action `advance` leads from s_i to s_{i+1}. The reward function assigns high values to the extremal states, with $R(s_0) = R(s_N) \gg R(s_i)$ for all $i \in 1, \ldots, N-1$. We use a tabular GFlowNet where each state s is mapped to a parameter θ_s such that $P_F(\texttt{exit}|s) = \sigma(\theta_s)$ (where σ is the sigmoid function) and a parameter Z that represents the total reward learned.

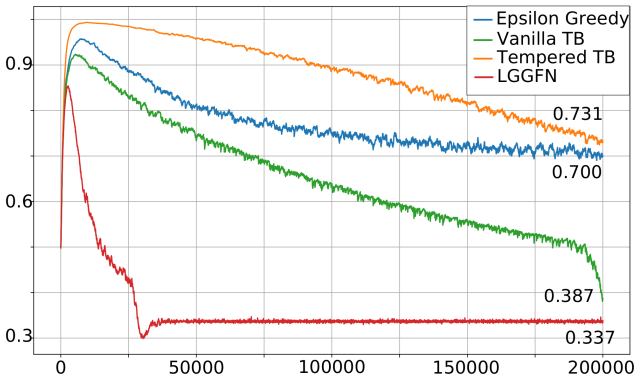

Figure 2: Plot of the evolution of $P_F(\texttt{exit} \mid s_0)$ for different algorithms as a function of number of sampled trajectories. The chain has 100 states and uses the reward setup $R(s_0) = R(s_N) = 101$ and $R(s_i) = 1$. At convergence, $P_F(\texttt{exit} \mid s_0) = \frac{101}{300} \approx 0.337$.

Despite its simplicity (linear, non-branching DAG structure with a single non-terminal action and binary reward pattern), on-policy GFlowNet training fails to converge within a reasonable number of iterations. Intuitively, for the GFlowNet to approximate the correct distribution, it must discover and propagate the high reward associated with s_N. However, assuming the initial policy is a uniform action policy where each action is chosen with probability 0.5, the probability of reaching s_N is 0.5^N. This probability diminishes exponentially with N, and as training progresses, the policy increasingly prioritizes the more easily discovered high-reward state s_0, further reducing the chance of ever reaching s_N. As a result, s_N is typically discovered far later than expected (approximately 190,000 trajectories in the case of a chain of length 100, as indicated by a noticeable inflection point in the green curve in Figure 2), which significantly slows down learning despite the simplicity of the environment.

In more complex settings where rewards vary more widely, a similar phenomenon can occur: trajectories may get trapped in suboptimal regions around locally high-reward states, further impeding exploration and slowing down convergence.

Algorithm

In the example illustrated in Figure 1, once the model has learned to terminate trajectories at s_0, such trajectories become dominant on policy, making the discovery of the final state slower. Figure 2 shows that even after 200,000 trajectories, training using an on-policy method (TB) has yet to converge for a 100-sized chain. Moreover, standard exploration techniques such as tempering or ε-greedy exploration prove insufficient in overcoming this limitation.

To address this, we propose **training the main GFlowNet using trajectories sampled from an auxiliary GFlowNet**, which is optimized to sample trajectories according to a modified reward:

$$R_{\text{aux}} = R_{\text{main}} + \lambda \mathcal{L}_{\text{main}},$$

where R_{main} is the original reward used to train the main GFlowNet, and $\mathcal{L}_{\text{main}}$ denotes the loss associated with a state or trajectory under the main GFlowNet. This auxiliary reward prioritizes trajectories that the main GFlowNet has not yet learned (i.e., those with high loss), while reducing focus on already well-learned regions (i.e., low-loss trajectories). While we mainly focus on trajectory-based objectives in this paper, the same idea holds for transition, state or sub-trajectory -based objective. As a result, this encourages broader exploration and mitigates **mode collapse**.

To leverage the auxiliary GFlowNet, we train the main agent using a mixture of on-policy trajectories and trajectories sampled by the auxiliary GFlowNet. This strategy helps the main agent retain knowledge of previously discovered modes while also exposing it to regions where its performance remains weak. To mitigate the training instability of the auxiliary gflownet that is due to a moving reward, we include R_{main} in the auxiliary reward to provide a consistent structural signal, thereby counteracting the inherent fluctuations of the loss term. The coefficient λ balances the influence of the loss and the reward, ensuring that the overall scale of R_{aux} remains comparable to that of R_{main} and avoids destabilizing the learning process. The training procedure is outlined in detail in Algorithm 1.

