LEARNING A MIXTURE OF GFLOWNETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning an ensemble of GFlowNets to sample from a discrete target distribution has become a common approach for achieving better state space exploration and convergence than that of a monolithic sampler. However, these methods often add a substantial runtime overhead to the base model, and their conceptual connection remains elusive. To address this, we first propose a general-purpose theoretical framework for describing a mixture of GFlowNets, which we specialize into continuously (CI) and discretely indexed (DI) collections. On the one hand, we show CI GFlowNets can be interpreted through the lens of a *random features expansion*, provably boosting the sampler's expressivity in graph-structured tasks and reducing learning instability via spectral shifting. On the other hand, we demonstrate DI GFlowNets encompass prior approaches for GFlowNet training and provide the foundation for the newly proposed *Stratum-Conditioned* (SC) GFlowNets. This method, which is inspired by the Doob's h-transform of Markov chains, decomposes the state space according to a prescribed modular function and restricts each component to sample from a distinct subset of it. Importantly, SC GFlowNets support centralized and component-wise embarrassingly parallel training, and we show both of them significantly speed up learning convergence and mode coverage without introducing any non-negligible extra computation.

1 INTRODUCTION

Suppose we are given an unnormalized distribution R over a discrete space $\mathcal{X}$. A common problem in both statistical inference and probabilistic machine learning is to draw samples from $\mathcal{X}$ distributed according to R (Bishop, 2007; Neal et al., 2011; Blei & et al., 2017). When $\mathcal{X}$ is intractably large—e.g., the space of graphs containing N nodes or of subsets of a set with N elements for largish N—addressing this challenge often requires the use of approximate algorithms such as Markov chain Monte Carlo (MCMC) or variational inference (VI). Notably, MCMC suffers from slow convergence and correlated samples, especially in discrete spaces, and its correctness cannot be tractably assessed in general (Owen, 2013). VI, in contrast, reframes the sampling problem as that of finding the best approximation to R within a tractable family, thereafter enabling the generation of independent samples from a measurably correct distribution (Malkin et al., 2023; Zimmermann et al., 2023).

As a major milestone of VI for discrete spaces, GFlowNets (Bengio et al., 2021; Lahlou et al., 2023) learn a sequential decision process that iteratively constructs each $x \in \mathcal{X}$ from an initial (empty) state in proportion to $R(x)$. Since their conception, they have been successfully deployed into applications ranging from causal discovery (Deleu et al., 2022; 2023) to LLM fine-tuning (Hu et al., 2023; Venkatraman et al., 2024) and phylogenetic inference (Zhou et al., 2024). Despite their potential, they suffer from insufficient state space exploration (Kim et al., 2025b; Madan et al., 2025), slow training convergence (Liu et al., 2024), and learning instability (Figure 10), often failing to simulate even simple multi-modal distributions defined over relatively small $\mathcal{X}$ (Figure 20).

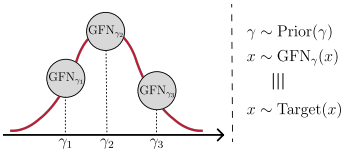


Figure 1: An ensemble of GFlowNets that, when averaged, matches the target.

To mitigate this issue, recent works have proposed learning an *ensemble* of GFlowNets in which each component focuses on a nearly-unique subset of the state space (Yu et al., 2026a;b; Dall'Antonia et al., 2026b); in doing so, they incentivize the exploration of unvisited high-probability regions and accelerate learning convergence. However, they often incur a substantial computational overhead—requiring several neural networks to be sep-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

arately trained—and, despite being similar, lack a shared conceptual foundation. In this scenario, our first contribution is to introduce a unifying theory to describe a mixture of GFlowNets (Figure 1), shedding light on the applicability and limitations of such an approach. Drawing on this framework, we propose distinct algorithms that we show to both stabilize and accelerate learning convergence.

In this context, we separate our presentation into continuously (CI) and discretely indexed (DI) mixtures of GFlowNets. On the one hand, CI GFlowNets learn an uncountable collection of samplers, each of which is associated to a real-valued vector whose value serves as an extra input to the underlying neural network parameterizing the sampler. We demonstrate this formulation is connected to the theory of random features for graph learning (Sato et al., 2021; Abboud et al., 2021), providing a computationally simple approach to provably boost the sampler's expressivity in graph-structured tasks (Sections C and 4). In addition, we show CI GFlowNets stabilize learning by shifting the spectrum of the loss function's Hessian, reducing the dependence of the learned model on exogenous factors such as the initial value of the pseudorandom number generator (Proposition C.7).

On the other hand, DI GFlowNets consist of a finite (or countably infinite) mixture of samplers. Each sampler learns in proportion to a *re-weighted* target; for the kth component, this is $x \mapsto R(x)q(k|x)$ for a *posterior mixing distribution* $q(\cdot|x)$ (see Section 5). In this sense, DI GFlowNets support two learning modalities: centralized, with an embedding of k being used as extra input to a component-wise shared model, and embarrassingly parallel (Neiswanger et al., 2014), with distinct components being independently trained across non-communicating processes. We prove this framework encompasses two prior methods, Subgraph Asynchronous Learning (Silva et al., 2025b) and Boosted GFlowNets (Dall'Antonia et al., 2026b), unifying otherwise disparate algorithms.

As the simplest non-trivial instantiation of DI GFlowNets, we propose setting $q(\cdot|x)$ as a sparse distribution assigning zero probability to all but a few components. This effectively corresponds to defining a *covering* of the state space. We show, however, that conditioning a sampler to exclusively explore an arbitrary subset of $\mathcal{X}$ is NP-complete in general. We consequently introduce *Stratum-Conditioning* (SC) as a tractable approach for learning DI GFlowNets. Simply put, SC partitions the state space by binning the outputs of a modular and real-valued function defined on $\mathcal{X}$, with each bin being called a *stratum* (hence the name). This function can be set to the trajectory length required for reaching $x \in \mathcal{X}$, for instance, or as the distance between $x \in \mathcal{X}$ and a chosen reference point according to a prescribed metric (Section 5.1). We demonstrate the set of samplers learned in this way are connected via Doob's h-transform of a base GFlowNet (Section 5.2).

Our experiments highlight SC GFlowNets substantially improve both learning convergence and state space exploration across diverse benchmark tasks. Importantly, contrarily to recent algorithmic advances in the GFlowNet literature, SC *adds no computational overhead* to training, being as fast to train as a single-component sampler (see Figure 4 and Figures 11 to 13). As with most of machine learning research, which historically started from simple algorithms that have been incrementally enhanced over time, we believe SC GFlowNets set a minimal baseline for learning a mixture of GFlowNets. In summary, our contributions are as follows.

