Random Features for Discrete Amortized Samplers

Tiago da Silva [1] **Amauri H Souza** [2 3] **Salem Lahlou** [1]

Abstract

Sequential discrete amortized samplers, such as GFlowNets, learn a stochastic decision process (SDP) that generates each state in proportion to a given unnormalized probability mass function. While successfully applied to wide-ranging tasks, they suffer from slow convergence, underdetermination, and limited expressiveness. Notably, we show these issues can be mitigated by adding random inputs to the SDP's policy network. This approach naturally provides a mixture semantics to the learning algorithm, which can be directly adapted to any existing SDP-based discrete samplers. In view of this, we refer to our method as Mixture Model Augmentation (MMA). Our empirical analysis on standard benchmark problems indicates MMA accelerates learning convergence and improves distributional accuracy.

1. Introduction & Preliminaries

The simulation of discrete stochastic models is the cornerstone of Bayesian analysis (Robert et al., 2007), with applications in therapeutics (Jain et al., 2023), vaccine development (Zhou et al., 2023), LLM fine-tuning (Hu et al., 2023), portfolio optimization (Chen et al., 2026), and robust task-scheduling (Zhang et al., 2023b). However, existing approaches are computationally intensive and sample-inefficient. We propose mitigating these challenges via the introduction of continuous, random features into the generative process, similar to the momentum variable in Hamiltonian Monte Carlo (Neal et al., 2011) or Underdamped Langevin Dynamics (Welling & Teh, 2011; Cheng et al., 2018). We show this improves learning convergence and goodness-of-fit to the target distribution.

The sampling problem. Let $\mathcal{X}$ be a discrete set and $R\colon \mathcal{X} \to \mathbb{R}_+$ be a (possibly unnormalized) probability mass function (PMF) on $\mathcal{X}$. Our objective is to generate samples from $\pi(x) \coloneqq Z^{-1}R(x)$ with $Z \coloneqq \sum_{x\in\mathcal{X}} R(x)$ as R's normalizing constant. This sampler can be used to evaluate expectations under π, which is the Gordian knot of Bayesian inference (Robert et al., 2007; Blei & et al., 2017), and to search for diverse high-probability regions in $\mathcal{X}$, a stepping stone of drug discovery (Bengio et al., 2021) and combinatorial optimization (Zhang et al., 2023a) applications.

Compositional spaces. We assume $\mathcal{X}$ is *compositional*, i.e., each $x \in \mathcal{X}$ can be described as the sink state in a directed acyclic *state graph* (SG) with nodes $\mathcal{S} \supset \mathcal{X}$. This DAG contains a single source, referred to as the *initial state* and denoted by s_o, from which each $x \in \mathcal{X}$ can be reached. Although $|\mathcal{X}|$ can be intractably large (in Section 3, for instance, we consider cases in which $|\mathcal{X}| \sim 10^9$), we assume each $s \in \mathcal{S}$ has an out-degree in $\mathcal{O}(\log|\mathcal{X}|)$.

Discrete amortized samplers. A sequential discrete amortized sampler learns a *forward policy* $p_F\colon \mathcal{S}\times\mathcal{S} \to [0,1]$ in a SG such that, starting from s_o, the marginal distribution over $\mathcal{X}$ matches R. More specifically, if $s_o \rightsquigarrow x$ denotes the set of trajectories starting at s_o and finishing at x, we aim for

$$p_\top(x) \coloneqq \sum_{\tau\in s_o\rightsquigarrow x} p_F(s_o,\tau) \propto R(x), \tag{1}$$

in which, for $\tau = (s_o,\ldots,s_T,x)$, $p_F(s_o,\cdot)$ factorizes as

$$p_F(s_o,\tau) = p_F(s_T,x)\prod_{1\le t\le T} p_F(s_{t-1},s_t).$$

In effect, $p_F(s,\cdot)$ is parameterized as a softmax neural network receiving s as input and returning a probability distribution over the children of s in the corresponding SG.

Learning p_F requires enforcing Equation (1), which is often intractable (in Section 3, for instance, $|s_o \rightsquigarrow x| \sim 10^{18}$—being even larger than $|\mathcal{X}|$). To circumvent this, we introduce a *backward policy* $p_B(x,\cdot)$ as a forward policy on the transposed state graph. For $\tau = (s_o,\ldots,s_T,x)$, we let

$$p_B(x,\tau) = p_B(x,s_T)\prod_{1\le t\le T} p_B(s_t,s_{t-1}).$$

Under these circumstances, Equation (1) can be written as

$$p_\top(x) = \mathbb{E}_{\tau\sim p_B(x,\cdot)}\left[\frac{p_F(s_o,\tau)}{p_B(x,\tau)}\right] = \frac{R(x)}{Z}. \tag{2}$$

[*]Equal contribution [1]MBZUAI, UAE [2]Federal Institute of Ceará, Brazil [3]Avra, São Paulo, Brazil. Correspondence to: Tiago da Silva <tiago.dasilva@mbzuai.ac.ae>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

Or, equivalently, $\mathbb{E}_{p_B}\left[{Zp_F(s_o,\tau)}/{R(x)p_B(x,\tau)}\right] = 1$. The condition $Zp_F(s_o,\tau) = R(x)p_B(x,\tau)$ is known as *trajectory balance* (TB) (Malkin et al., 2022). In practice, p_F is learned via stochastic gradient descent by minimizing the expected log-squared difference between TB's left- and right-hand sides under an exploratory policy p_E. There is no strict guideline for choosing p_E and p_B; they can either be fixed (Malkin et al., 2023) or learned (Shen et al., 2023).