Benefits of loss-guided auxiliary GFlowNet

Using the loss of the main GFlowNet as a guide for the auxiliary GFlowNet allows to leverage the generalization induced by the use of Neural Networks. During training, neural networks learn to recognize patterns and generalize them to

Algorithm 1: Training with Auxiliary GFlowNet

Initialize: Main GFlowNet F_{main}, Auxiliary GFlowNet F_{aux}, reward function R, auxiliary weight λ
for each training iteration **do**
  Sample trajectories $\tau_{\text{aux}} \sim F_{\text{aux}}$, with reward R
  Sample trajectories $\tau_{\text{main}} \sim F_{\text{main}}$
  Compute main loss $\mathcal{L}(\tau_{\text{aux}}; F_{\text{main}}, R)$
  Compute auxiliary reward: $R_{\text{aux}} = R + \lambda\mathcal{L}(\tau_{\text{aux}}; F_{\text{main}}, R)$
  Update F_{aux} using $\mathcal{L}(\tau_{\text{aux}}; F_{\text{aux}}, R_{\text{aux}})$
  Concatenate trajectories: $\tau = \tau_{\text{main}} \cup \tau_{\text{aux}}$
  Update F_{main} using $\mathcal{L}(\tau; F_{\text{main}}, R)$
end for

previously unseen states. For instance, as discussed in the BitSequence subsection in the experiments, the model may infer that when a sequence is incomplete but valid, the optimal next action leading to high-reward states is consistently appending a 0 (an open parentheses). Due to this generalization, some trajectories or transitions, despite never being explicitly encountered, can still produce a low loss. Leveraging the loss to guide exploration takes advantage of this property.

In contrast, relying solely on novelty-based intrinsic rewards might drive the model toward areas that are technically new but were already implicitly understood through the network's generalization. Using the loss is also cheaper computationally, since it will be computed anyway to train the main agent, whereas intrinsic rewards ask for a separate set of calculations, and can be expensive depending on which novelty-based algorithm is used.

While similar in spirit, the reward formulation proposed in Kim et al. (2025) differs in two aspects. First, it is asymmetric, placing greater emphasis on trajectories where the estimated forward reward $ZP_F(\tau)$ is lower than the expected reward $R(x_\tau)P_B(\tau \mid x_\tau)$. Second, it combines the primary and auxiliary objectives multiplicatively, using the formulation $\log \mathcal{R}_{\text{aux}} = \alpha \log \mathcal{R} + \log \mathcal{L}$. However, our experiments indicate that this added complexity does not lead to improved performance. In fact, our simplified variant outperforms it by a small margin.

A short theoretical discussion of why loss-guided sampling prevents partial-support stationary points and how it implicitly induces a curriculum over trajectories is provided in the Appendix.

Experiments

In the experiments section, we compare three GFlowNets training procedures with the trajectory balance objective: on-policy (that we simply refer to as TB), Siblings Augmented GFlowNet (SAGFN; Madan et al. 2025), and our loss-guided auxiliary GFlowNet (LGGFN). We first validate the ability of both SAGFN and LGGFN to discover all modes in the hyper-grid environemnt introduced in Bengio et al. (2021), then we show that LGGFN outperforms SAGFN in settings where the rewards have a strong structure: a bit-sequence environment similar to (Malkin et al. 2022)'s, a causal structure learning environment (Deleu et al. 2022), and a structured biological sequence design setting. All experiments are run on 3 different seeds and were done using `torchgfn` (Lahlou et al. 2023b).

Hypergrid

The Hypergrid environment, introduced in (Bengio et al. 2021), is a standard setup in GFlowNet literature. In this environment, states are represented as integer vectors corresponding to points in a grid, and actions involve incrementing one of the coordinates by 1, starting from $s_0 = \mathbf{0}$. The reward function typically assigns higher values to the corners of the grid (for more details on the environment, refer to the Appendix).To showcase that getting stuck in a mode is one of the main reasons GFlowNets fail to explore the state space, we present a thorough experimentation on the Hypergrid environment.

Comparison between LGGFN and adaptive teachers: As discussed in the Methodology section, Kim et al. (2025) propose a related approach. In this section, we present experimental results (Table 1) demonstrating that our simpler method performs on par with their more complex formulation. Therefore, for the remainder of our experiments, we adopt our version of the loss-guided auxiliary GFlowNet.

Method	Grid Size		
	32×32	**64×64**	**128×128**
Adaptive Teachers	7.20 ± 2.00	2.20 ± 0.31	0.92 ± 0.36
LGGFN	$\mathbf{5.33} \pm 1.89$	$\mathbf{2.13} \pm 0.61$	$\mathbf{0.83} \pm 0.21$

Table 1: Final ℓ_1 distance (scaled by $\times 10^{-5}$) on Hypergrid benchmarks after 10^4 sampled trajectories.

