Stochastic Generative Flow Networks

Ling Pan[*1,2] **Dinghuai Zhang**[*1,2] **Moksh Jain**[1,2] **Longbo Huang**[3] **Yoshua Bengio**[1,2,4]

[1]Mila - Québec AI Institute
[2]Université de Montréal
[3]Tsinghua University
[4]CIFAR AI Chair

Abstract

Generative Flow Networks (or GFlowNets for short) are a family of probabilistic agents that learn to sample complex combinatorial structures through the lens of "inference as control". They have shown great potential in generating high-quality and diverse candidates from a given energy landscape. However, existing GFlowNets can be applied only to deterministic environments, and fail in more general tasks with stochastic dynamics, which can limit their applicability. To overcome this challenge, this paper introduces Stochastic GFlowNets, a new algorithm that extends GFlowNets to stochastic environments. By decomposing state transitions into two steps, Stochastic GFlowNets isolate environmental stochasticity and learn a dynamics model to capture it. Extensive experimental results demonstrate that Stochastic GFlowNets offer significant advantages over standard GFlowNets as well as MCMC- and RL-based approaches, on a variety of standard benchmarks with stochastic dynamics.

1 INTRODUCTION

Recently, Generative Flow Networks [GFlowNets; Bengio et al., 2021a,b] have been successfully applied to a wide variety of tasks, including molecule discovery [Bengio et al., 2021a, Jain et al., 2022b], biological sequence design [Jain et al., 2022a], and robust scheduling [Zhang et al., 2023a]. GFlowNets learn policies to generate objects $x \in \mathcal{X}$ sequentially, and are related to Monte-Carlo Markov chain (MCMC) methods [Metropolis et al., 1953, Hastings, 1970, Andrieu et al., 2003], generative models [Goodfellow et al., 2014, Ho et al., 2020], and amortized variational inference [Kingma and Welling, 2013]. The sequential process of generating an object following a policy bears a close resemblance to reinforcement learning [RL; Sutton and Barto, 2018]. Contrary to the typical reward-maximizing policy in RL [Mnih et al., 2015, Lillicrap et al., 2015, Haarnoja et al., 2017, Fujimoto et al., 2018, Haarnoja et al., 2018], GFlowNets aim to learn a stochastic policy for sampling composite objects x with probability *proportional to the reward function* $R(x)$. This is desirable in many real-world tasks where the diversity of solutions is important, and we aim to sample a diverse set of high-reward candidates, including recommender systems [Kunaver and Požrl, 2017], drug discovery [Bengio et al., 2021a, Jain et al., 2022a], and sampling causal models from a Bayesian posterior [Deleu et al., 2022], among others.

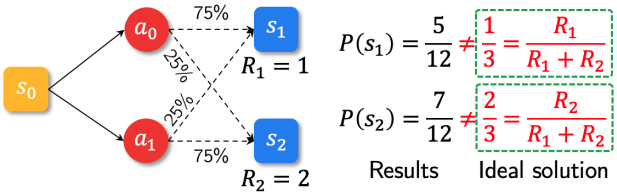


Figure 1: An example illustrating the failure of existing GFlowNet approaches. (Left) Squares and circles denote states and actions, while solid and dotted arrows correspond to policy decisions and stochastic environment dynamics. The numbers above the dotted lines represent state transition probabilities, and the numbers below the blue squares (terminal states) denote the terminal reward. (Right) Results from existing GFlowNet approaches and the ideal solution.

Existing work on GFlowNets [Bengio et al., 2021a, Malkin et al., 2022a, Madan et al., 2022], however, is limited to deterministic environments, where state transitions are deterministic, which may limit their applicability in the more general stochastic cases in practice. Figure 1 illustrates an example with stochastic transition dynamics where existing GFlowNet approaches can fail. Standard GFlowNet approaches will result in $P(s_1) = \frac{5}{12}$ and $P(s_2) = \frac{7}{12}$ when trained to completion (with $P(s)$ denoting the probability

[*]Equal contribution.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

of sampling state s), which does not match the ideal case where $P(s_1) = \frac{1}{3}$ and $P(s_2) = \frac{2}{3}$. Therefore, the learned policy does not sample proportionally to the reward function in the presence of stochastic transition dynamics. In practice, many tasks involve stochasticity in state transitions, which are more challenging to solve but are applicable to a wide variety of problems [Antonoglou et al., 2021, Yang et al., 2022, Paster et al., 2022].

To address this limitation, in this paper, we introduce a novel methodology, Stochastic GFlowNet, which is the first empirically effective approach for tackling environments with stochastic transition dynamics with GFlowNets. Stochastic GFlowNet decomposes the state transitions based on the concept of *afterstates* [Sutton and Barto, 2018, Bengio et al., 2021b]. Specifically, each stochastic state transition is decomposed into a deterministic step that transitions from the environment state s to an intermediate state (s, a) and a stochastic step that branches from the intermediate state (s, a) to the next state s'. We propose a practical way for training the dynamics model to capture the stochastic environment dynamics. The methodology is general and can be applied to different GFlowNet learning objectives. The code is publicly available at `https://github.com/ling-pan/Stochastic-GFN`.

In summary, the contribution of this work is as follows:

- We propose a novel method, Stochastic GFlowNet, which is the first empirically effective approach extending GFlowNets to the more general stochastic environments based on Bengio et al. [2021b].
- We conduct extensive experiments on GFlowNet benchmark tasks augmented with stochastic transition dynamics, and validate the effectiveness of our approach in tackling stochastic environments. Results show that our method significantly outperforms existing baselines and scales well to the more complex and challenging biological sequence generation tasks.