2. Random Features for Discrete Samplers

In what follows, Section 2.1 discusses the challenges that make the search for a solution to Equation (2) a computationally difficult problem. Then, Section 2.2 introduces a novel framework for learning discrete amortized samplers, which is shown to partly address these challenges.

2.1. Challenges in Discrete Amortized Sampling

In their seminal work, Bengio et al. (2021) showed there always exists a policy function p_F abiding by Equation (1). Notably, such a p_F is unique when p_B is given. On the other hand, there are infinite pairs (p_F, p_B) satisfying Equation (2). The learning problem is thus *underdetermined*.

Challenge 2.1 (Underdetermination)**.** There are infinitely many solutions (p_F, p_B) to Equation (2).

As per the maximum entropy principle (Jaynes, 2003), a common strategy consists of setting $p_B(s,\cdot)$ as an uniform distribution (Malkin et al., 2022; Bengio et al., 2023; Deleu et al., 2022). Disputing this, however, Shen et al. (2023) demonstrated such a choice may be sub-optimal, assigning relatively low-probability to shared substructures of high-probability states and hindering training convergence.

On top of this, it is not guaranteed that the models for p_F and p_B have enough expressive power to learn the optimal solution to Equation (2). In fact, as shown in (Silva et al., 2025a; Kim et al., 2025a), there are SGs and neural network-based parameterizations for (p_F, p_B) for which any realizable sampler will be arbitrarily distant from the target R (in terms of the KL divergence (Kullback & Leibler, 1951)).

Challenge 2.2 (Limited expressiveness)**.** A neural network may not be capable of providing a sufficiently accurate approximation to any solution for Equation (2).

Although an universal approximator could in principle address Challenge 2.2, the sheer size of the state space $\mathcal{S}$ makes gradient-based search for (p_F, p_B) remarkably challenging. A reason for this is that, given $|\mathcal{S}|$'s cardinality, we must either use a very high-dimensional vector encoding for each $s \in \mathcal{S}$ to effectively represent the features which distinguish s from other states in $\mathcal{S} \setminus \{s\}$—or contend with the fact that distinct states may have near-indistinguishable representations. In both cases, the optimal policy function $s \mapsto p_F(s,\cdot)$ can be highly non-smooth, exhibiting a large Lipschitz constant and sharp sensitivity to small input variations, making it difficult to approximate using conventional gradient descent methods (Shalev-Shwartz & Ben-David, 2014).

This phenomenon, known as (near-)*state aliasing* (Whitehead & Ballard, 1991; Pardo et al., 2022), has been shown to constraint the sampler's ability to visit high-probability regions of the target distribution, which is crucial for effective learning (Malkin et al., 2023; Silva et al., 2025b). In fact, the *exploration-exploitation trade-off* has been widely studied in the amortized sampling literature (Kim et al., 2025b; Madan et al., 2025; Dall'Antonia et al., 2026), with existing methods focusing on learning an exploratory policy that generates trajectories from under-explored regions of the state space during training. In doing so, however, training cost is doubled—and the learned exploration model is discarded after training, having no inferential purpose.

Challenge 2.3 (Inefficient exploration)**.** Learning a single pair (p_F, p_B) hampers the exploration of high-probability regions of the target distribution during training.

We discuss how to address Challenges 2.1, 2.2, and 2.3 in Section 2.2. Simply put, our approach consists of adding random features as extra inputs to both p_F and p_B. These features index the space of feasible policies, alleviating Challenge 2.1—as we no longer have to commit to a single pair (p_F, p_B). In addition, we show our method provably boosts the policy network's expressive power, overcoming Challenge 2.2, and empirically improves exploration on standard benchmark tasks, mitigating Challenge 2.3.

2.2. Mixture Model Augmentation

To describe our method, which we call Mixture Model Augmentation (MMA; see Algorithm 1 in the supplement), we let $\Gamma \subseteq \mathbb{R}^d$ be a subset of the Euclidean space, which will correspond to the support of the random features' distribution. We then define the *augmented state graph* as follows.

Definition 2.4 (Augmented State Graph)**.** Let $\mathcal{G} = (\mathcal{S}, \mathcal{X})$ be a SG, $\Gamma \subseteq \mathbb{R}^d$, $\bar{\mathcal{S}} = \mathcal{S} \times \Gamma$, $\bar{\mathcal{X}} = \mathcal{X} \times \Gamma$. We call $\bar{\mathcal{G}} = (\bar{\mathcal{S}}, \bar{\mathcal{X}})$ an augmentation of $\mathcal{G}$ when $(s,\gamma) \to (s',\gamma') \in \bar{\mathcal{G}}$ iff $s \to s' \in \mathcal{G}$ and $\gamma = \gamma'$. When $\mathcal{G}$ is clear from context, we refer to $\bar{\mathcal{G}}$ as the Augmented State Graph (ASG).