Additionally, we observed that the performance and stability of (Kim et al. 2025) were highly dependent on the selection of hyperparameters. Specifically, deviations from the hyperparameter values specified in the original work frequently led to numerical instability and NaN values, particularly for larger grid sizes (128 x 128). In contrast, our version exhibits a significantly lower sensitivity to hyperparameter variations, as demonstrated later in the Experiments.

Different sizes: As shown by the toy example, the size of the environment exponentially impacts the ability of the model to escape explored modes. We validate this observation with a much more complete experimentation (Table 2). In the following experiments, we use a sparse reward parameter that makes the exploration of modes far from s_0 harder ($R_0 = 0.0001$, see the Appendix for more details).

We observe that on-policy training quickly becomes ineffective as the size of the hypergrid increases, as it struggles to escape the first mode within a reasonable amount of time (see the appendix for visualizations of the learnt distributions). In contrast, both SAGFN and LGGFN perform similarly on this task: they consistently discover all the modes and achieve nearly identical L1 losses across all

experiments. For these two algorithms, we can distinguish two distinct training phases in Figure 3. The first phase is characterized by a steep decline in the loss curve, indicating active exploration and significant influence from the auxiliary GFlowNet's loss or intrinsic reward. The second phase begins when the curve starts to plateau, with only slow, incremental improvement: at this point, the loss or intrinsic reward has diminished to the extent that switching to on-policy training would yield comparable results. It is also worth noting that LGGFN shows more consistent training behavior during the first phase across experiments with different random seeds, as evidenced by a smaller standard deviation (reflected by the narrower standard deviation band in the plot) compared to SAGFN.

	TB	SAGFN	LGGFN
Size	*2-Dimensional Grid*		
32	1460.00 ± 0.016	63.9 ± 12.4	$\mathbf{53.3 \pm 18.9}$
64	366.00 ± 0.010	24.8 ± 5.64	$\mathbf{21.3 \pm 6.14}$
80	234.00 ± 0.0039	17.5 ± 4.51	$\mathbf{15.7 \pm 3.44}$
96	163.00 ± 0.0070	11.4 ± 3.09	$\mathbf{10.6 \pm 1.95}$
128	91.6 ± 0.0008	$\mathbf{7.07 \pm 1.00}$	8.30 ± 2.10
144	72.3 ± 0.0009	2.71 ± 0.493	$\mathbf{2.57 \pm 0.373}$
	4-Dimensional Grid		
4	113.0 ± 39.9	169.0 ± 47.7	$\mathbf{113.0 \pm 58.9}$
8	417.0 ± 38.2	27.1 ± 5.04	$\mathbf{21.2 \pm 3.19}$
16	28.6 ± 0.000099	2.15 ± 0.314	$\mathbf{2.13 \pm 0.309}$

Table 2: Final ℓ_1 distance (scaled by $\times 10^{-6}$) for various Hypergrid sizes after 10^4 sampled trajectories. Lower is better.

Different sparsity: In the toy example, the main reason the GFlowNet couldn't access the last high reward state is the low reward of the intermediate states, that led to a low move-on probability in the initial state. To validate this hypothesis, we evaluate a range of reward configurations with varying levels of sparsity by systematically modifying the value of R_0 (Figure 4).

We observe that, even in relatively easy settings, on-policy training results in delayed convergence. Furthermore, once all modes have been discovered, the behavior of all three algorithms converges, suggesting that the auxiliary agent can be discarded at this stage. This allows continued on-policy training without performance degradation, while reducing computational overhead.

Influence of the coefficient λ: The auxiliary reward incorporates a coefficient λ to balance the original reward R and the additional loss term $\mathcal{L}$. However, experimental results indicate that when $\mathcal{L}$ is initially scaled to be in the same range as R, variations in λ within a reasonable range (i.e., not approaching zero) have negligible effect on performance, as demonstrated in Figure 5.