2 BACKGROUND

2.1 GFLOWNET PRELIMINARIES

We denote a directed acyclic graph (DAG) by $\mathcal{G} = (\mathcal{S}, \mathcal{A})$, with $\mathcal{S}$ the set of vertices corresponding to the states and $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ the set of edges, which corresponds to the set of actions. There is a unique *initial state* $s_0 \in \mathcal{S}$ which has no parent state; on the other hand, we define all states without children to be *terminal states*, whose set is denoted by $\mathcal{X} \subseteq \mathcal{S}$. A GFlowNet learns stochastic policy which aims to sample complete trajectories $\tau = (s_0 \to s_1 \to \cdots \to s_n)$ where $s_n \in \mathcal{X}$ and $(s_i \to s_{i+1}) \in \mathcal{A}, \forall i$ to sample terminal states. Each trajectory is assigned a non-negative *flow* $F(\tau)$. A trajectory can be generated sequentially by sampling iteratively from the *forward policy* $P_F(s_{t+1}|s_t)$, which is a collection of distributions over the children at each state. Existing work on GFlowNets assumes a one-to-one correspondence between action and next state, making the definition of forward policy to be consistent to the notion of policy in general RL. Nonetheless, in this work, we relax this assumption and generalize GFlowNets to more flexible stochastic environments. The objective of GFlowNet learning is to sample terminal states with probability proportional to a given non-negative reward function $R(x)$ for all $x \in \mathcal{X}$. This indicates that all the flows that end up with x should sum up to $R(x)$, namely $\sum_{\tau \to x} F(\tau) = R(x), \forall x \in \mathcal{X}$, where $\tau \to x$ is a trajectory τ that ends in x and the sum is thus over all complete trajectories that lead to terminal state $x \in \mathcal{X}$. To formalize this, we first define the *terminating probability* $P_T(x)$ to be the marginal likelihood of sampling trajectories terminating at a terminal state x:

$$P_T(x) = \sum_{\tau \to x} P_F(\tau) = \sum_{\tau \to x} \prod_{i=1}^{n} P_F(s_i|s_{i-1}). \quad (1)$$

Therefore, the goal of GFlowNet learning is to obtain a policy such that $P_T(x) \propto R(x), \forall x \in \mathcal{X}$.

2.2 LEARNING OBJECTIVES FOR GFLOWNETS

In applicative tasks, practitioners need to design the GFlowNet modules (*e.g.*, policies, flows) with parameterized neural networks, and further choose a training criterion to train these networks. In this subsection, we briefly summarize some learning criteria of GFlowNets.

Detailed balance (DB). By summing the flows $F(\tau)$ of all the trajectories τ going through a state s, we can define a state flow $F(s) := \sum_{\tau \ni s} F(\tau)$. Such a function can be learned, together with forward and backward policy $P_F(s'|s), P_B(s|s')$, where s' is a child state of s. The backward policy P_B, a collection of distributions over the parents of each state, is not part of the generative process, but serves as a tool for learning the forward policy P_F. The GFlowNet detailed balance (DB) constraint is defined as

$$F(s)P_F(s'|s) = F(s')P_B(s|s'), \ \forall (s \to s') \in \mathcal{A}. \quad (2)$$

It is also worth noting that at terminal states x, it pushes the flow at x to match the terminal rewards $R(x)$. In practice, we transform the DB constraint into a training objective by setting the loss function to be a squared difference between the logarithm of the left and right-hand sides Bengio et al. [2021a] of Eq. (2). If the optimization objective is perfectly minimized, it would make the above flow consistency constraint satisfied, thus making the forward policy P_F sample proportionally to given reward values, as desired. It means that after training, the constraint is only approximately achieved (and in general it would be intractable to obtain an exact solution).

Trajectory balance (TB). In analogy to the forward decomposition of a complete trajectory in Eq. (1), we could use $\prod_{i=1}^{n} P_B(s_{i-1}|s_i)$ to represent the trajectory backward probability. As an alternative to DB, Malkin et al. [2022a] propose the trajectory balance (TB) criterion which operates on complete trajctories, instead of state transitions, defined as follows

$$Z \prod_{i=1}^{n} P_F(s_i|s_{i-1}) = R(x) \prod_{i=1}^{n} P_B(s_{i-1}|s_i), \qquad (3)$$

where $\tau = (s_0 \to s_1 \to \ldots \to s_n = x)$ is any complete trajectory and Z is a learned scalar parameter, denoting the partition function of the reward distribution. Note that TB does not explicitly learn a flow function.

For on-policy training, we can simply use trajectories sampled from the forward policy P_F to evaluate the training loss and its gradient with respect to the parameters of the neural networks. The GFlowNet training objectives can be further improved with off-policy training, *i.e.*, with trajectories sampled from a broader and more exploratory distribution than P_F [Malkin et al., 2022b]. A popular choice is using a tempered version of the current forward policy [Zhang et al., 2022b] or a mixture of the forward policy and a uniform random policy [Bengio et al., 2021a] that mimics ϵ-greedy exploration in RL.