The policy functions are extended accordingly to $\bar{p}_F$, $\bar{p}_B \colon \bar{\mathcal{S}} \times \bar{\mathcal{S}} \to [0,1]$, and denoted by $\bar{p}_F((s,\gamma),\cdot)$ and $\bar{p}_B((s,\gamma),\cdot)$. An important aspect of the construction in Definition 2.4 is that γ is constant along a trajectory. This choice has two notable consequences. First, the sampler's length complexity remains unchanged; a continuous transition kernel dictating the distribution of γ_{t+1} given s_{t+1}, s_t, and γ_t, which would be difficult to estimate, is not required. Second, the SDP in $\mathcal{G}$—i.e., after marginalizing γ out—becomes a convex mixture of Markov processes. Besides

conceptual clarity, this allows the use of well-known learning algorithms for mixture models (Dempster et al., 1977).

Proposition 2.5. *Let $\{(s_t, \gamma_t)\}_{t\geq 0}$ be a Markov process having $\bar{p}_F$ as the transition kernel, and let $\gamma_o \sim p(\gamma_o)$ be the initial distribution. Then, $\{s_t\}_{t\geq 0}$ is a* convex mixture *of Markov processes, i.e., defining $p_\top$ as in Equation (1),*

$$p_\top(s_t) = \int_\Gamma p(\gamma_t)\bar{p}_\top(s_t|\gamma_t)\mathrm{d}\gamma_t \tag{3}$$

for every s_t, in which we define $\bar{p}_\top(s_t, \gamma_t) = \sum_{\bar{\tau}\in(s_o,\gamma_o)\rightsquigarrow(s_t,\gamma_t)} \bar{p}_F((s_o, \gamma_o), \bar{\tau})$—as the number of trajectories from (s_o, γ_o) to (s_t, γ_t) is finite. Also, if $\tau = (s_o, \ldots, s_T, x)$ and $\bar{\tau} = ((s_o, \gamma_o), \ldots, (s_T, \gamma_o), (x, \gamma_o))$ are trajectories in SG and ASG, respectively, then

$$p_F(s_o, \tau) = \int_\Gamma p(\gamma_o)\bar{p}_F((s_o, \gamma_o), \bar{\tau})\mathrm{d}\gamma_o. \tag{4}$$

Equation (3) underlines the fact that $\{\bar{p}_\top(s_t, \gamma_t)\}_{\gamma_t\in\Gamma}$ can be thought of as the components of a (possibly continuous) mixture model. Given the problem's underdetermination (Challenge 2.1), we also expect $\bar{p}_F(\cdot|(s, \gamma))$ to be distinct from $\bar{p}_F(\cdot|(s, \gamma'))$ for $\gamma \neq \gamma'$. This suggests a learning algorithm for $\bar{p}_F$ and $\bar{p}_B$. Let $R(x, \gamma) := R(x) \cdot p(\gamma)$, in which p is a density function on Γ. Clearly, the normalizing constant of $R(x, \gamma)$ is $\sum_{x\in\mathcal{X}} \int_\Gamma R(x, \gamma)\mathrm{d}\gamma = \sum_{x\in\mathcal{X}} R(x) \int_\Gamma p(\gamma)\mathrm{d}\gamma =: Z$, i.e., the same of $R(x)$. As in Equation (2), our goal is for each $p_\top(x, \gamma)$ to match $R(x, \gamma)$, i.e.,

$$\bar{p}_\top(x, \gamma) \propto R(x, \gamma). \tag{5}$$

Under these conditions, Proposition 2.5 ensures $p_\top(x) \propto \int R(x, \gamma)\mathrm{d}\gamma \propto R(x)$. Again, similarly to Equation (2), $\bar{p}_\top(x, \gamma) \propto R(x, \gamma)$ can be satisfied through the *augmented trajectory balance* (ATB) condition,

$$Z \cdot \bar{p}_F((s_o, \gamma), \bar{\tau}) = R(x)p(\gamma) \cdot \bar{p}_B((x, \gamma), \bar{\tau}),$$

with $\bar{\tau} = ((s_o, \gamma), \ldots, (x, \gamma))$, which can be enforced by minimizing the corresponding *ATB loss*,

$$\mathcal{L}_{\text{ATB}}(\bar{p}_F, \bar{p}_B) = \mathop{\mathbb{E}}_{\substack{\gamma\sim q(\cdot),\\ \bar{\tau}\sim\bar{p}_E(\cdot|(s_o,\gamma))}} \left(\frac{Z \cdot \bar{p}_F((s_o, \gamma), \bar{\tau})}{R(x, \gamma)p_B((x, \gamma), \bar{\tau})}\right)^2. \tag{6}$$

Here, q and $\bar{p}_E$ are exploratory policies for γ and $\bar{\tau}$, respectively. Our training algorithm is summarized in Algorithm 1. Once $\bar{p}_F$ is learned, we generate samples from $R(x)$ by picking $\gamma \sim p(\gamma)$ and $(x, \gamma) \sim \bar{p}_\top(x, \gamma)$; this is described in Algorithm 2.