Valid bit sequences

The BITSEQUENCES environment was introduced as a testbed for studying structured sequence generation by Malkin et al. (2022). Each state in the environment corresponds to a binary sequence, and the agent's actions consist of appending either individual bits or blocks of bits to the current sequence. Episodes start with an empty sequence $s_0 = \epsilon$ and terminate when the sequence reaches a predefined length. In the original formulation, the reward function quantifies how close a given sequence is to the set of modes $R(s) = e^{-d(s,\text{modes})}$. This set of modes is constructed by first arbitrarily selecting a set of words H and then generating different complete sequences by randomly combining these words. Malkin et al. (2022) argues that this construction inherently imposes a structural property on the reward function. However, this claim is not entirely accurate, as the candidate sequences that were not ultimately selected as modes still share the same underlying structure but may receive a significantly lower reward. For example, if we work on sequences of size 6 and the set of arbitrary words is $H = \{01, 10\}$ and the set of modes is $\{010101, 101010, 100110\}$. In this setting, the sequence 011001 shares the same "structure" of the modes as it is built from the same process but has a much lower reward e^{-2} (the reward of the modes being 1).
We introduce a new reward structure for this environment (the motivation behind this modification is given in the Appendix). The set of valid sequences is defined recursively as the smallest set that contains the empty sequence, is closed under concatenation, and is stable under the simultaneous operation of prepending a 0 and appending a 1. Interpreting 0 as an opening parenthesis `(` and 1 as a closing parenthesis `)`, a valid sequence corresponds to a balanced parentheses sequence. When considering sequences of maximum length $2N$, we define a sequence as *complete* if it attains this maximum length. The objective of the task is to train a GFlowNet to discover the full set of complete valid sequences.

To evaluate the performance of different algorithms in this environment, we sample 16,000 sequences and assess them using two key metrics:

- **Exploration ↓:** We estimate the probability mass assigned to the set of complete valid sequences using Monte Carlo sampling and report its difference to the true probability mass.
- **Diversity ↑:** We measure the fraction of distinct complete valid sequences sampled. For sequences of length $2N$, the number of possible valid bit sequences is given by the N-th catalan number $C_N = \frac{1}{N+1}\binom{2N}{N} = \frac{(2N)!}{(N+1)!N!}$.

The results are presented in Table 3. Overall, LGGFN significantly outperforms the other algorithms, particularly on longer sequences. Its advantage over SAGFN is likely attributable to its ability to leverage generalization, as hypothesized earlier. Architectural details and hyperaparameters are provided in the Appendix.

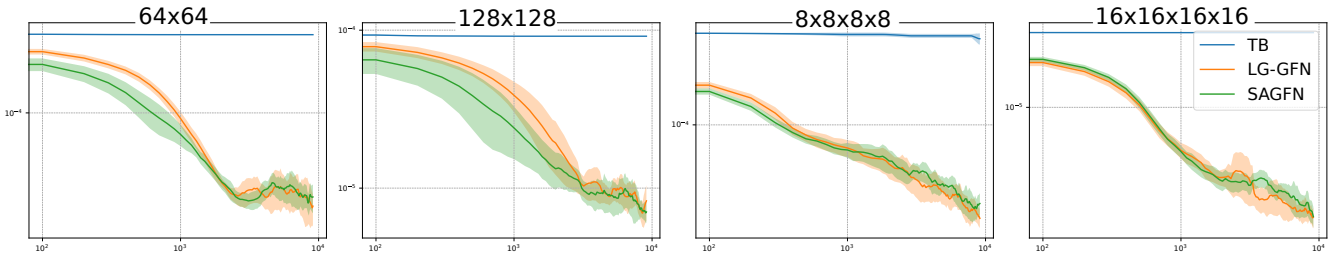

Figure 3: L1-loss during training for different sizes of hypergrid as a function of training iterations.

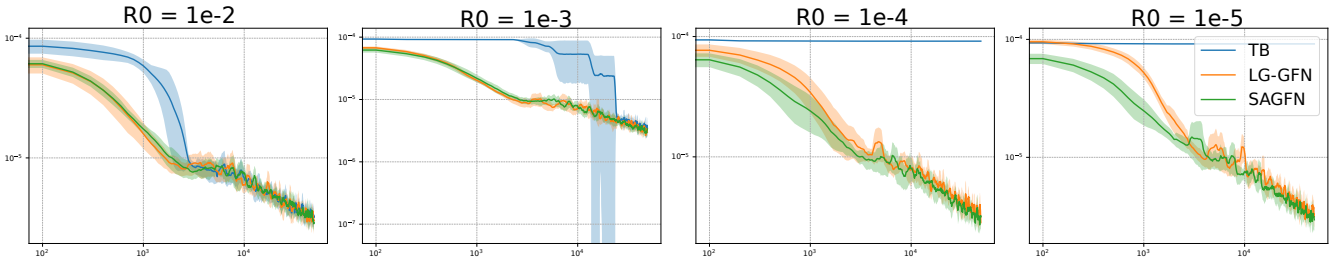

Figure 4: L1-loss during training for different values of R_0 and a fixed size of 128x128. as a function of training iterations.

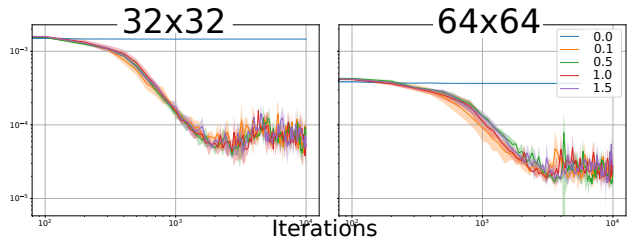

Figure 5: L1-loss evolution over training iterations for different λ values and grid sizes.