3 STOCHASTIC GFLOWNETS

We now describe the Stochastic GFlowNet, a novel method that learns a model of the environment to capture the stochasticity of state transitions. We first describe a key idea introduced by Bengio et al. [2021b] to decompose the GFlowNet transitions as per Figure 2, and then introduce a new approach to learn the GFlowNet policy and the dynamics model. We also discuss the applicability to different GFlowNet learning objectives and the resulting effects.

3.1 PROPOSED METHOD

Existing work on GFlowNets [Bengio et al., 2021a, Malkin et al., 2022a] typically makes the assumption that all transitions from a state s_t to the next state s_{t+1} within a trajectory are defined deterministically based on the selected action a_t (and also there is only one action a_t that can transition from s_t to $s_{t+1} = T(s_t, a_t)$, with T denoting the deterministic transition function). This applies to problems where the generative process for the objects is deterministic, which is appropriate when the actions are internal, *e.g.*, choosing what to attend, what to imagine (such as solutions to problems), how to make inferences, etc. Yet, a number of real-world tasks are stochastic, either inherently or due to the environment complexity [Antonoglou et al., 2021, Paster et al., 2022, Yang et al., 2022]. In the more general stochastic environments, the action a_t at s_t can land in several possible next states. For instance, synthesizing proteins with oligo pools can result in the generation of variants of the specified protein [Song et al., 2021].

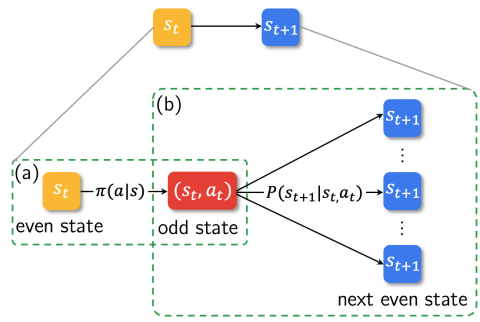


Figure 2: We decompose traditional GFlowNet transitions (top) into two steps to facilitate the GFlowNet formalization: (a) stochastically choosing the action, (b) stochastically transitioning to a new state. We call the intermediate state after (1) an odd state (and starting and final points even states), as illustrated above.

To cope with stochasticity in the transition dynamics, we decompose state transitions based on the concept of *after-states* [Sutton and Barto, 2018, Bengio et al., 2021b]. Specifically, for a transition from a state s_t to the next state s_{t+1}, we decompose the transition in two steps. First, as illustrated in part (a) in Figure 2, we sample an action a_t based on a policy π in the current state s_t (called an even state), and transit deterministically to an intermediate state (s_t, a_t), called an odd state. The flow consistency constraint for detailed balance (DB) for even-to-add transitions is shown in Eq. (4), since the backward policy probability is 1 here (we can only get to (s_t, a_t) from s_t):

$$F(s_t)\pi(a_t|s_t) = F((s_t, a_t)). \qquad (4)$$

The odd state can be considered as a hypothetical state after we apply an action before the environment gets involved [Antonoglou et al., 2021]. The environment dynamics then transform (s_t, a_t) into to the next even state s_{t+1} stochastically according to a distribution $P(\cdot|s_t, a_t)$, which is the state transition function. This second step corresponds to part (b) in Figure 2, and the corresponding flow consistency constraint is

$$F((s_t, a_t))P(s_{t+1}|(s_t, a_t)) = F(s_{t+1})\pi_B((s_t, a_t)|s_{t+1}). \qquad (5)$$

Note that an odd state can lead to many possible next even states due to stochasticity in the environment. With the introduction of odd states, we isolate the effect of choosing the action to apply in the environment and of the stochastic state transition given an action, with a deterministic and a stochastic step.

Training a Stochastic GFlowNet. Combining the two steps, we obtain a novel flow consistency constraint which

we call *stochastic GFlowNets* based on detailed balance (DB), where P denotes the state transition function:

$$\begin{aligned} &F(s_t)\pi(a_t|s_t)P(s_{t+1}|(s_t,a_t)) \\ =&F(s_{t+1})\pi_B((s_t,a_t)|s_{t+1}). \end{aligned} \tag{6}$$

In practice, for training stochastic GFlowNets, we would minimize the loss $\mathcal{L}_{\text{StochGFN-DB}}(s,a,s')$ in Eq. (7) based on the flow consistency constraint from Eq. (6), which is trained on a log-scale.

$$\begin{aligned} &(\log F(s_t) + \log \pi(a_t|s_t) + \log P(s_{t+1}|(s_t,a_t)) \\ &- \log F(s_{t+1}) - \log \pi_B((s_t,a_t)|s_{t+1}))^2 . \end{aligned} \tag{7}$$

Note that our proposed methodology is general and can be applied to other GFlowNet learning objectives such as trajectory balance (TB), as we discuss in Section 3.2.

Learning the dynamics model. Since the transition dynamics $P(\cdot|s,a)$ are unknown in general, we need to learn it. In practice, we learn a model $\hat{P}$ with parameters ϕ to approximate P through maximum likelihood estimation (other techniques from generative and dynamics modeling [Venkatraman et al., 2015] could also be applied). We optimize its parameters with the interaction data using the Adam optimizer [Kingma and Ba, 2015] based on the loss function in Eq. (8), where the output of $\hat{P}$ is a softmax distribution across all possible next states. The data is sampled from a experience replay buffer, which stores interaction data $\{s,a,s'\}$ from the GFlowNet policy and the environment.

$$\mathcal{L}_{\text{model}}(s,a,s') = -\log \hat{P}(s'|s,a) \tag{8}$$

Practical implementation. Figure 3 illustrates the major components of Stochastic GFlowNets as described above and how they interact with each other. The procedure for training Stochastic GFlowNet based on DB is summarized in Algorithm 1.