Learning $\bar{p}_F$ and $\bar{p}_B$. To learn $\bar{p}_F$ and $\bar{p}_B$, we define $\phi\colon \mathcal{S} \to \mathbb{R}^{d_{\text{state}}}$ and $\psi\colon \Gamma \to \mathbb{R}^{d_{\text{noise}}}$ as state and random feature encoders, respectively, and let

$$\eta(s, \gamma) = \phi(s) \oplus \psi(\gamma),$$
$$\bar{p}_F((s, \gamma), \cdot) = \text{Softmax}\left((\mathbf{W}_F\eta(s, \gamma)) \odot \mathbf{m}_F(s)\right),$$
$$\text{and } \bar{p}_B((s, \gamma), \cdot) = \text{Softmax}\left((\mathbf{W}_B\eta(s, \gamma)) \odot \mathbf{m}_B(s)\right),$$

in which $\oplus$ represents concatenation, $\odot$, element-wise product, and $\mathbf{W}_F \in \mathbb{R}^{d_{\text{F}}\times(d_{\text{state}}+d_{\text{noise}})}$ and $\mathbf{W}_B \in \mathbb{R}^{d_{\text{B}}\times(d_{\text{state}}+d_{\text{noise}})}$ are learnable weights for the forward and backward policies. Similarly, $\mathbf{m}_F(s) \in \{1, -\infty\}^{d_{\text{F}}}$ and $\mathbf{m}_B(s) \in \{1, -\infty\}^{d_{\text{B}}}$ are domain-dependent masks representing which transitions are allowed in s. This is, except for $\psi(\gamma)$, a standard approach for parameterizing sequential discrete amortized samplers.

Connection to RNI GNNs. Our approach is connected to Random Node Initialization (RNI) in Graph Neural Networks (GNNs) (Sato et al., 2021; Abboud et al., 2021; Papp et al., 2021). Briefly, in graph prediction tasks, RNI assigns random vectors to each node, concatenating them to the existing features. This deceptively simple strategy significantly increases a GNN's expressive power and, under standard assumptions, even endows GNNs with universal approximator capabilities (Abboud et al., 2021, Theorem 1). Interestingly, as per Proposition 2.5, a RNI GNN may be interpreted as an GNN ensemble.

Drawing on this connection and on the analysis in (Silva et al., 2025a) of the limitations of GNN-based discrete amortized samplers, we illustrate below how MMA can strictly improve a model's expressivity, as in Challenge 2.2.

Example 2.6 (MMA for graph-structured tasks)**.** Consider the following SG: s_o is a 2-regular graph with 5 anonymous (i.e., with identical features) nodes, and each child is obtained by adding an edge to s_o (i.e., v_1—v_4 and v_2—v_3).

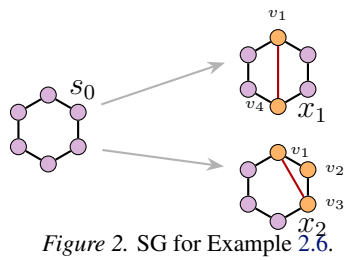


Figure 2. SG for Example 2.6.

We define R_1 and R_2 as the target PMFs for x_1 and x_2, respectively. Since x_1 and x_2 are non-isomorphic, we may set $R_1 \neq R_2$. As in (Silva et al., 2025a), we parameterize $p_F(s_o, \cdot)$ with a 1-WL GNN encoder (Xu et al., 2019), which produces an embedding $\mathbf{h} \in \mathbb{R}^{5\times h}$ for each node, and sample each child according to

$$p_F(s_o, x_1) \propto \exp\left\{\phi\left(\mathbf{h}_{v_1} + \mathbf{h}_{v_4}\right)\right\}$$
$$\text{and } p_F(s_o, x_2) \propto \exp\left\{\phi\left(\mathbf{h}_{v_1} + \mathbf{h}_{v_3}\right)\right\}$$

with $\phi\colon \mathbb{R}^h \to \mathbb{R}$ as a learnable function. Due to v_i's indistinguishability, $\mathbf{h}_{v_i} = \mathbf{h}_{v_1}$ for $i \in \{1, \ldots 5\}$, and

(a) CORNERS. *(b)* GAUSSIAN.

Figure 1. MMA improves the sampler's goodness-of-fit in the Hypergrid domain.

the distribution above is uniform, being incapable of approximating our target (R_1, R_2) regardless of the encoder's size (h). Upon introducing random features, $\gamma \in \mathbb{R}^{5\times d_{\text{noise}}}$, however, our model becomes

$$\begin{aligned} \bar{\mathbf{h}} &= \mathbf{h} \oplus \gamma \in \mathbb{R}^{5\times(h+\text{d}_{\text{noise}})}, \\ \bar{p}_F((s_o,\gamma),(x_1,\gamma)) &\propto \exp\left\{\phi\left(\bar{\mathbf{h}}_{v_1} + \bar{\mathbf{h}}_{v_4}\right)\right\}, \\ \text{and } \bar{p}_F((s_o,\gamma),(x_2,\gamma)) &\propto \exp\left\{\phi\left(\bar{\mathbf{h}}_{v_1} + \bar{\mathbf{h}}_{v_3}\right)\right\}, \end{aligned} \tag{7}$$

with $\phi\colon \mathbb{R}^{h+d_{\text{noise}}} \to \mathbb{R}$. When γ is drawn from a continuous distribution (e.g., Gaussian), $\gamma_{v_i} \neq \gamma_{v_j}$ for $i \neq j$ almost surely; the nodes are no longer indistinguishable. Indeed, such parameterization can sample from any distribution as long as ϕ has enough capacity.