1. We introduce an abstract framework for learning an ensemble of GFlowNets, demonstrating its usefulness in providing an unifying view of prior techniques and in designing novel algorithms.
2. We define continuously (CI) and discretely indexed (DI) GFlowNets, outlining their empirical and theoretical advantages over a single-component sampler.
3. We propose Stratum-Conditioned (SC) GFlowNets as an effective form of learning DI GFlowNets, supporting both centralized and parallel training.
4. We show that SC GFlowNets speed up convergence and mode discovery across benchmark tasks.

2 PRELIMINARIES

Definitions. We let $\mathcal{X}$ and $\mathcal{S}$ be finite spaces, with $\mathcal{S}$ corresponding to *partial realizations* of $\mathcal{X}$. We also let $s_o \in \mathcal{S}$ be the *initial state* from which any $x \in \mathcal{X}$ can be constructed. An example the reader should keep in mind is when $\mathcal{X}$ is the set of subsets of size K of a given finite set and $\mathcal{S}$ is the set of subsets of size less than K, with $s_o = \emptyset$. We call a *state graph* a directed acyclic graph having s_o as the only state without incoming edges and $x \in \mathcal{X}$ as the only states without outgoing edges, which we call *terminal*, and such that there is a path from s_o to every $x \in \mathcal{X}$ containing only states from $\mathcal{S}$.

Notationally, we let $[N] = \{1, \dots, N\}$ for each positive integer $N \in \mathbb{N}$ and $s \rightsquigarrow s'$ be the set of trajectories $\tau = \{(s, s_1), (s_1, s_2), \dots, (s_d, s')\}$ from $s \in \mathcal{S}$ to $s' \in \mathcal{S} \cup \mathcal{X}$.

GFlowNets. A GFlowNet learns to stochastically construct $x \in \mathcal{X}$ from s_o by following a policy $p_F \colon \mathcal{S} \times (\mathcal{S} \cup \mathcal{X}) \to [0, 1]$ such that $p_F(s, s') > 0$ if and only if $s \to s'$ in the state graph. The objective is for the marginal distribution of p_F over $\mathcal{X}$ to match the unnormalized distribution $R \colon \mathcal{X} \to \mathbb{R}_+$ from which we aim to sample, i.e.,

$$m_F(x) \coloneqq \sum_{\tau \in s_o \rightsquigarrow x} \prod_{(s_i, s_{i+1}) \in \tau} p_F(s_i, s_{i+1}) \coloneqq \sum_{\tau \in s_o \rightsquigarrow x} p_F(s_o, \tau) \propto R(x). \tag{1}$$

As $m_F(x)$ is generally intractable, we introduce a *backward policy* $p_B(x, \tau)$ representing the probability of having followed τ given that x was sampled. We then search for (p_F, p_B) satisfying a *consistency condition*, with a popular choice being the trajectory balance (Malkin et al., 2022, TB): $Z \cdot p_F(s_o, \tau) = R(x) \cdot p_B(x, \tau)$ and Z as a learned constant. In practice, $p_F(s, \cdot)$ is parameterized as a neural network and this search is carried out via stochastic gradient descent. Please refer to Sections A and I for a comprehensive background and recent advances on GFlowNets.

3 Learning a Mixture of GFlowNets

Simply put, the central challenges in achieving Equation (1) are the problem's *under-determination*, owning to the existence (in general) of infinitely many pairs (p_F, p_B) abiding by any given consistency condition, and *sample inefficiency*, requiring extensive state graph exploration for finding a resolutive (p_F, p_B). We will show a mixture formulation can naturally address both of them, focusing on the latter in the main text and elaborating on the former in Section C of the supplement.

Towards this objective, we develop an abstract framework for learning a mixture of GFlowNets. For starters, we let Γ be an *index space* and $\{p_F^{(\gamma)}, p_B^{(\gamma)}\}_{\gamma \in \Gamma}$ be the corresponding set of policies. Our goal is for the marginal distribution of these policies on each $x \in \mathcal{X}$, when weightedly averaged over $\gamma \in \Gamma$, to match the target $R(x)$. Formally, we assume Γ to be embedded in a topological measurable space equipped with a measure ν over its Borel σ-algebra. This allows us to set aside the specific properties of Γ, while accounting for common use cases via a unified language.

Remark 3.1 (Γ can be either discrete or continuous)**.** Common choices for Γ and ν are $\Gamma = \mathbb{R}^{d_\gamma}$ with ν as the Lebesgue measure and $\Gamma = [K]$ with ν as the counting measure (Sections C and 5).

Designing a mixture of GFlowNets. With this setup, we let p_M be a probability measure on Γ absolutely continuous with respect to ν. The subscript M indicates p_M is a *mixing (prior)* distribution over Γ. Our goal is for the marginal identity analogous to Equation (1)

$$\int_\Gamma p_M(\mathrm{d}\gamma) \sum_{\tau \in s_o \rightsquigarrow x} p_F^{(\gamma)}(s_o, \tau) \coloneqq \int_\Gamma p_M(\mathrm{d}\gamma) m_F^{(\gamma)}(x) \propto R(x) \tag{2}$$

to be satisfied. As in Equation (1), we call $m_F^{(\gamma)}(x)$ the marginal of $p_F^{(\gamma)}(s_o, \cdot)$ over $\mathcal{X}$. Under this condition, we can sample from R by first drawing $\gamma \sim p_M(\mathrm{d}\gamma)$ and then sampling $x \sim m_F^{(\gamma)}(x)$. Nevertheless, Equation (2) is doubly intractable: neither $m_F^{(\gamma)}$ nor its average under $\gamma \sim p_M$ can be computed efficiently. We circumvent this issue by introducing a *mixing (posterior) distribution* $q_M(\mathrm{d}\gamma | x)$ and rewriting Equation (2) as[1]

$$p_M(\mathrm{d}\gamma) m_F^{(\gamma)}(x) \propto R(x) q_M(\mathrm{d}\gamma | x). \tag{3}$$

The role of q_M in the equation above is akin to that of *responsibilities* in expectation-maximization algorithms (Dempster et al., 1977), representing the probability a given $x \in \mathcal{X}$ was drawn from a γ-indexed sampler. As in Section 2, we turn Equation (3) into a learning objective—referred to as *mixed trajectory balance* (MTB)—by replacing the intractable marginal with tractable policies.