Bayesian structure learning

Bayesian networks (Pearl 2014) are probabilistic graphical models that represent a set of variables and their conditional dependencies using a directed acyclic graph (DAG). When this graph is not known *a priori* (e.g., from expert knowledge), it can be inferred from a dataset of observations $\mathcal{D}$.

Traditional algorithms for this task typically return a single DAG (Chickering 2002), which is suboptimal in practice. One reason for this limitation is that such methods fail to account for the inherent uncertainty arising from Markov equivalence (explained in the Appendix).

Graphs within the same equivalence class are therefore indistinguishable solely on observational data. Another important consideration is that, when the dataset $\mathcal{D}$ is limited, it may not provide sufficient evidence to clearly favor a particular equivalence class. In such cases, predicting a single graph (or a single equivalence class) may lead to poor calibration. Hence, it is more appropriate to estimate a posterior distribution over DAGs, $P(G \mid \mathcal{D})$, which better captures the range of plausible graph structures supported by the data.

The use of GFlowNets for Bayesian structure learning was previously explored by Deleu et al. (2022), yielding strong empirical results. Building on this work, we evaluate the effectiveness of various algorithms in the context of Bayesian structure learning. Details of the environment, architectures, and hyperparameters, are in the appendix.

We evaluate the models using two metrics:

- **Expected Structural Hamming Distance** ($\mathbb{E}$-SHD) $\downarrow$: the number of missing or extra edges, and the number of reversals required to transform one graph into another.
- **ROC-AUC** $\uparrow$: measures how well the predicted edge probabilities distinguish real edges from non-existent edges.

Experimental results are presented in Table 4. For graphs with a small number of nodes (5 to 8), all three algorithms exhibit competitive performance, with no single method consistently outperforming the others. However, LGGFN demonstrates a clear advantage as the graph size increases (9 nodes and above). Given that the size of the state space grows rapidly with the number of nodes (see the Appendix), this performance gap is likely due to LGGFN's ability to leverage generalization effectively. By learning to identify and avoid unpromising regions of the search space, LGGFN reduces the need for exhaustive exploration, directly leading to better results.

Structured Biological Sequence Generation

Designing messenger RNA (mRNA) sequences that efficiently and stably express target proteins is a central challenge in synthetic biology and medicine (Zhang et al. 2023c; Fallahpour et al. 2025). Due to the redundancy of the genetic code, each protein can be encoded by an exponentially large number of synonymous mRNA sequences, which makes

Method	16		24		32		40		48	
	Diversity	**Exploration**	**Diversity**	**Exploration**	**Diversity**	**Exploration**	**Diversity**	**Exploration**	**Diversity**	**Exploration**
TB	23 ± 3	0.92 ± 0.00	**15309** ± 40	**0.01** ± 0.00	23 ± 3	0.92 ± 0.00	89 ± 7	0.99 ± 0.00	54 ± 5	0.99 ± 0.00
SAGFN	473 ± 640	0.63 ± 0.41	7387 ± 95	0.05 ± 0.01	4392 ± 3129	0.45 ± 0.39	102 ± 13	0.98 ± 0.01	35 ± 8	0.99 ± 0.01
LGGFN	**1425** ± 2	**0.01** ± 0.01	7770 ± 10	0.01 ± 0.01	**7805** ± 39	**0.02** ± 0.01	**7746** ± 53	**0.02** ± 0.01	**7553** ± 65	**0.05** ± 0.01

Table 3: Comparison of on-policy (TB), SAGFN, and LGGFN across different sequence sizes.

Method	Graph Size (Nodes)								
	5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
					ROC AUC				
TB	**0.72** ± 0.08	0.57 ± 0.15	0.58 ± 0.08	**0.60** ± 0.06	0.53 ± 0.07	0.53 ± 0.13	0.56 ± 0.09	0.52 ± 0.08	0.56 ± 0.05
SAGFN	0.66 ± 0.15	**0.68** ± 0.10	0.44 ± 0.07	0.51 ± 0.16	**0.56** ± 0.13	0.59 ± 0.06	0.49 ± 0.03	0.52 ± 0.08	0.55 ± 0.04
LGGFN	0.57 ± 0.18	**0.68** ± 0.05	**0.62** ± 0.16	0.52 ± 0.08	**0.57** ± 0.05	**0.62** ± 0.07	**0.63** ± 0.11	**0.59** ± 0.03	**0.57** ± 0.07
					$\mathbb{E}$*-SHD*				
TB	7.33 ± 1.08	12.28 ± 1.06	16.69 ± 0.12	**20.25** ± 2.62	29.52 ± 2.11	37.31 ± 3.97	42.60 ± 3.38	54.09 ± 0.41	64.20 ± 3.05
SAGFN	7.62 ± 0.23	**11.13** ± 0.85	18.03 ± 0.49	23.19 ± 2.20	**29.34** ± 2.60	**36.53** ± 1.12	45.58 ± 0.76	52.88 ± 4.78	63.31 ± 3.77
LGGFN	**7.12** ± 1.38	11.83 ± 0.37	**15.81** ± 2.47	23.78 ± 1.36	**29.11** ± 1.17	**36.68** ± 0.63	**41.33** ± 3.50	**51.77** ± 3.50	**63.00** ± 4.50