In fact, Abboud et al. (2021, Theorem 1) ensures a sampler parameterized as in Equation (7) can fit any distribution over graphs—given enough model capacity. An important empirical issue, however, is whether learning a policy function minimizing $\mathcal{L}_{\text{ATB}}$ is harder than finding an approximate solution to Equation (2), which brings us back to Challenge 2.3. In this regard, the following section shows MMA significantly enhances state space exploration and learning convergence.

3. Experiments & Discussions

We consider below the Hypergrid, Lazy Random Walk (supplement), and Set Generation domains, which are standard benchmark tasks for discrete amortized sampling. The first two probe the sampler's exploration capabilites and allow for direct visualization of the learned distribution (Figures 1 and 4). The Set Generation domain, on the other hand, admits tractable goodness-of-fit assessment even in intractably large spaces (e.g., $|\mathcal{X}| \sim 10^9$; see Figures 3). Across all experiments, we let $\gamma \sim \mathcal{N}(\mathbf{0}, 0.25 \cdot \mathbf{I})$ with $\mathbf{0} \in \mathbb{R}^{16}$ and $\mathbf{I} \in \mathbb{R}^{16\times 16}$ as the 16-dimensional identity matrix. We also set $p_E = (1-\epsilon)\cdot p_F + \epsilon \cdot p_U$ and $q = \mathcal{N}(\mathbf{0}, 0.25 \cdot \mathbf{I})$ for our loss function in Equation (6).

Hypergrid. (Bengio et al., 2021; Malkin et al., 2023) The initial state is $s_o = (0,0)$, and each transition corresponds to adding $(1,0)$ or $(0,1)$ to the current state, or stopping. The state space is $\mathcal{X} = \{0,\ldots,11\}^2 \times \{\top\}$, with $\top$ being the stop sign, and $\mathcal{S} = \{0,\ldots,11\}^2$. The target distributions, alongside the approximations obtained by a discrete amortized sampler with and without MMA, are shown in Figure 1. We also present in Figure 5 the average $\log R(x)$ of the top-10 highest probability states encountered throughout training, which measures the sampler's capability in visiting the target distribution's modes. As expected, MMA accelerates learning convergence.

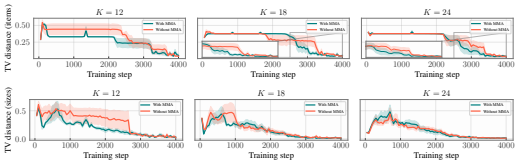


Figure 3. Distributional accuracy for the Set Generation task for samplers with and without MMA; MMA accelerates convergence.

Set Generation. (Bengio et al., 2023) The task starts with $s_o = \emptyset$, to which elements from a given set $\mathcal{U} = \{1,\ldots,U\}$ are added until either a stop sign is sampled or a prescribed size, K, is achieved. The target distribution is defined as

$$\log R(x) = \sum_{u\in x} \ell(u),$$

with $\ell\colon \mathcal{U} \to \mathbb{R}$ as a prespecified function (see Section B in the supplement). Under this formulation, both quantities

$$p_{\text{item}}(u) = \mathbb{P}_{x\sim R}\left[u \in x\right] \text{ and } p_{\text{size}}(k) = \mathbb{P}_{x\sim R}\left[|u| = k\right],$$

for $u \in \mathcal{U}$ and $k \in \{0,\ldots,K\}$, can be tractably computed. Given samples $\{x_1,\ldots,x_N\}$, we may estimate the probabilities above as

$$\hat{p}_{\text{item}}(u) = \textstyle\sum_{n=1}^{N} \frac{\mathbf{1}[u\in x_n]}{N} \text{ and } \hat{p}_{\text{size}}(k) = \sum_{n=1}^{N} \frac{\mathbf{1}[|x_n|=k]}{N}.$$

We set $U = 32$ and report $\max_{u\in\mathcal{U}} |\hat{p}_{\text{item}}(u) - p_{\text{item}}(u)|$ and $\max_k |p_{\text{size}}(k) - p_{\text{size}}(k)|$ throughout training in Figure 3. Notably, MMA results in a better goodness-of-fit.

Conclusions. In this short communication, we outlined the current challenges in training discrete amortized samplers, and introduced a strategy for addressing them based on augmenting the state space with an auxiliary, continuous variable. Conceptually, our approach may be interpreted as learning a mixture of Markovian samplers (Proposition 2.5), which provably boosts expressive power (Example 2.6) and improves accuracy (Figures 1, 3 and 4 and Table 1). We are presently investigating how to better use the auxiliary variable, γ, during training to further improve convergence, and how to properly incorporate it into the policy network.

References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, pp. 2112–2118, 2021.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. In *NeurIPS (NeurIPS)*, 2021.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *Journal of Machine Learning Research (JMLR)*, 2023.