Definition 3.2 (MTB condition)**.** Let $\mathfrak{g}_\gamma = (p_F^{(\gamma)}, p_B^{(\gamma)})$ and $\{\mathfrak{g}_\gamma\}_{\gamma \in \Gamma}$ be a Γ-indexed set of GFlowNets, and $p_M(\mathrm{d}\gamma)$ and $q_M(\mathrm{d}\gamma | x)$ be *prior* and *posterior* mixing distributions over Γ. Then,

$$p_M(\mathrm{d}\gamma) p_F^{(\gamma)}(s_o, \tau) = \frac{R(x)}{Z} q_M(\mathrm{d}\gamma | x) p_B^{(\gamma)}(x, \tau)$$

for each $x \in \mathcal{X}$ and $\tau \in s_o \rightsquigarrow x$ is the *mixed trajectory balance* condition, with $Z = \sum_{x \in \mathcal{X}} R(x)$.

[1] Recall $f(\gamma)\mu(\mathrm{d}\gamma) = g(\gamma)\nu(\mathrm{d}\gamma)$ means $\int_B f(\gamma)\mu(\mathrm{d}\gamma) = \int_B g(\gamma)\nu(\mathrm{d}\gamma)$ for any measurable set B.

The sufficiency of Definition 3.2 for Equation (2) is a direct consequence of (Malkin et al., 2022, Proposition 1). As it will soon become clear, it will be notationally convenient to define

$$Z(\mathrm{d}\gamma) = Z \cdot p_M(\mathrm{d}\gamma) \text{ and } R(x, \mathrm{d}\gamma) = R(x) q_M(\mathrm{d}\gamma|x)$$

as unnormalized measures over Γ and $\mathcal{X} \times \Gamma$. This allows us to rewrite MTB as

$$Z(\mathrm{d}\gamma) \cdot p_F^{(\gamma)}(s_o, \tau) = R(x, \mathrm{d}\gamma) \cdot p_B^{(\gamma)}(x, \tau),$$

resembling the familiar TB condition. We also deduce from this that other balance conditions, such as Detailed Balance (DB) (Bengio et al., 2023) and SubTB (Madan et al., 2022), remain valid in our setting. Given a function $F\colon s \in \mathcal{S} \mapsto F(s, \mathrm{d}\gamma)$ mapping s to a measure in Γ, for instance, the DB condition can be written as, for $s \in \mathcal{S}$, $s' \in \mathcal{S} \cup \mathcal{X}$, and $x \in \mathcal{X}$

$$F(s, \mathrm{d}\gamma) \cdot p_F^{(\gamma)}(s, s') = F(s', \mathrm{d}\gamma) p_B^{(\gamma)}(s', s) \text{ and } F(x, \mathrm{d}\gamma) = R(x, \mathrm{d}\gamma).$$

This raises the question of how to select both $Z(\mathrm{d}\gamma)$ and $R(x, \mathrm{d}\gamma)$, which will be addressed in Section 5. Under our abstract setting, however, we show both measures are intimately related.

Proposition 3.3 (Relationship between $Z(\mathrm{d}\gamma)$ and $R(x, \mathrm{d}\gamma)$)**.** *Let $\{\mathfrak{g}_\gamma\}_{\gamma\in\Gamma}$ and p_M and q_M be as in Definition 3.2, and $Z(\mathrm{d}\gamma)$ and $R(x, \mathrm{d}\gamma)$ be as above. Then, $Z(\mathrm{d}\gamma)$ and $R(x, \mathrm{d}\gamma)$ jointly satisfy*

$$Z(\mathrm{d}\gamma) = \sum_{x\in\mathcal{X}} R(x, \mathrm{d}\gamma) \text{ and } R(x, \mathrm{d}\gamma) = Z(\mathrm{d}\gamma) \cdot m_F^{(\gamma)}(x).$$

Under Proposition 3.3, $Z(\mathrm{d}\gamma)$ may be understood as the fraction of total probability mass (Z) being assigned to the model indexed by γ, while $x \mapsto R(x, \mathrm{d}\gamma)$ represents the (unnormalized) marginal distribution of such model over $\mathcal{X}$. This supports the interpretation of Γ-indexing as a *soft partition* of $\mathcal{X}$: each $\mathfrak{g}_\gamma$ primarily focuses on a (potentially) distinct subset of $\mathcal{X}$.

Learning a GFlowNet ensemble. We learn a MTB-abiding $\mathfrak{g}_\Gamma = (p_M, q_M, \{\mathfrak{g}_\gamma\}_{\gamma\in\Gamma})$ by optimizing

$$\mathcal{L}_{\mathrm{TB}}(\tau, \mathrm{d}\gamma; \mathfrak{g}_\Gamma) = \left(\log \frac{Z(\mathrm{d}\gamma) \cdot p_F^{(\gamma)}(s_o, \tau)}{R(x, \mathrm{d}\gamma) \cdot p_B^{(\gamma)}(x, \tau)}\right)^2 \tag{4}$$

for all τ, γ; we call $\mathcal{L}_{\mathrm{TB}}$ the *TB loss function* (Malkin et al., 2022). As an aside, we notice in Definition B.2 that similar learning objectives from Bengio et al. (2023); Deleu et al. (2024), such as SubTB and DB, can also be seamlessly extended to our setting.

From this, learning continues business as usual: we parameterize $p_F^{(\gamma)}(s, \cdot)$ and (optionally) $p_B^{(\gamma)}(s, \cdot)$ as neural networks receiving both s and γ as inputs and, defining exploratory (full support) distributions $\rho_M(\mathrm{d}\gamma)$ and $\rho^{(\gamma)}(s_o, \tau)$, we solve the stochastic program

$$\min_{\mathfrak{g}_\Gamma} \mathbb{E}_{B\sim\rho_M(\cdot), \tau\sim\rho^{(\gamma)}(s_o,\cdot)} \left[\mathcal{L}_{\mathrm{TB}}\left(\tau, B; \mathfrak{g}_\Gamma\right)\right] \tag{5}$$

via stochastic gradient descent (Jordan et al., 2024). In the following sections, we present both specific instantiations and an empirical validation of the proposed framework.

Remark 3.4 (Modularity)**.** We also notice methods for learning both p_F and ρ in Equation (5) can be directly applied to our setting (Kim et al., 2025b; Madan et al., 2025; Dall'Antonia et al., 2026a).

4 Continuously Indexed GFlowNets

The most straightforward form of Equation (5) assumes $q(\mathrm{d}\gamma|x) = p_M(\mathrm{d}\gamma)$, i.e., each component is equally likely to generate $x \in \mathcal{X}$. In this case, the ratio $Z(\mathrm{d}\gamma)/R(x,\mathrm{d}\gamma) = Z/R(x)$ becomes γ-independent, and our problem reduces to minimizing the familiar TB loss averaged over $\gamma \in \Gamma$ (Malkin et al., 2022). Although deceptively simple, this approach corresponds to *augmenting* our model with *random features* (RFs) when $\Gamma = \mathbb{R}^{d_\gamma}$. This is a well-established procedure for boosting the expressivity of graph neural networks (Abboud et al., 2021), often-used by GFlowNets in graph-structured spaces. Drawing on this, we show *continuously indexed* (CI) GFlowNets—as we call the RF-augmented sampler—can arbitrarily approximate any graph distribution.