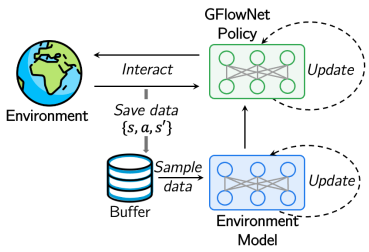


Figure 3: Illustration of Stochastic GFlowNets.

3.2 DISCUSSION ON THE APPLICABILITY TO TRAJECTORY BALANCE (TB)

As discussed in Section 3.1, our proposed method is versatile and can be applied to other GFlowNet learning objectives beyond DB. We state the flow consistency constraint

Algorithm 1 Stochastic Generative Flow Networks

1: Initialize the forward and backward policies π, π_B, and the state flow function F with parameters θ
2: Initialize the transition dynamics $\hat{P}$ with parameters ϕ
3: Initialize experience replay buffer $\mathcal{B}$
4: **for** each training step $t = 1$ to T **do**
5: Collect a batch of M trajectories $\tau = \{s_0 \to \cdots \to s_n\}$ from the policy π, and store them in $\mathcal{B}$
6: Update the stochastic GFN model according to the loss $\mathcal{L}_{\text{StochGFN-DB}}$ in Eq. (7) based on $\{\tau\}_{i=1}^{M}$
7: Sample a batch of K trajectories from $\mathcal{B}$
8: Update the transition dynamics model according to the loss $\mathcal{L}_{\text{model}}$ in Eq. (8) using data sampled from the replay buffer
9: **end for**

for Stochastic TB in Eq. (9), which is obtained via a telescoping calculation based on Eq. (6).

$$Z\prod_{t=0}^{n-1}\pi(a_t|s_t)P(s_{t+1}|(s_t,a_t)) = R(x)\prod_{t=0}^{n-1}\pi_B((s_t,a_t)|s_{t+1}) \tag{9}$$

In practice, we can train with Stochastic TB by minimizing the loss $\mathcal{L}_{\text{StochGFN-TB}}(s,a,s')$ obtained from Eq (9), *i.e.*,

$$\begin{aligned} \Bigg[\log Z + \sum_{t=0}^{n-1}\log\pi(a_t|s_t) + \sum_{t=0}^{n-1}\log\hat{P}(s_{t+1}|(s_t,a_t)) \\ - \log R(x) - \sum_{t=0}^{n-1}\log\pi_B((s_t,a_t)|s_{t+1})\Bigg]^2 . \end{aligned} \tag{10}$$

However, as TB is optimized based on a sampled trajectory instead of each transition, it can lead to a larger variance as studied in Madan et al. [2022] even in deterministic environments. This problem can be further exacerbated in stochastic environments. In Section 4.1.3, we find that the Stochastic TB underperforms relative to Stochastic DB, presumable due to a larger variance, as studied by Madan et al. [2022].

4 EXPERIMENTS

In this section, we conduct extensive experiments to investigate the following key questions: i) How much can Stochastic GFNs improve over GFNs in the presence of stochastic transition dynamics? ii) Can Stochastic GFNs be built upon different GFlowNets learning objectives? iii) Can Stochastic GFNs scale to the more complex and challenging tasks of generating biological sequences?

4.1 GRIDWORLD

4.1.1 Experimental Setup

We first conduct a series of experiments in the GridWorld task introduced in Bengio et al. [2021a] to understand the effectiveness of Stochastic GFlowNets. An illustration of the task with size $H \times H$ is shown in Figure 4. At each time step, the agent takes an action to navigate in the grid, where possible actions include operations to increase one coordinate and also a stop operation to terminate the episode, ensuring the underlying Markov decision process (MDP) is a directed acyclic graph. The agent obtains a reward $R(x)$ as defined in Bengio et al. [2021a] when the trajectory ends at a terminal state x. The reward function $R(x)$ has 4 modes located at the corners of the map as illustrated in Figure 4. The goal for the agent is to model the target reward distribution, and captures all the modes of the reward function. The shade of color in Figure 4 indicates the magnitude of rewards, where a darker color corresponds to a larger reward. We consider a variant with stochastic transition dynamics, where the randomness in the environment is injected following Machado et al. [2018], Yang et al. [2022] in GridWorld and all other benchmark tasks in Sections 4.2.1-4.2.3. Specifically, the environment transitions according to the selected action with probability $1 - \alpha$, while with probability α the environment executes a uniformly chosen action (like slipping to its neighboring regions randomly in Figure 4).

Figure 4: The GridWorld environment. The agent starts at the top-left corner and reward is largest at the four dark blue positions near the four corners (with the keys), lower in the 2×2 squares near the corner, and yet lower in other (light blue) positions. This can be extended to different sizes H, as well as different degrees of noise α in the (state, action)-to-state transitions.