Blei, D. M. and et al. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

Chen, B., Ding, H., Shen, N., Guo, T., Huang, J., Liu, L., and Zhang, M. AlphaSAGE: Structure-aware alpha mining via GFlownets for robust exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zRKF4ln2VE.

Cheng, X., Chatterji, N. S., Bartlett, P. L., and Jordan, M. I. Underdamped langevin mcmc: A non-asymptotic analysis, 2018. URL https://arxiv.org/abs/1707.03663.

Dall'Antonia, P., da Silva, T., Csillag, D., Lahlou, S., and Mesquita, D. Avoid what you know: Divergent trajectory balance for gflownets, 2026. URL https://arxiv.org/abs/2602.17827.

Deleu, T., Góis, A., Emezue, C. C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. Bayesian structure learning with generative flow networks. In *UAI*, 2022.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1): 1–38, 1977.

Hu, E. J., Jain, M., Elmoznino, E., Kaddar, Y., and et al. Amortizing intractable inference in large language models, 2023.

Jain, M., Deleu, T., Hartford, J., Liu, C.-H., Hernandez-Garcia, A., and Bengio, Y. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2023.

Jaynes, E. T. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, 2003. ISBN 978-0-521-59271-0. doi: 10.1017/CBO9780511790423.

Jordan, K., Jin, Y., Boza, V., Jiacheng, Y., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Kim, H., Lee, S., and hwan Oh, M. Symmetry-aware gflownets, 2025a. URL https://arxiv.org/abs/2506.02685.

Kim, M., Choi, S., Yun, T., Bengio, E., Feng, L., Rector-Brooks, J., Ahn, S., Park, J., Malkin, N., and Bengio, Y. Adaptive teachers for amortized samplers. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=BdmVgLMvaf.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kullback, S. and Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 1951.

Madan, K., Lamb, A., Bengio, E., Berseth, G., and Bengio, Y. Towards improving exploration through sibling augmented GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HH4KWP8RP5.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in GFlownets. In *NeurIPS (NeurIPS)*, 2022.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2011.

Papp, P. A., Martinkus, K., Faber, L., and Wattenhofer, R. DropGNN: Random dropouts increase the expressiveness of graph neural networks. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=fpQojkIV5q8.

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. Time limits in reinforcement learning, 2022. URL https://arxiv.org/abs/1712.00378.

Robert, C. P. et al. *The Bayesian choice: from decision-theoretic foundations to computational implementation*, volume 2. Springer, 2007.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the*

2021 SIAM International Conference on Data Mining (SDM), 2021.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

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

Silva, T., Alves, R. B., de Souza da Silva, E., Souza, A. H., Garg, V., Kaski, S., and Mesquita, D. When do GFlownets learn the right distribution? In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=9GsgCUJtic.

Silva, T., Souza, A. H., Rivasplata, O., Garg, V., Kaski, S., and Mesquita, D. Generalization and distributed learning of GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=PJNhZoCjLh.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011.

Whitehead, S. D. and Ballard, D. H. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR)*, 2019.

Zhang, D., Dai, H., Malkin, N., Courville, A., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial optimization problems with gflownets. In *NeurIPS (NeurIPS)*, 2023a.

Zhang, D. W., Rainone, C., Peschl, M., and Bondesan, R. Robust scheduling with gflownets. In *International Conference on Learning Representations (ICLR)*, 2023b.

Zhou, M., Yan, Z., Layne, E., Malkin, N., Zhang, D., Jain, M., Blanchette, M., and Bengio, Y. Phylogfn: Phylogenetic inference with generative flow networks, 2023.

Table 1. Total variation between the learned and target distributions. MMA improves goodness-of-fit for both the Hypergrid and Lazy Random Walk domains. Results were averaged across three independent runs.

	With MMA	Without MMA
Hypergrids (CORNERS)	$\mathbf{0.039}_{\pm 0.010}$	$0.189_{\pm 0.042}$
Hypergrids (GAUSSIAN)	$\mathbf{0.085}_{\pm 0.027}$	$0.121_{\pm 0.003}$
Lazy Random Walk	$\mathbf{0.052}_{\pm 0.006}$	$0.107_{\pm 0.086}$

A. Proof of Proposition 2.5

Recall $\{(s_t, \gamma_t)\}_{t\geq 0}$ is a Markov process. Also, by definition, $\gamma_o = \gamma_t$ for $t \geq 0$. As a consequence, the law for the (generally non-Markovian) stochastic process $\{s_t\}_{t\geq 0}$ is

$$\begin{aligned} p_F(s_{t+1}|s_t, \dots, s_o) = \frac{p_F(s_o, \dots, s_{t+1})}{p_F(s_o, \dots, s_t)} &\propto \int_{\Gamma^{t+2}} p(\gamma_o)\bar{p}_F((s_o, \gamma_o), \dots, (s_{t+1}, \gamma_t))\mathrm{d}(\gamma_o, \dots, \gamma_{t+1}) \\ &= \int_{\Gamma} p(\gamma_o)\bar{p}_F((s_o, \gamma_o), \dots, (s_{t+1}, \gamma_o))\mathrm{d}\gamma_o \\ &\propto \int_{\Gamma} p(\gamma_o)\bar{p}_F((s_{t+1}, \gamma_o)|(s_o, \gamma_o), \dots, (s_t, \gamma_o))\mathrm{d}\gamma_o. \end{aligned}$$