Table 4: Comparison of on-policy (TB), SAGFN, and LGGFN on ROC AUC and SHD metrics across different graph sizes.

this task well suited to GFlowNet-based generative modeling (Jain et al. 2022; Cretu et al. 2024). However, not all synonymous sequences are equally effective biologically. We define a multi-objective reward function (Jain et al. 2023) that captures desirable biological properties. We create a generative framework that learns to generate synonymous codon sequences while adhering to user-customized biological constraints. The objective involves the Codon Adaptation Index (CAI), the GC content, and the Minimum Free energy (MFE), similar to Laajil et al. (2025), all explained in the Appendix.

Results: We evaluated three GFlowNet variants, TB, LGGFN, and SAGFN, for designing mRNA sequences encoding the protein, *Acyl-CoA thioesterase 13* (ACOT13), a 151-amino acid enzyme involved in lipid metabolism. LGGFN demonstrated superior multi-objective mRNA design by generating sequences with higher GC content (up to 55.11% vs. 45.33% natural), more stable RNA secondary structures (lowest MFE of -91.30 kcal/mol vs. -68.20 kcal/mol natural), and improved codon adaptation index (CAI up to 0.62 vs. 0.54 natural). The generated sequences also exhibited substantial diversity, as measured by Levenshtein distance between the generated sequences averaging around 50–60 from each other.

Conclusion

In this work, we introduced Loss-Guided GFlowNets (LGGFN), a simple yet effective auxiliary training strategy for GFlowNets. Through extensive experiments across four distinct domains, we demonstrated that LGGFN significantly improves exploration efficiency and sample diversity,

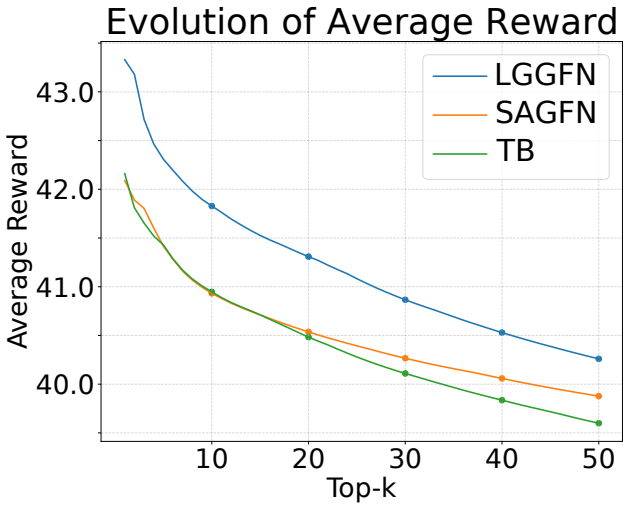

Figure 6: LGGFN consistently outperforms other methods and provides a higher average reward.

with improvements ranging from 10% in complex tasks to over 99% in sparse reward settings. While LGGFN shows strong empirical results, computing the auxiliary reward requires evaluating the main model's loss, which adds computational overhead during training. Additionally, the method's performance depends on the choice of loss function and the balance coefficient λ, though we found it to be relatively robust to these choices in practice. Future work could explore using alternative signals as external rewards.

References

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

Bengio, Y.; Lahlou, S.; Deleu, T.; Hu, E. J.; Tiwari, M.; and Bengio, E. 2023a. GFlowNet Foundations. *Journal of Machine Learning Research*, 24(210): 1–55.

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

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

Chapelle, O.; and Li, L. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24.

Chen, C.; Ye, W.; Zuo, Y.; Zheng, C.; and Ong, S. P. 2019. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9): 3564–3572.

Chickering, D. M. 2002. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov): 507–554.

Cipcigan, F.; Booth, J.; Barros Ferreira, R. N.; Ribeiro dos Santos, C.; and Steiner, M. 2024. Discovery of novel reticular materials for carbon dioxide capture using GFlowNets. *Digital Discovery*, 3: 449–455.