We compare Stochastic GFlowNet against vanilla GFlowNets trained with detailed balance (DB) [Bengio et al., 2021b] and trajectory balance (TB) [Malkin et al., 2022a] learning objectives, Metropolis-Hastings-MCMC [Xie et al., 2021], and PPO [Schulman et al., 2017] methods. We evaluate each method in terms of the empirical L_1 error defined as $\mathbb{E}[|p(x) - \pi(x)|]$, with $p(x) = \frac{R(x)}{Z}$ denoting the true reward distribution, and we estimate π according to repeated sampling and calculating frequencies for visiting every possible state x. We also compare them in terms of the number of modes discovered by each method during the course of training. Each algorithm is run for 5 different seeds, and the performance is reported in its mean and standard deviation. We implement all baselines based on the open-source code[1], and a detailed description of the hyperparameters and setup can be found in Appendix A.1.

4.1.2 Performance Comparison

We now study the effectiveness of Stochastic GFNs on small, medium, and large GridWorlds with increasing sizes H, and different levels of stochasticity.

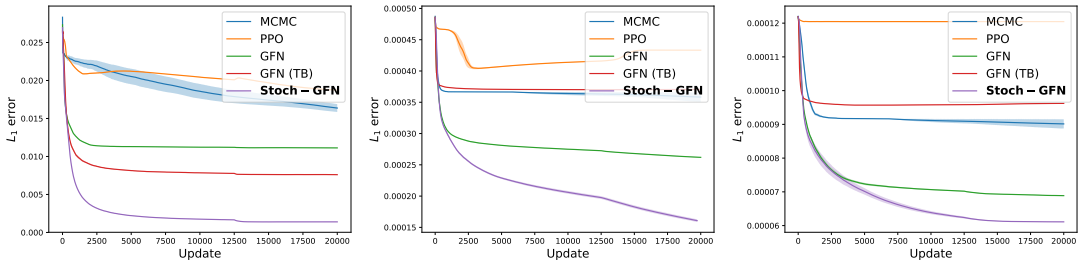
Varying sizes of the map. Figure 5 demonstrates the empirical L_1 error for each method in GridWorld (with a stochasticity level of $\alpha = 0.25$) with increasing sizes. As shown, MCMC does not perform well and PPO fails to converge. We also observe that the performance of TB gets much worse as the size of the problem increases, which may be attributed to a larger gradient variance [Madan et al., 2022]. Stochastic GFlowNets significantly outperform the baselines, and converge fastest and to the lowest empirical L_1 error. Figure 6 illustrates the number of modes discovered by each method during the course of training. As demonstrated, in stochastic environments (where the original convergence guarantees of GFlowNets do not hold), existing GFlowNet methods including DB and TB fail to discover all of the modes in maps with larger sizes. It is also worth noting that TB performs much worse than DB in terms of the number of modes discovered with increasing sizes of the maps, as it is optimized on the trajectory level with a sampled trajectory instead of the transition level as in DB, and can induce large variance. The proposed Stochastic GFlowNet method outperforms previous GFlowNet methods as well as MCMC and PPO by a large margin, while being able to efficiently discover different modes in maps with different sizes.

Varying stochasticity levels. In Figure 7, we compare different methods in a small GridWorld with an increasing level of stochasticity α. We observe that TB also fails to learn well with an increasing α, and performs worse than DB besides the decreased performance with increasing sizes. On the other hand, Stochastic GFlowNets outperform the baselines by a significant margin, and are robust to higher levels of stochasticity, successfully handling stochastic transition dynamics.

4.1.3 Compatibility with Different GFlowNet Learning Objectives

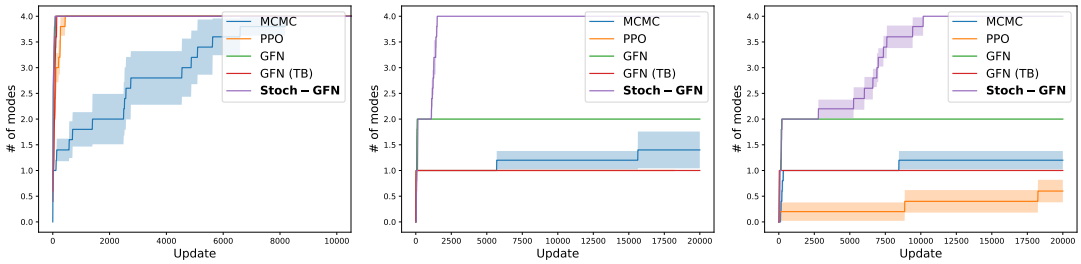
In this section, we study Stochastic GFlowNets with the trajectory balance (TB) objective as described in Section 3.2. We evaluate Stochastic TB in GridWorlds with different

[1] https://github.com/GFNOrg/gflownet



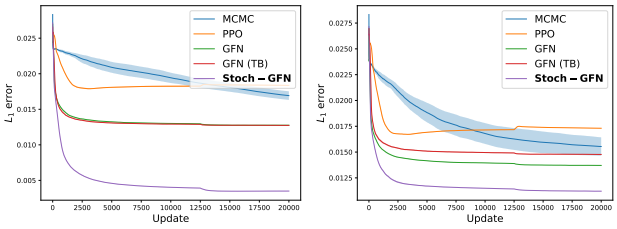
(a) Small. (b) Medium. (c) Large.

Figure 5: Comparison results of L_1 error in GridWorld with increasing sizes of the map.



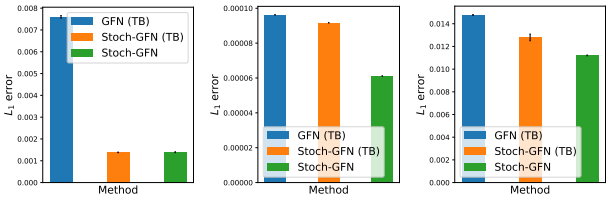
(a) Small. (b) Medium. (c) Large.