In particular, we showed that

$$p_F(s_o, \dots, s_t) = \int_{\Gamma} \bar{p}_F((s_o, \gamma_o), \dots, (s_t, \gamma_o))\mathrm{d}\gamma_o.$$

Consequently,

$$\begin{aligned} p_\top(s_t) = \sum_{(s_o,\dots,s_{t-1})} p_F(s_o, \dots, s_t) &= \sum_{(s_o,\dots,s_{t-1})} \int_{\Gamma} p(\gamma_o)\bar{p}_F((s_o, \gamma_o), \dots, (s_t, \gamma_o))\mathrm{d}\gamma_o \\ &= \int_{\Gamma} p(\gamma_o) \sum_{(s_o,\dots,s_{t-1})} p_F((s_o, \gamma_o), \dots, (s_t, \gamma_o))\mathrm{d}\gamma_o \\ &= \int_{\Gamma} p(\gamma_o)\bar{p}_\top(s_t|\gamma_o)\mathrm{d}\gamma_o = \int_{\Gamma} \bar{p}_\top(s_t, \gamma_o)\mathrm{d}\gamma_o. \end{aligned}$$

In conclusion, if $\tau = (s_o, \dots, x)$ and $\bar{\tau} = ((s_o, \gamma_o), \dots, (x, \gamma_o))$,

$$p_F(s_o, \tau) = \int_{\Gamma} p(\gamma_o)\bar{p}_F((s_o, \gamma_o), \bar{\tau})\mathrm{d}\gamma_o,$$

by definition. This demonstrates Proposition 2.5, and shows that $\{s_t\}_{t\geq 0}$ is a convex mixture of Markov chains.

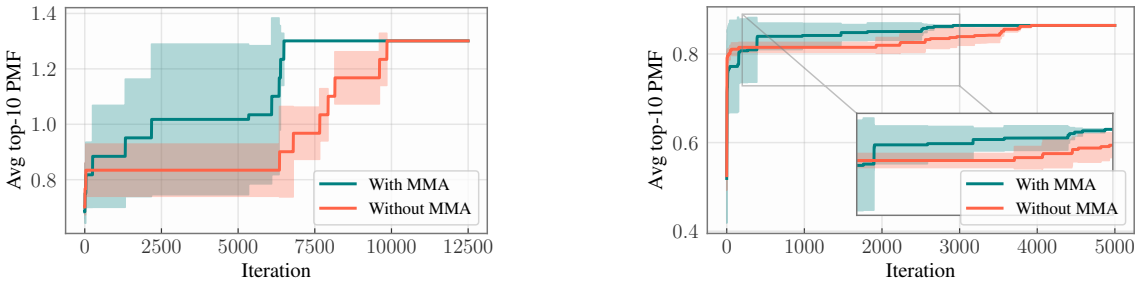
B. Additional experiments & further details

This section presents further experiments and provides further implementation details.

B.1. More experiments

Figure 4. MMA improves the sampler's accuracy.

Lazy Random Walk. (Dall'Antonia et al., 2026) Similarly to the Hypergrid domain, we let $s_o = (0, 0, 0)$, with each transition corresponding to shifting the current state by $(i, 0, 1)$ or $(0, i, 1)$ for $i \in \{-1, 0, 1\}$ until the third coordinate, representing the timestamp, reaches a predefined limit T. The state space is $\mathcal{X} = \{-12, \dots, 12\}^2 \times \{25\}$ and $\mathcal{S} = \{-12, \dots, 12\}^2 \times \{0, \dots, 24\}$ (i.e., $T = 25$), and the target distribution is depicted in Figure 4 with the learned approximations. Again, MMA results in better



(a) CORNERS. *(b)* GAUSSIAN.

Figure 5. MMA speeds up the discovery of high-probability regions, measured as the average PMF of the top-10 most probable encountered states throughout training (i.e., according to $R(x)$; the auxiliary γ is not considered in this assessment).

distributional accuracy. In addition, Table 1 shows the total variation (TV) distance between each sampler and its target averaged across three independent runs.

Algorithm 1 MMA Training

Input: Target R, state graph $\mathcal{G} = (\mathcal{S}, \mathcal{X})$, prior $p(\gamma)$, exploratory policies $\bar{p}_E$ and q, learning rate α, number of steps T
Output: Learned forward policy $\bar{p}_F$ and backward policy $\bar{p}_B$
Initialize parameters θ of $\bar{p}_F$, φ of $\bar{p}_B$, and $\log Z$ **for** $n = 1, \ldots, N$ **do**
  Sample $\gamma \sim q(\cdot)$ `// Draw γ from exploratory distribution`
  Set augmented initial state $\bar{s}_o \leftarrow (s_o, \gamma)$ Sample trajectory $\bar{\tau} = \big(\bar{s}_o, \bar{s}_1, \ldots, \bar{s}_T, (x, \gamma)\big) \sim \bar{p}_E(\cdot \mid \bar{s}_o)$ `// Roll out policy on ASG`
  Compute ATB loss:

$$\mathcal{L}_{\text{ATB}} = \Big(\log Z + \log \bar{p}_F(\bar{s}_o, \bar{\tau}) - \log R(x) - \log p(\gamma) - \log \bar{p}_B((x, \gamma), \bar{\tau})\Big)^2$$