Remark 4.1 (CI GFlowNets as Universal Approximators)**.** CI GFlowNets are strictly more expressive, in terms of which distributions they can realize, than GFlowNets (see Example C.3). In particular, when considering graph-structured tasks based on edge-additive generative processes, CI GFlowNets are *universal approximators* (Proposition C.5).

We formalize this discussion in Section C, where we also demonstrate CI GFlowNets can have a smoothing effect on the loss surface (as in Bishop, 1995), significantly reducing training instability.

5 DISCRETELY INDEXED GFLOWNETS

To start our discussion on discrete GFlowNet mixtures, we first notice that the TB loss function in Equation (4) does not impose a preference of a non-trivial solution over a collapsed sampler in which $Z(\mathrm{d}\gamma) = Z$ and $R(x, \mathrm{d}\gamma) = R(x)$ or, more generally, in which the ratio $R(x,\mathrm{d}\gamma)/Z(\mathrm{d}\gamma)$ does not depend on γ. Although a γ-independent target can exhibit beneficial properties, such as greater expressivity and a smoother loss surface (as in Section 4), this should be a design decision—achieved by setting $q_M(\mathrm{d}\gamma|x) = p_M(\mathrm{d}\gamma)$—rather than a consequence of training. In practice, learning a non-collapsed mixture from Equation (5) requires constraining the search space for the mixing distributions q_M and p_M either via soft regularization or hard constraints.

We advocate for the latter in this section, suggesting to fix Γ as a finite set and $q_M(\mathrm{d}\gamma|x)$ as a *sparse distribution* assigning zero probability to most $\gamma \in \Gamma$. We also briefly elaborate on using a countably infinite Γ when discussing Boosted GFlowNets (Dall'Antonia et al., 2026b) in Section F.

Discretely Indexed GFlowNets. We refer to the corresponding sampler as Discretely Indexed (DI) GFlowNets. As notational shortcuts, we let $\Gamma = [K]$, and write the measures Z and R and q_M as $Z(\{k\}) = Z_k$ and $R(x, \{k\}) = R_k(x)$ and $q_M(\{k\}|x) = q_k(x)$, respectively. In addition, we let $\mathbf{Z} = (Z_k)_{k\in[K]}$ and ρ_k be the exploration policy for the mixture's kth component. Our objectives are to define q_k and ρ_k; as with single-component GFlowNets, we set $\mathbf{Z}$ as a learnable parameter.

5.1 STRATUM-CONDITIONED GFLOWNETS

In this context, we propose setting $q_k(x)$ according to a hand-crafted *covering* $\{X_k\}_{k\in[K]}$ of $\mathcal{X}$: under this condition, $q_k(x)$ corresponds to a uniform distribution over the indices k for which $\mathcal{X}_k$ contains x. We call each $\mathcal{X}_k$ a *stratum* due to its construction as the pre-image of an interval with respect to a real-valued function on $\mathcal{X}$; see Remark 5.5. This section formalizes the ensuing algorithm, called Stratum-Conditioned (SC) GFlowNets, explains how it can be implemented, and describes sufficient conditions for which sampling from $\mathcal{X}_k$ is computationally tractable.

Stratum Conditioning. The first requirement for tractability is for us to know how many sets $N(x)$ in $\{\mathcal{X}_k\}_{k\in[K]}$ contain a given x, allowing for the efficient computation of $q_k(x)$. As such, we define a *tractable K-cover* of $\mathcal{X}$, denoted as $\mathrm{TC}(\mathcal{X}, K)$.

Definition 5.1 (Tractable K-Cover)**.** Let K be a positive integer and $\mathcal{X}$ be a set. We say $\mathrm{TC}(\mathcal{X}, K) = \{\mathcal{X}_k\}_{k=1}^K$ is a *Tractable K-Cover* of $\mathcal{X}$ if the following conditions are satisfied.

1. $\mathrm{TC}(\mathcal{X}, K)$ is an exact cover of $\mathcal{X}$, i.e., each $\mathcal{X}_k \subseteq \mathcal{X}$ and $\bigcup_{k=1}^K \mathcal{X}_k = \mathcal{X}$.
2. For each $x \in \mathcal{X}$, the quantity $N(x) \coloneqq \#\{k \in [K] \colon x \in \mathcal{X}_k\}$ is known[2].

The second condition allows efficient querying of $q_k(x)$, defined as $q_k(x) = N(x)^{-1}$ if $x \in \mathcal{X}_k$ and $q_k(x) = 0$ otherwise. In practice, we often let $\mathrm{TC}(\mathcal{X}, K)$ be a disjoint cover—a partition—of $\mathcal{X}$; in this case, $q_k(x) = 1$ for the only k with $x \in \mathcal{X}_k$. A corollary of Proposition 3.3 cleanly characterizes both Z_k and $R_k(x)$ as a fraction of the probability masses assigned to $\mathcal{X}_k$ and x under a TC.

Corollary 5.2 (Z_k as the mass within $\mathcal{X}_k$)**.** *Let $\mathrm{TC}(\mathcal{X}, K) = \{\mathcal{X}_k\}_{k=1}^K$ and $\mathfrak{g}_{[K]}$ be a GFlowNet ensemble satisfying the MTB condition in Definition 3.2. Then, denoting by $\mathbb{1}[\cdot]$ the indicator function,*

$$Z_k = \sum_{x\in\mathcal{X}_k} \frac{R(x)}{N(x)} \text{ and } R_k(x) = \frac{R(x)}{N(x)} \cdot \mathbb{1}[x \in \mathcal{X}_k].$$

In particular, when $N(x)$ is constant, the learned policy $p_F^{(k)}$ generates samples in proportion to the restriction of $R(x)$ to $\mathcal{X}_k$, a property that might be of independent interest when the covering has a meaningful semantic value for the problem at hand.

Remark 5.3 ($R_k(x)$ for a disjoint cover)**.** Under the conditions of Corollary 5.2, assume $N(x) = N$ (e.g., $N = 1$ for a disjoint cover). Then, $m_F^{(k)}(x) \propto R(x) \cdot \mathbb{1}[x \in \mathcal{X}_k]$.

We emphasize $\{p_F^{(k)}\}_{k\in[K]}$ are parameterizable via a shared neural network architecture receiving that receives an embedding of k as input; they need not be completely separate samplers. To understand the benefits of partitioning, we consider the LINES domain (Silva et al., 2026), described next.