Cobb, H. M. 2010. *The History of Stainless Steel*. Materials Park, OH: ASM International. ISBN 9781615030530.

Cretu, M.; Harris, C.; Igashov, I.; Schneuing, A.; Segler, M.; Correia, B.; others; and Liò, P. 2024. SynFlowNet: Design of diverse and novel molecules with synthesis constraints. *arXiv preprint arXiv:2405.01155*.

Curtarolo, S.; Hart, G. L. W.; Nardelli, M. B.; Mingo, N.; Sanvito, S.; and Levy, O. 2013. The high-throughput highway to computational materials design. *Nature Materials*, 12(3): 191–201.

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

Fallahpour, A.; Gureghian, V.; Filion, G. J.; Lindner, A. B.; and Pandi, A. 2025. CodonTransformer: a multispecies codon optimizer using context-aware neural networks. *Nature Communications*, 16(1): 3205.

Ghari, P. M.; Tseng, A. M.; Eraslan, G.; Lopez, R.; Biancalani, T.; Scalia, G.; and Hajiramezanali, E. 2024. GFlowNet Assisted Biological Sequence Editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Häse, F.; Aldeghi, M.; Hickman, R. J.; Roch, L. M.; and Aspuru-Guzik, A. 2021. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3): 031406.

Häse, F.; Roch, L. M.; Kreisbeck, C.; and Aspuru-Guzik, A. 2018. Phoenics: A Bayesian optimizer for chemistry. *ACS Central Science*, 4(9): 1134–1145.

Ho, M.; Zhu, V.; Chen, X.; Jain, M.; Malkin, N.; and Zhang, E. 2024. Proof Flow: Preliminary Study on Generative Flow Network Language Model Tuning for Formal Reasoning. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

Hu, E. J.; Jain, M.; Elmoznino, E.; Kaddar, Y.; Lajoie, G.; Bengio, Y.; and Malkin, N. 2024. Amortizing intractable inference in large language models. In *The Twelfth International Conference on Learning Representations*.

Hughes, J. P.; Rees, S.; Kalindjian, S. B.; and Philpott, K. L. 2011. Principles of early drug discovery. *British Journal of Pharmacology*, 162(6): 1239–1249.

Jain, M.; Bengio, E.; Hernandez-Garcia, A.; Rector-Brooks, J.; Dossou, B. F.; Ekbote, C. A.; Fu, J.; Zhang, T.; Kilgour, M.; Zhang, D.; et al. 2022. Biological sequence design with gflownets. In *International Conference on Machine Learning*, 9786–9801. PMLR.

Jain, M.; Raparthy, S. C.; Hernández-García, A.; Rector-Brooks, J.; Bengio, Y.; Miret, S.; and Bengio, E. 2023. Multi-objective GFlowNets. In *International Conference on Machine Learning*, 14631–14653. PMLR.

Jiralerspong, M.; Sun, B.; Vucetic, D.; Zhang, T.; Bengio, Y.; Gidel, G.; and Malkin, N. 2024. Expected flow networks in stochastic environments and two-player zero-sum games. In *The Twelfth International Conference on Learning Representations*.

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T.; Petersen, S.; Reiman, D.; Clancy, E.; Zielinski, M.; Steinegger, M.; Pacholska, M.; Berghammer, T.; Bodenstein, S.; Silver, D.; Vinyals, O.; Senior, A. W.; Kavukcuoglu, K.; Kohli, P.; and Hassabis, D. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589.

Kim, M.; Choi, S.; Yun, T.; Bengio, E.; Feng, L.; Rector-Brooks, J.; Ahn, S.; Park, J.; Malkin, N.; and Bengio, Y. 2025. Adaptive teachers for amortized samplers. In *The Thirteenth International Conference on Learning Representations*.

Kim, M.; Yun, T.; Bengio, E.; Zhang, D.; Bengio, Y.; Ahn, S.; and Park, J. 2023. Local search gflownets. *arXiv preprint arXiv:2310.02710*.

Klayman, D. L. 1985. Qinghaosu (artemisinin): an antimalarial drug from China. *Science*, 228(4703): 1049–1055.

Laajil, A.; Shtanchaev, A.; Muhammad, S.; Moulines, E.; and Lahlou, S. 2025. Curriculum-Augmented GFlowNets For mRNA Sequence Generation. *arXiv preprint arXiv:2510.03811*.

Lahlou, S.; Deleu, T.; Lemos, P.; Zhang, D.; Volokhova, A.; Hernández-Garcıa, A.; Ezzine, L. N.; Bengio, Y.; and Malkin, N. 2023a. A theory of continuous generative flow

networks. In *International Conference on Machine Learning*, 18269–18300. PMLR.