  `// In practice, we average gradients over a batch of trajectories.`
  Update $(\theta, \varphi, \log Z) \leftarrow (\theta, \varphi, \log Z) - \alpha \nabla_{\theta, \varphi, \log Z} \mathcal{L}_{\text{ATB}}$
return $\bar{p}_F$, $\bar{p}_B$

Algorithm 2 MMA Sampling

Input: Learned forward policy $\bar{p}_F$ (from Algorithm 1), prior $p(\gamma)$, state graph $\mathcal{G} = (\mathcal{S}, \mathcal{X})$
Output: Sample $x \sim \pi(x) \propto R(x)$
Sample $\gamma \sim p(\gamma)$ `// e.g.,` $\mathcal{N}(\mathbf{0}, 0.25 \cdot \boldsymbol{I})$
Set augmented initial state $\bar{s} \leftarrow (s_o, \gamma)$ **while** $\bar{s}$ *is not a terminal state* **do**
  Compute transition distribution:

$$\bar{p}_F(\bar{s}, \cdot) = \text{Softmax}\Big(\big(W_F\, \eta(s, \gamma)\big) \odot m_F(s)\Big), \qquad \eta(s, \gamma) = \varphi(s) \oplus \psi(\gamma)$$

  Sample next state $\bar{s}' \sim \bar{p}_F(\bar{s}, \cdot)$ `// γ held fixed throughout`
  $\bar{s} \leftarrow \bar{s}'$
Extract terminal state x from $\bar{s} = (x, \gamma)$ **return** x

B.2. Experimental details

We set $d_{\text{noise}} = 16$. All models use a 2-layer MLP with hidden dimension 128 (64 for Lazy Random Walk), batch size 64, and the (augmented) trajectory balance loss. Policy parameters are optimised with Muon (lr $= 10^{-3}$) (Jordan et al., 2024) and $\log Z$ with Adam (lr $= 10^{-2}$) (Kingma & Ba, 2014). We discuss below the target distribution for each domain.

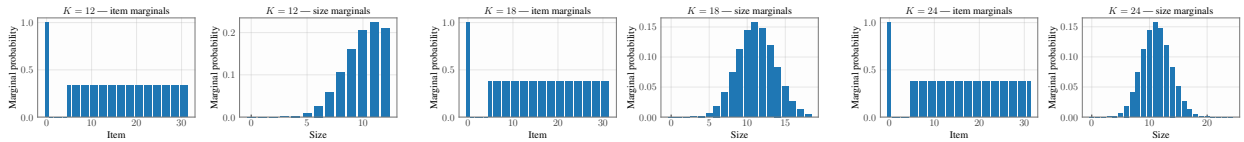


Figure 6. Marginal probabilities over item belongings and set sizes for the target distribution described in Equation (10), computed via SOS.

Hypergrids – CORNERS. ($d = 2$, $H = 12$, (Malkin et al., 2023)). The reward places mass in $[-1, 1]^2$:

$$\log R(x) = \log\big(0.5 \cdot \mathbf{1}[\|x'\|_\infty > 0.5] + 2 \cdot \mathbf{1}[0.6 < \|x'\|_\infty < 0.8] + 10^{-3}\big)\,, \tag{8}$$

where $x' = 2x/(H-1) - 1$.

Hypergrids – GAUSSIAN. The target is an equal-weight mixture of 9 isotropic Gaussians ($\sigma^2 = 10^{-2}$) whose means tile a 3×3 grid in $[-0.9, 0.9]^2$:

$$\log R(x) = \log \frac{1}{9} \sum_{k=1}^{9} \mathcal{N}(x';\, \mu_k,\, \sigma^2 I). \tag{9}$$

Lazy Random Walk States are signed-integer vectors in $[-H, H]^d$; the reward follows a graded cosine along the first coordinate: $\log R(x) = \cos(2\,x[0])$.

Set Generation ($U = 32$ items, $K \in \{12, 18, 24\}$). States are binary vectors $s \in \{0, 1\}^U$ representing subsets; the reward is log-linear with fixed item utilities $\ell(u)$:

$$\log R(x) = \sum_{u=1}^{U} x_u\, \ell(u), \quad \ell(0) = 100,\ \ell(i) = -50 \text{ if } i \in \{1, \ldots, 4\},\ \ell(i) = -0.5,\ \text{otherwise}. \tag{10}$$

Item 0 is a high-value surrounded by toxic items at indices 1–4, while the remaining items carry a small penalty. We display the corresponding item and size marginals, used for the evaluation of the distance metric in Figure 3, in Figure [ref]. As explained, these quantities can be tractably computed through via dynamic programming adapted from the Sum of Subsets, SOS, algorithm. Importantly, Figure 6 helps in explaining the reason for which there is no significant difference between samplers with and without MMA for the size marginals in Figure 3 (bottom row) when $K \geq 12$: in this case, the distribution over sizes is unimodal and binomial-shaped, being straightforward to approximate. In contrast, the marginal distribution over item belonging (top row in Figure 3) is multi-modal and sparse, making the search for an accurate approximation challenging.