[2] An *intractable* cover, in contrast, could be built upon an expensive multiclass classifier $f \colon \mathcal{X} \to [K]$.

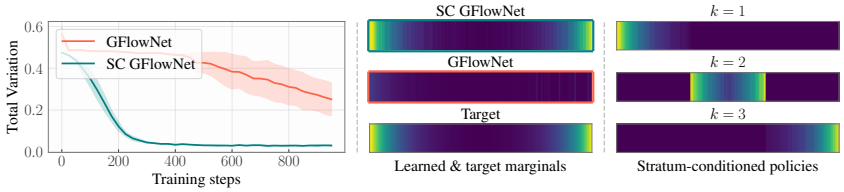


Figure 2: SC GFlowNets for the LINES environment ($N = 256$). (Left) We show the TV distance between the learned and target distributions, emphasizing the faster convergence promoted by stratum-conditioning. (Middle) The marginal distribution of both models after 1000 training steps, and the target. (Right) The marginal distribution of SC GFlowNets for each stratum $k \in \{1, 2, 3\}$.

Example 5.4 (LINES domain)**.** We define $\mathcal{S} = \{1, \dots, N\}$ and $\mathcal{X} = \{1, \dots, N\} \times \{\top\}$ with $N \geq 2$ and $\top$ indicating the state is in $\mathcal{X}$ and not in $\mathcal{S}$. The initial state is $s_o = 1$ and each transition corresponds to adding 1 to the current state if it is less than N or attaching $\top$ and terminating. Let

$$\mathcal{X}_1 = \{1, \dots, \lfloor N/3 \rfloor\},\ \mathcal{X}_2 = \{\lfloor N/3 \rfloor + 1, \dots, \lfloor 2 \cdot N/3 \rfloor\},\ \text{and}\ \mathcal{X}_3 = \{\lfloor 2 \cdot N/3 \rfloor + 1, \dots, N\}$$

be the partition. We design $p_F^{(k)}(s, \cdot)$ by masking out the terminating transition if $s \notin \mathcal{X}_k$ and masking out the successor transition if $s + 1 \notin \mathcal{X}_k$, i.e.,

$$p_F^{(k)}(s, s+1) \propto \mathbb{1}[s+1 \in \mathcal{X}_k] \cdot e^{\phi(s,k)^\top w_1} \text{ and } p_F^{(k)}(s, (s, \top)) \propto \mathbb{1}[s \in \mathcal{X}_k] \cdot e^{\phi(s,k)^\top w_2},$$

with $\phi \colon [N] \times [K] \to \mathbb{R}^d$ and $w_1, w_2 \in \mathbb{R}^d$ being learnable parameters. Figure 2 shows the error curves measured in terms of total variation distance (TV) for (i) a GFlowNet and (ii) a SC GFlowNet for the target distribution $R(x) = \exp\{-(\beta N)^{-1} \cdot \min_{y \in \{3, N-2\}} |x - y|\}$ with $N = 256$ and $\beta = 0.1$. Notably, while SC GFlowNet quickly converges to the target distribution, a GFlowNet struggles to achieve a comparable goodness-of-fit, requiring over three times more training steps.

How to design a tractable covering? A pivotal question for SC GFlowNets is how to define the mapping $\mathrm{TC}(\mathcal{X}, K)$ for each $\mathcal{X}$ and K. Although distinct problems may benefit from idiosyncratic strategies, we provide next a blueprint for designing $\mathrm{TC}(\mathcal{X}, K)$ in commonly encountered tasks.

Remark 5.5 (Modulo-based TC)**.** Example 5.4 hints on how to construct TC: we let $\mathcal{X}_k$ be the set of states whose distance to a reference point is within a prescribed interval. As most state spaces of interest lack an inherent geometric structure, however, we use a *modular function* as a general proxy for distance. To make sense of such definition, recall a *compositional* object can be described as a *multisubset* of a *component set* $\mathcal{C}$, i.e., each $s \in \mathcal{S} \cup \mathcal{X}$ can be identified with $s \subseteq \mathcal{C}$. In Example 5.4, for instance, $\mathcal{C} = \{1, \top\}$ and $s \equiv \{\{1, \dots, 1\}\}$ (s times). In this scenario, let ψ be a bounded modular function with $\psi(\{c\}) \geq 0$ for all $c \in \mathcal{C}$. We let

$$\mathcal{X}_k = \left\{ x \in \mathcal{X} \colon \frac{k-1}{K}(\psi(\mathcal{C}) + 1) \leq \psi(x) < \frac{k}{K}(\psi(\mathcal{C}) + 1) \right\}.$$

When $\psi(s) = \#s$, we call $\mathrm{TC}(\mathcal{X}, K)$ to be a *length-based covering*. In general, we let $\psi(s) = \#(s \cap r)$ for a given *reference set* $r \subseteq \mathcal{C}$; if $r = \emptyset$, the SC GFlowNet reduces to a GFlowNet.

Importantly, while easily solved for some ψ such as the length-based one in Example 5.4, we show next that the *decision problem* of inferring whether $\mathcal{X}_k$ is *reachable* from a state in $\mathcal{S}$ is NP-complete in general. This is required for masking out from $p_F^{(k)}$ transitions leading to states for which $\mathcal{X}_k$ becomes unreachable, ensuring the sampler does not become deadlocked in a zero-probability region.

Proposition 5.6 (Sampling from a general TC is NP-complete)**.** *Let $\mathrm{TC}(\mathcal{X}, K)$ be a modulo-based TC built upon a modular function ψ and $\mathcal{X}_k = \{x \in \mathcal{X} \colon \psi(x) \in [a_k, b_k)\}$ for numbers $\{(a_k, b_k)\}_{k=1}^K$. Let $s \in \mathcal{S} \subset 2^{\mathcal{C}}$. Given $k \in [K]$, we consider the following decision problem,*

$$\exists s' \in 2^{\mathcal{C} \setminus s} \text{ s.t. } s \cup s' \in \mathcal{X} \text{ and } \psi(s \cup s') \in [a_k, b_k), \tag{6}$$

i.e., whether there is a terminal $x \in \mathcal{X}_k$ reachable from $s \in \mathcal{S}$. Then, if ψ is an arbitrary modular function evaluable in polynomial time on the size of its input, Equation (6) *is NP-complete.*

We demonstrate Proposition 5.6 by reducing Equation (6) to the `SubsetSum` problem, which is NP-complete (Cormen et al., 2009). Rather than a strict limitation of SC GFlowNets, however, Proposition 5.6 should be read as a guiding design principle for $\mathrm{TC}(\mathcal{X}, K)$, suggesting there are partitions for which sampling cannot be efficiently implemented. In contrast, drawing on Example 5.4, we show there is a specific class of functions ψ for which Equation (6) is decidable in polynomial time.