Lahlou, S.; Viviano, J. D.; Schmidt, V.; and Bengio, Y. 2023b. torchgfn: A pytorch gflownet library. *arXiv preprint arXiv:2305.14594*.

Lau, E.; Vemgal, N.; Precup, D.; and Bengio, E. 2023. DGFN: Double generative flow networks. *arXiv preprint arXiv:2310.19685*.

Madan, K.; Lamb, A.; Bengio, E.; Berseth, G.; and Bengio, Y. 2025. Towards Improving Exploration through Sibling Augmented GFlowNets. In *The Thirteenth International Conference on Learning Representations*.

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

Malkin, N.; Jain, M.; Bengio, E.; Sun, C.; and Bengio, Y. 2022. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967.

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

Manta, D. C.; Hu, E. J.; and Bengio, Y. 2023. GFlowNets for Causal Discovery: an Overview. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*.

Nguyen, T. M.; Tawfik, S. A.; Tran, T.; Gupta, S.; Rana, S.; and Venkatesh, S. 2023. Hierarchical GFlownet for Crystal Structure Generation. In *AI for Accelerated Materials Design-NeurIPS 2023 Workshop*.

Pan, L.; Malkin, N.; Zhang, D.; and Bengio, Y. 2023a. Better training of gflownets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, 26878–26890. PMLR.

Pan, L.; Zhang, D.; Courville, A. C.; Huang, L.; and Bengio, Y. 2023b. Generative Augmented Flow Networks. In *ICLR*.

Pan, L.; Zhang, D.; Jain, M.; Huang, L.; and Bengio, Y. 2023c. Stochastic Generative Flow Networks. In *The 39th Conference on Uncertainty in Artificial Intelligence*.

Pandey, M.; Subbaraj, G.; and Bengio, E. 2024. GFlowNet Pretraining with Inexpensive Rewards. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*.

Pearl, J. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Rector-Brooks, J.; Madan, K.; Jain, M.; Korablyov, M.; Liu, C.-H.; Chandar, S.; Malkin, N.; and Bengio, Y. 2023. Thompson sampling for improved exploration in gflownets. *arXiv preprint arXiv:2306.17693*.

Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; and Müller, K.-R. 2018. SchNet–A deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24): 241722.

Sendera, M.; Kim, M.; Mittal, S.; Lemos, P.; Scimeca, L.; Rector-Brooks, J.; Adam, A.; Bengio, Y.; and Malkin, N. 2024. Improved off-policy training of diffusion samplers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shen, T.; Seo, S.; Lee, G.; Pandey, M.; Smith, J. R.; Cherkasov, A.; Kim, W. Y.; and Ester, M. 2024. TacoGFN: Target-conditioned GFlowNet for Structure-based Drug Design. *Transactions on Machine Learning Research*.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Takase, R.; Tsunokake, M.; Tsuchiya, Y.; and Inuzuka, S. 2024. GFlowNet Fine-tuning for Diverse Correct Solutions in Mathematical Reasoning Tasks. *arXiv preprint arXiv:2410.20147*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wani, M. C.; Taylor, H. L.; Wall, M. E.; Coggon, P.; and McPhail, A. T. 1971. Plant antitumor agents. VI. Isolation and structure of taxol, a novel antileukemic and antitumor agent from *Taxus brevifolia*. *Journal of the American Chemical Society*, 93(9): 2325–2327.

Younsi, A.; Abubaker, A.; Seddik, M. E. A.; Hacid, H.; and Lahlou, S. 2025. Accurate and diverse LLM mathematical reasoning via automated PRM-guided GFlowNets. *arXiv preprint arXiv:2504.19981*.

Zhang, D.; Dai, H.; Malkin, N.; Courville, A.; Bengio, Y.; and Pan, L. 2023a. Let the Flows Tell: Solving Graph Combinatorial Problems with GFlowNets. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhang, D.; Zhang, Y.; Gu, J.; Zhang, R.; Susskind, J. M.; Jaitly, N.; and Zhai, S. 2024. Improving GFlowNets for Text-to-Image Diffusion Alignment. *CoRR*, abs/2406.00633.

Zhang, D. W.; Rainone, C.; Peschl, M.; and Bondesan, R. 2023b. Robust Scheduling with GFlowNets. In *The Eleventh International Conference on Learning Representations*.

Zhang, H.; Zhang, L.; Lin, A.; Xu, C.; Li, Z.; Liu, K.; and Huang, L. 2023c. Algorithm for optimized mRNA design improves stability and immunogenicity. *Nature*, 621(7978): 396–403.