Figure 6: Comparison results of the number of modes captured during the training process in GridWorld with increasing sizes of the map.



(a) $\alpha = 0.5$. (b) $\alpha = 0.9$.

Figure 7: Results in small GridWorld with increasing stochasticity levels α.



(a) (b) (c)

Figure 8: Results of Stochastic GFlowNet when built upon the trajectory balance (TB) objective in GridWorld with increasing sizes H and stochasticity levels α. (a) Small, low stochasticity level. (b) Large, low stochasticity level. (c) Small, high stochasticity level.

sizes (including small with $H = 8$ and large with $H = 128$) and stochasticity levels (including low with $\alpha = 0.25$ and high with $\alpha = 0.9$). Specifically, Figure 8(a) corresponds to the result in a small map with a low stochasticity level, Figure 8(b) illustrates the results in a large map with a low stochasticity level, while Figure 8 shows the results in a small map with a high stochasticity level.

As shown in Figure 8, Stochastic TB (abbreviated as Stoch-GFN (TB) in the figure) greatly improves the performance of TB, validating the effectiveness of our proposed methodology. However, we observe that it underperforms relative to Stochastic DB when the scale of the problem increases or with a higher level of stochasticity (Figure 8(c)), which can be attributed to the larger variance of TB [Madan et al., 2022] in stochastic environments.

4.2 AUTOREGRESSIVE SEQUENCE GENERATION

In this section, we study Stochastic GFN on autoregressive sequence generation tasks [Malkin et al., 2022a]. We first consider a bit sequence generation task to investigate the

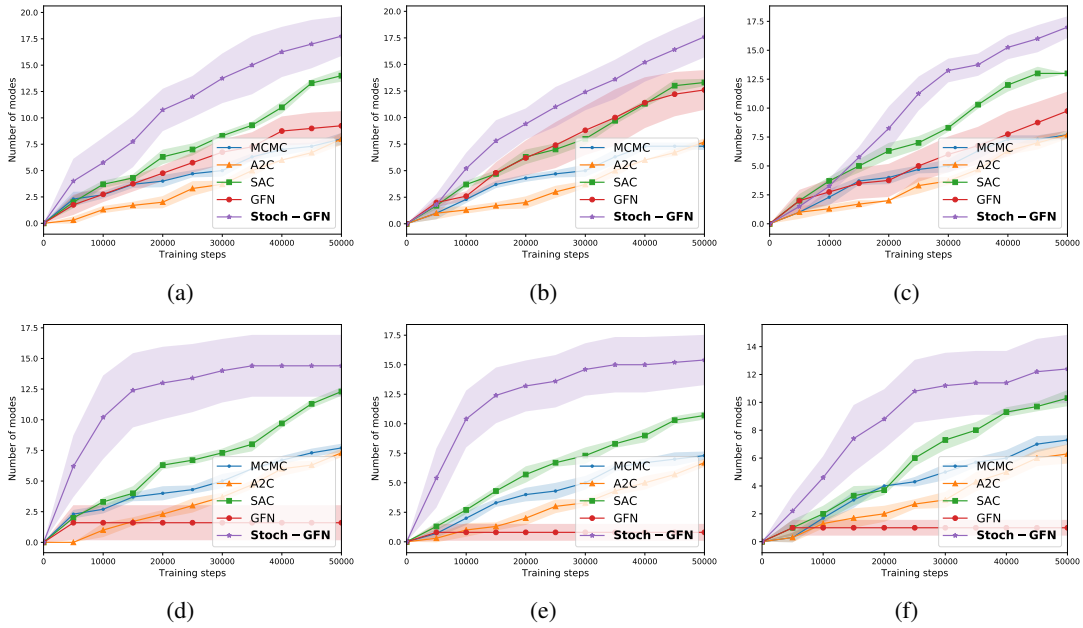


Figure 9: Results in the bit sequence generation task. The first and second rows correspond to the results of the number of bits $k = 4$ and $k = 2$. The first, second, and third columns correspond to the results of different stochasticity levels of 0.1, 0.3, and 0.5, respectively.

effect of the size of the action space and length of the trajectory with varying levels of environment stochasticity. We then study the more realistic and complex tasks of generating biological sequences.

4.2.1 Bit Sequences

Task. In the bit sequence generation task [Malkin et al., 2022a], the agent aims to generate bit sequences of length $n = 120$. At each step, the agent appends a k-bit "word" from a vocabulary V to the current state from left to right, which is a partial sequence. Note that we consider a stochastic variant of the task, with noise level α as described in Section 4.1.1. The resulting action space has a size of $|V| = 2^k$, and the length of the complete trajectories is $\frac{n}{k}$. Following Malkin et al. [2022a], we define the reward function $R(x)$ to have modes at a fixed set of bit sequences M with $R(x) = \exp(-\min_{y \in M} d(x, y))$, where d is the edit distance. We evaluate each method in terms of the number of modes discovered during the course of training.

We study the performance of Stochastic DB with different levels of stochasticity, and compare it against vanilla DB and strong baselines including Advantage Actor-Critic (A2C) [Mnih et al., 2016], Soft Actor-Critic (SAC) [Haarnoja et al., 2018], and MCMC [Xie et al., 2021]. Each method is run for 3 different seeds and we report the mean and standard deviation. More details about the experimental setup in the stochastic bit sequence generation task can be found in Appendix A.2. We use the same hyperparameters and architectures as in Malkin et al. [2022a].