Proposition 5.7 (Efficient sampling from symmetric junta-based TC)**.** *We call a function* $\psi\colon 2^{\mathcal{C}} \to \mathbb{R}$ *a* symmetric junta *if there are a* reference set $r \subseteq 2^{\mathcal{C}}$ *and a function* $\xi\colon \mathbb{Z} \to \mathbb{R}$ *for which* $\psi(x) = \xi(\#(x \cap r))$ *for all* $x \subseteq \mathcal{C}$*. We assume* ψ *can be evaluated in polynomial time. Then, if*

$$\mathcal{X}_k = \{x \in \mathcal{X}\colon \psi(x) \in [a_k, b_k)\}$$

for $k \in [K]$ *is our* $\mathrm{TC}(\mathcal{X}, K)$*, the decision problem in Equation* (6) *can be resolved in polynomial time with respect to* $\#\mathcal{C}$*. This remains true when we replace* $2^{\mathcal{C}}$ *by the bounded multisubsets of* $\mathcal{C}$*.*

As we will see in Sections 6 and G, symmetric junta-based TCs can be applied to a broad range of problems in the GFlowNet literature. Building on the solution to Equation (6), we can efficiently mask out undesirable transitions from each policy. In practice, this operation requires a negligible amount of arithmetic operations, adding an imperceptible extra cost to the sampling process. In Example 5.4, as an illustration, we solely verify whether $s + 1$ is within the boundaries of $\mathcal{X}_k$.

5.2 Complementary Characterization of SC GFlowNets

Connection to Doob's h-transform.[3] A natural question raised by our exposition is whether there is a relationship between each $p_F^{(k)}$ for $k \in [K]$. We show in this section that, given a solution p_F to the original problem of sampling from $x \in \mathcal{X}$ in proportion to $R(x)$ and *any* partition $\{\mathcal{X}_k\}_{k=1}^K$, a solution $p_F^{(k)}$ to the optimization problem in Equation (5) can be derived from a Doob's h-transform of p_F when TC is a disjoint cover. To understand this, we review Doob's h-transform of Markov chains.

Definition 5.8 (Doob's h-transform)**.** Let $\{S_t\}_{t\geq 1}$ be a Markov chain with state space $\bar{\mathcal{S}} \coloneqq \mathcal{S} \cup \mathcal{X}$ and transition kernel $\kappa\colon \bar{\mathcal{S}} \times \bar{\mathcal{S}} \to [0, 1]$. Let also $h\colon \bar{\mathcal{S}} \to \mathbb{R}$ be a *harmonic function*, i.e., $h(s) = \mathbb{E}_{s'\sim\kappa(s,\cdot)}[h(s')]$. We call $\kappa^{(h)}(s, s') = \kappa(s, s') \cdot {}^{h(s')}/_{h(s)}$ the *Doob's h-transform* of κ.

We refer the reader to (Rogers & Williams, 2000, Chapter III) for a comprehensive overview of Definition 5.8 and its applications to stochastic processes. This section shows that there is a harmonic function h_k for which $p_F^{(k)}$ is the Doob's h_k-transformation of p_F for each $k \in [K]$. To see this, we describe the generative process of a GFlowNet as a Markov chain having $\mathcal{X}$ as the absorbing set.

Proposition 5.9 (GFlowNet as an absorbing Markov chain)**.** *Let* p_F *be a GFlowNet's policy abiding by any balance condition and* κ_F *be a Markov kernel on* $\bar{\mathcal{S}}$ *defined as* $\kappa_F(s, s') = p_F(s, s')$ *if* $s \in \mathcal{S}$ *and* $\kappa_F(x, s') = \mathbb{1}[s' = x]$ *if* $x \in \mathcal{X}$ *for each* $s' \in \bar{\mathcal{S}}$*. Let* $\{S_t^{(f)}\}_{t\geq 1}$ *be a Markov chain following* κ_F*, and let* $T_{\mathcal{X}} = \min\{t\colon S_t^{(f)} \in \mathcal{X}\}$ *be its hitting time on* $\mathcal{X}$*. When* p_F *satisfies the TB,*

$$\mathbb{P}_{\kappa_F}[T_{\mathcal{X}} < \infty | s_o] = 1 \text{ and } \mathbb{P}_{\kappa_F}[S_{T_{\mathcal{X}}}^{(f)} = x \in \mathcal{X} | s_o] = \frac{R(x)}{\sum_{x\in\mathcal{X}} R(x)}$$

with $\mathbb{P}_{\kappa_F}$ *being the probability measure induced by the transition kernel* κ_F*. Equivalently, let* p_B *be a GFlowNet's backward policy and* κ_B *a Markov kernel on* $\bar{\mathcal{S}}$ *such that* $\kappa_B(s, s') = p_B(s, s')$ *for* $s \in \bar{\mathcal{S}} \setminus \{s_o\}$ *and* $s' \in \bar{\mathcal{S}}$ *and* $\kappa_B(s_o, s) = \mathbb{1}[s = s_o]$*. As such, let* $\{S_t^{(b)}\}_{t\geq 1}$ *follow* κ_B*; if*

$$T_{\{s_o\}} = \min\{t\colon S_t^{(b)} = s_o\}, \text{ then } \mathbb{P}_{\kappa_B}\left[T_{\{s_o\}} < \infty | x\right] = 1 \text{ for every } x \in \mathcal{X}.$$

Under Proposition 5.9, we show that there is a transition kernel $\kappa_F^{(k)}$ for which (i) $\mathcal{X}_k$ is the only absorbing set, (ii) each $x \in \mathcal{X}_k$ is an absorbing state being arrived at with probability proportional to $R(x)$, (iii) and that $\kappa_F^{(k)}$ is a Doob's transformation of κ_F. For this, let $\{S_t^{(f)}\}_{t\geq 1}$ be as above and

$$T_k = \min\{t\colon S_t^{(f)} \in \mathcal{X}_k\}$$

be the hitting time on $\mathcal{X}_k$. We define $h_k(s) = \mathbb{P}_{\kappa_F}[T_k = \min_{k'} T_{k'} | s]$ as the probability that the Markov chain $\{S_t^{(f)}\}_{t\geq 1}$ reaches $\mathcal{X}_k$ before any other $\mathcal{X}_{k'}$ when starting at s. The next lemma shows h_k is a harmonic function in the sense of Definition 5.8.

Lemma 5.10. *The functions* $h_k\colon \bar{\mathcal{S}} \to \mathbb{R}$*,* $k \in [K]$*, are harmonic in* $\mathcal{S}$ *with respect to* κ_F*. They also satisfy* $h_k(x) = \mathbb{1}[x \in \mathcal{X}_k]$ *for* $x \in \mathcal{X}$ *and, defining* Z_k *and* Z *as in Corollary 5.2,* $h_k(s_o) = {}^{Z_k}/_{Z}$*.*

Lemma 5.10 ensures Doob's h_k-transform of κ_F is well-defined. We show in the next proposition that the kernels resulting from such transform sample each $x \in \mathcal{X}_k$ in proportion to $R(x)$.

[3] This section is fairly technical and can be skipped by readers solely interested in our method.

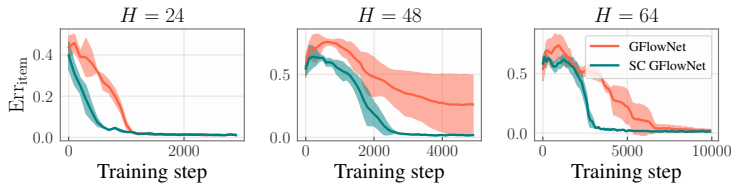


Figure 3: **Length-based SC GFlowNets trains faster than conventional GFlowNets** in the SET GENERATION domain, requiring less than half the number of iterations to achieve the same accuracy.

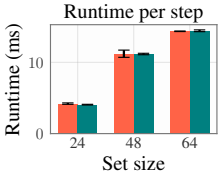


Figure 4: Average wall-clock times are similar for both samplers.

Proposition 5.11 ($p_F^{(k)}$ is a Doob's h_k-transform of p_F)**.** *Let κ_F be as in Proposition 5.9 and $h_k \colon \bar{\mathcal{S}} \to \mathbb{R}$ be the harmonic function of Lemma 5.10 for $k \in [K]$. We define $\kappa_F^{(k)}(s, s') = \kappa_F(s, s') \cdot {}^{h_k(s')}/_{h_k(s)}$ and $\{S_t^{(k)}\}_{t \geq 1}$ as a Markov chain following $\kappa_F^{(k)}$. Then, if $T_m^{(k)} = \min\{t \colon S_t^{(k)} \in \mathcal{X}_m\}$,*

$$\mathbb{P}_{\kappa_F^{(k)}}\left[T_k^{(k)} < \infty = T_m^{(k)}\right] = 1 \text{ for } m \neq k \text{ and } \mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k^{(k)}}^{(k)} = x\right] = \frac{R(x)}{\sum_{x \in \mathcal{X}_k} R(x)}$$

for all $x \in \mathcal{X}_k$. In particular, the policy function $p_F^{(k)}$ defined as $p_F^{(k)}(s, s') = \kappa_F^{(k)}(s, s')$ for $s \in \mathcal{S}$ and $s' \in \bar{\mathcal{S}}$ satisfies Equation (2)*, i.e., the marginal of $p_F^{(k)}$ on $\mathcal{X}$ matches R_k up to a constant.*

In other words, Proposition 5.11 states $p_F^{(k)}$ solves SC GFlowNet's learning problem.

5.3 SAMPLING PARTITIONS & PARALLEL LEARNING FOR SC GFLOWNETS

Although Proposition 5.11 clearly connects each $p_F^{(k)}$ to a base policy p_F, the quantity $h_k(x)$ cannot be efficiently computed in general even for known κ_F—let alone for the unknown solution to the GFlowNet's balance equations. Besides, learning h_k from Lemma 5.10 can be impractical, as it would severely increase the per-transition computational cost. In practice, we parameterize our policies as in Example 5.4 and learn them by minimizing the objective function

$$\mathbb{E}_{k \sim \rho_M,\, \tau \sim \rho_k}\left[\mathcal{L}_{\mathrm{TB}}\left(\tau, \{k\}; \mathfrak{g}_{[K]}\right)\right]. \tag{7}$$

Active Partition Sampling. Equation (7) begs the question of how to sample k from $[K]$ during training. Intuitively, we should select most often the k for which the probability mass under $\mathcal{X}_k$ is largest, while also serendipitously picking the k such that $\mathcal{X}_k$ is thought to only include low-probability states. As Z_k serves as a surrogate for the probability mass under $\mathcal{X}_k$ (Corollary 5.2), this intuition can be naturally operationalized by drawing k according to

$$k \sim \rho_M \coloneqq \mathrm{Categorical}\left((1 - \epsilon) \cdot {}^{\mathbf{Z}}/_{\mathbf{1}^\intercal \mathbf{Z}} + \epsilon \cdot K^{-1}\right)$$

with $\epsilon \in [0, 1]$ and $\mathbf{1}$ as a K-sized vector of ones. We emphasize more sophisticated approaches, such as curriculum-based sampling (Bengio et al., 2009; Laajil et al., 2025) focused on high-loss instead of high-mass components, could also be developed; however, we leave this for future research. We summarize both the training and inference procedures for SC GFlowNets in Algorithms 1 and 2.

Embarrassingly Parallel Learning. Besides being trainable as a monolithic model through active partition sampling, SC GFlowNets enable distributed learning of GFlowNets without inter-process communication. We call this *embarrassingly parallel learning*: we learn K independent models, each of which generates samples from $\mathcal{X}_k$ according to $R_k(x)$. Afterwards, during inference, we pick each model in proportion to its learned partition function to generate samples from $\mathcal{X}$. To separate SC GFlowNet from their asynchronous variants, we refer to the latter as *ASC GFlowNets*.

6 EXPERIMENTS

Our experiments evaluate the effectiveness of SC GFlowNets. We consider the tasks of SET GENERATION, HYPERGRID, and ANCESTRAL GRAPHS, representing both classical and recent GFlowNet applications with tractable assessment metrics (Malkin et al., 2022; 2023; Kim et al., 2023; 2025b), and defer LAZY RANDOM WALK and VARIABLE SELECTION, along with an evaluation of ASC GFlowNets and a comparison against Boosted GFlowNets (Dall'Antonia et al., 2026b), to Section G.