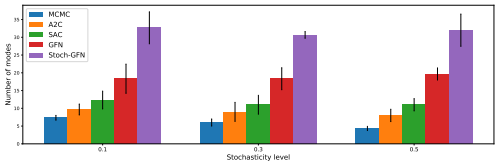
Results. Figure 9 demonstrates the number of modes captured by each method throughout the training process with different levels of stochasticity ranging from 0.1 to 0.5, where the first and second rows correspond to the results for $k = 4$ and $k = 2$, respectively. We observe that regular GFlowNets (GFN in the figure) fail to learn well, particularly when the trajectories are longer (with a smaller value of k). On the other hand, the Stochastic GFlowNet (Stoch-GFN in the figures) is robust to increasing trajectory lengths, and also performs well when the stochasticity level increases. In addition, Stoch-GFN significantly outperforms strong baselines including MCMC, A2C, and SAC, discovering more modes faster.

4.2.2 TF Bind 8 Generation

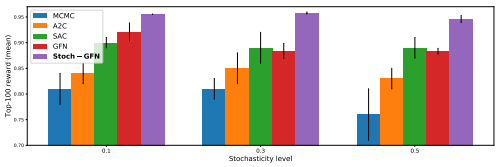
Task. We now consider the practical task of generating DNA sequences with high binding activity with particular transcription factors, following Jain et al. [2022a]. At each time step, the agent appends a symbol from the vocabulary to the right of the current state. As with the bit generation task, we consider a stochastic variant of the task following Yang et al. [2022] with random actions taken with probability α (as described in Section 4.1.1). We adopt a pre-trained neural network as the reward function following Jain et al. [2022a] that estimates the binding activity. We investigate how well Stochastic DB performs by comparing it with vanilla DB, MCMC, and RL-based methods including A2C and SAC. For evaluation, we evaluate each method in terms of the number of modes with rewards above

a threshold discovered in the batch of generated sequences. We also use the mean reward and 50th percentile score for the top 100 sequences ranked by their rewards from a batch of 2048 generated sequences for each method as in [Jain et al., 2022a, Trabucco et al., 2022]. We run each algorithm for 3 different seeds, and report their mean and standard deviation. We follow the same hyperparameters, architectures, and setup as in Jain et al. [2022a], and a detailed description of the setup can be found in Appendix A.3.

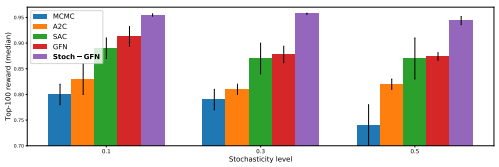
Results. Comparison results of Stoch-GFN and baselines with varying stochasticity levels (ranging from 0.1 to 0.5) in terms of the number of modes discovered with rewards above a threshold during the training process and top-100 mean rewards are summarized in Figure 10. As shown in Figure 10(a), Stoch-GFN discovers many more modes than GFN, MCMC, and RL-based methods in different stochasticity levels. Stoch-GFN also achieves higher top-100 rewards (in mean and median) than baselines as demonstrated in Figures 10(b)-(c), where the top-100 reward of GFN decrease with an increasing stochastic level. These results validate the effectiveness of Stoch-GFN in the more realistic task for biological sequence design with stochasticity in the environment.



(a) The number of modes.



(b) Top-100 reward (mean).



(c) Top-100 reward (median).

Figure 10: Results on the TF Bind 8 generation task, with better results for Stoch-GFN against MCMC, A2C, SAC and GFN baselines.

4.2.3 Antimicrobial Peptide Generation

Task. In this section, we study the realistic task of generating peptide sequences with anti-microbial properties [Malkin et al., 2022a, Jain et al., 2022a]. The agent chooses a symbol from the vocabulary that consists of 20 amino acids and a special end-of-sequence action to the current state in a left-to-right manner at each time step. The maximum length of the sequence is 60, and the size of the resulting state space is 21^{60}. We consider a stochastic variant of the task (as in Section 4.1.1) with a stochasticity level of $\alpha = 0.1$. The reward function is a pre-trained neural network that estimates the anti-microbial activity following [Malkin et al., 2022a] from the DBAASP database [Pirtskhalava et al., 2021]. As in Section 4.2.2, we generate 2048 sequences from each method and evaluate them in terms of the top-100 rewards and the number of modes discovered above a threshold. We study the performance of Stochastic DB by comparing it with DB, MCMC, and RL-based methods. We report the mean and standard deviation over 3 runs for each method. A detailed description of the setup is in Appendix A.4 following Malkin et al. [2022a].

Results. As shown in Table 1, we observe that Stoch-GFN significantly outperforms GFN and other baselines in terms of the top-100 reward. In addition, it also discovers more modes with rewards above a threshold than baseline methods, which further validates its effectiveness on the more complex and challenging task.

Table 1: Better results with Stoch-GFN on the AMP generation task. Larger is better.

	Top-100 reward	Number of modes
MCMC	0.632 ± 0.035	3.67 ± 0.58
A2C	0.682 ± 0.032	2.66 ± 0.58
SAC	0.754 ± 0.047	4.33 ± 1.33
GFN	0.748 ± 0.048	3.0 ± 3.0
Stoch-GFN	0.834 ± 0.023	19.5 ± 2.5

5 RELATED WORK

GFlowNets. The universality and effectiveness of GFlowNets have been demonstrated in various kinds of applications, including biological sequence design [Jain et al., 2022a], causal discovery and structure learning [Deleu et al., 2022, Nishikawa-Toomey et al., 2022], substructure learning of deep neural network weights via Dropout [Liu et al., 2022], multi-objective optimization [Jain et al., 2022b], and robust job scheduling problems [Zhang et al., 2023a]. Malkin et al. [2022a] proposed the trajectory balance (TB) objective to optimize GFlowNet at a trajectory level instead of at the transition level as in detailed balance Bengio et al. [2021b], but can induce large variance, where the problem is exacerbated in stochastic environments. Madan et al. [2022] propose the sub-trajectory balance method considers sub-trajectories. The early GFlowNet proposals from Bengio et al. [2021a,b] first formulated GFlowNets and pointed out possible future development directions. Originating from reinforcement learning, GFlowNets face the same long-term

credit assignment challenges to propagate downstream reward signals to earlier states. Pan et al. [2023] proposed a forward-looking GFlowNet formulation to exploit intermediate energies or rewards for more efficient credit assignment, making it possible to learn from incomplete trajectories. Pan et al. [2022] incorporates intrinsic intermediate rewards into GFlowNets by augmenting the flow values for better exploration. EB-GFN [Zhang et al., 2022b] jointly learns from data an energy/reward function along with the corresponding GFlowNet. Zhang et al. [2022a] recently points out that the relationship between generative models and GFlowNets. It is worth mentioning that Zhang et al. [2023b] shares a similar goal to our work; it extends the GFlowNet framework for stochastic reward settings with distributional modeling, while this work focuses on stochasticity in the environment transition dynamics.

Model-based Reinforcement Learning. Model-based reinforcement learning (RL) is a promising approach for improved sample efficiency compared with model-free (RL) methods [Lillicrap et al., 2015, Fujimoto et al., 2018], and has been successfully applied to many tasks such as robotics leveraging different dynamics models. The stochastic value gradient method [Heess et al., 2015] learns a hybrid of model-based and model-free RL which can learn stochastic policies in stochastic continuous control tasks. Dreamer [Hafner et al., 2019] learns latent dynamics to solve long-horizon tasks from high-dimensional images. MuZero [Antonoglou et al., 2021] combines model-based methods with Monte-Carlo tree search for planning, and it has achieved great success in game playing. Stochastic MuZero [Schrittwieser et al., 2020] learns a stochastic model for extending MuZero to stochastic environments.

6 CONCLUSION

In this paper, we introduce a new methodology, Stochastic GFlowNets, which is the first empirically effective approach to extend GFlowNets to the more general and realistic stochastic environments, where existing GFlowNet methods can fail. Our method learns the GFlowNet policy and also the environment model to capture the stochasticity in the environment. We conduct extensive experiments in standard tasks for benchmarking GFlowNets with stochastic transition dynamics. Results show that Stochastic GFlowNet learns significantly better than previous methods in the presence of stochastic transitions. It is interesting for future work to study advanced model-based approaches for approximating the transition dynamics, and also apply our method to other challenging real-world tasks.

ACKNOWLEDGEMENTS

The authors would like to thank Almer Van der Sloot, Kanika Maden, and Qingpeng Cai for insightful discussions about the paper and the baselines in the AMP generation task. Longbo Huang is supported in part by the Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108400 and 2020AAA0108403, and Tsinghua Precision Medicine Foundation 10001020109. Yoshua Bengio acknowledges the funding from CIFAR, Genentech, Samsung, and IBM.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2021.

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

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *arXiv preprint 2111.09266*, 2021b.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy

deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F.P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. *International Conference on Machine Learning (ICML)*, 2022a.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. *arXiv preprint arXiv:2210.12765*, 2022b.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Dianbo Liu, Moksh Jain, Bonaventure F. P. Dossou, Qianli Shen, Salem Lahlou, Anirudh Goyal, Nikolay Malkin, Chris C. Emezue, Dinghuai Zhang, Nadhir Hassen, Xu Ji, Kenji Kawaguchi, and Yoshua Bengio. Gflowout: Dropout with generative flow networks. *ArXiv*, abs/2210.12928, 2022.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. *ICLR'2023; arXiv:2209.12782*, 2022.

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

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *Neural Information Processing Systems (NIPS)*, 2016.

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 2211.02763*, 2022.

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

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. *ArXiv*, abs/2302.01687, 2023.

Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers fail in stochastic environments. *arXiv preprint arXiv:2205.15967*, 2022.

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, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

Li-Fu Song, Zheng-Hua Deng, Zi-Yi Gong, Lu-Lu Li, and Bing-Zhi Li. Large-scale de novo oligonucleotide synthesis for whole-genome synthesis and data storage: Challenges and opportunities. *Frontiers in bioengineering and biotechnology*, 9:689797, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

Arun Venkatraman, Martial Hebert, and J Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*, 2021.

Mengjiao Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Dichotomy of control: Separating what you can control from what you cannot. *arXiv preprint arXiv:2210.13435*, 2022.

David Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with GFlowNets. *International Conference on Learning Representations (ICLR)*, 2023a.

Dinghuai Zhang, Ricky T. Q. Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with GFlowNets. *arXiv preprint 2209.02606*, 2022a.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. *International Conference on Machine Learning (ICML)*, 2022b.

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