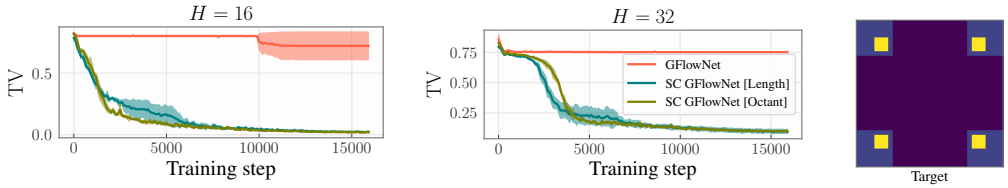
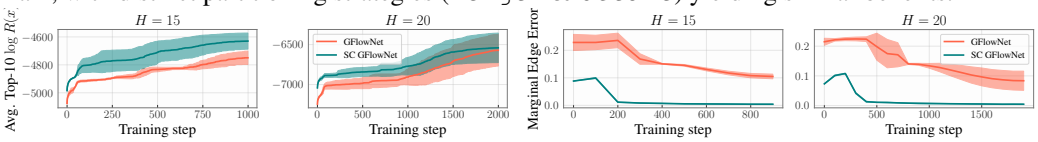


Figure 5: **SC GFlowNets converge substantially faster than GFlowNets** in the HYPERGRID domain, with distinct partitioning strategies (`Length` & `Octant`) yielding similar benefits.



(a) BIC (Equation (20)).

(b) Erdős-Renyi model.

Figure 6: **SC GFlowNets enhance exploration (left) and speed up learning (right)** for the ANCESTRAL GRAPHS domain. We denote by H the number of variables (nodes).

Set generation. We let $\mathcal{S} = \{s \subseteq [H]\}$ for some $H \in \mathbb{N}$ and $\mathcal{X} = \mathcal{S} \times \{\top\}$ for the state space. We also define $u\colon [H] \to \mathbb{R}_+$ and let $R(x) = \prod_{e \in x} u(e)$ be the target distribution, which allows the probability $\pi_{\text{item}}(e)$ of e being a member of x for $e \in [H]$ to be tractably computed. We evaluate the model via $\text{Err}_{\text{item}} = \frac{1}{2}\sum_{e \in [H]} |\hat{\pi}_{\text{item}}(e) - \pi_{\text{item}}(e)|$, with $\hat{\pi}_{\text{item}}$ as the Monte Carlo estimate of π_{item}. For SC GFlowNets, we use a length-based partitioning with $K = 3$. Results for $H \in \{24, 48, 64\}$ in Figure 3 highlight the benefits of SC GFlowNet over its unpartitioned counterpart. In addition, we notice in Figure 4 the difference in runtime for both samplers is not significant. The main reason for this is that the cost of the extra steps required by the mixture model—active partition sampling and mask computation—is dwarfed by that of policy and reward evaluation. We observed similar runtime patterns across all our experiments.

Hypergrid. The state space for HYPERGRID is $\mathcal{S} = \{0, \ldots, H-1\}^d$ and $\mathcal{X} = \mathcal{S} \times \{\top\}$. The initial state is $s_o = \mathbf{0}$ and each transition from $s \in \mathcal{S}$ either adds 1 to one of s's coordinates, respecting the boundaries of $\mathcal{S}$, or interrupts the generative process, producing $x = (s, \top)$. The target distribution, described in Section G, is shown in Figure 5. When designing SC GFlowNets, we consider both a length- ($K = 3$) and octant-based ($K = 4$) partitions, with the latter perfectly separating the four high-probability regions of $\mathcal{X}$; see Figure 27 for illustrations. Results in Figure 5 confirm SC GFlowNets's effectiveness even when using an imperfectly balanced partitioning.

Ancestral graphs. This task was studied in (da Silva et al., 2026) for addressing the problem of causal discovery under latent confounding. In a nutshell, $\mathcal{S}$ contains all *partially edge-labelled* ancestral graphs (Richardson & Spirtes, 2002, AGs) with H nodes, while $\mathcal{X}$ contains all *fully edge-labelled* AGs. The generative process starts at a graph with unlabelled edges, and each transition labels a node pair with either $\emptyset$, $\rightarrow$, $\leftarrow$, or $\leftrightarrow$; $\emptyset$ represents the absence of an edge. The objective is to generate high-scoring AGs x according to the Bayesian information criterion (BIC) based on a dataset modelled as a linear Gaussian structural equation model (Drton & Richardson, 2004); we set $R(x) = \exp\{-\text{BIC}(x)\}$. As the distributional correctness of such method cannot be tractably measured for all but tiny H, we also consider an Erdős-Renyi model, for which edge marginals can be efficiently computed using Robinson's formula (Robinson, 1973) and compared to that learned by our samplers (see Section G). In both cases, we partition $\mathcal{X}$ according to the number of non-$\emptyset$ edges for SC GFlowNets. Figure 6 shows that our mixture model improves both exploration and convergence.

7 CONCLUSIONS

The central questions we addressed in this work are why and how to learn a mixture of GFlowNets. We first developed a general-purpose framework for answering them, outlining the capabilities and limitations of such approach. In particular, we showed SC GFlowNets, which learn an ensemble of samplers restricted to subsets of the state space, provide substantial empirical benefits at a negligible computational cost. Future research should investigate better ways of operationalizing our framework and, specifically, develop heuristics for automatic and tractable partitioning of the state space.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. 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.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *NeurIPS (NeurIPS)*, 2021.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research (JMLR)*, 2023.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

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

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Oussama Boussif, Léna Néhale Ezzine, Joseph D Viviano, Michał Koziarski, Moksh Jain, Nikolay Malkin, Emmanuel Bengio, Rim Assouel, and Yoshua Bengio. Action abstractions for amortized sampling, 2024. URL https://arxiv.org/abs/2410.15184.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

Leo Brunswic, Yinchuan Li, Yushun Xu, Yijun Feng, Shangling Jui, and Lizhuang Ma. A theory of non-acyclic generative flow networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):11124–11131, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i10.28989. URL http://dx.doi.org/10.1609/aaai.v38i10.28989.

Anthony Caterini, Rob Cornish, Dino Sejdinovic, and Arnaud Doucet. Variational inference with continuously-indexed normalizing flows, 2021. URL https://arxiv.org/abs/2007.05426.

Binqi Chen, Hongjun Ding, Ning Shen, Jinsheng Huang, Taian Guo, Luchen Liu, and Ming Zhang. Alphasage: Structure-aware alpha mining via gflownets for robust exploration, 2026. URL https://arxiv.org/abs/2509.25055.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009. ISBN 9780262033848.

Rob Cornish, Anthony L. Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows, 2021. URL https://arxiv.org/abs/1909.13833.

Tiago da Silva, Bruna Bazaluk, Eliezer de Souza da Silva, António Góis, Salem Lahlou, Dominik Heider, Samuel Kaski, Diego Mesquita, and Adèle Helena Ribeiro. Expert-aided causal discovery of ancestral graphs. *Information Sciences*, 756:123816, 2026. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2026.123816.

Pedro Dall'Antonia, Tiago da Silva, Daniel Csillag, Salem Lahlou, and Diego Mesquita. Avoid what you know: Divergent trajectory balance for gflownets, 2026a. URL https://arxiv.org/abs/2602.17827.

Pedro Dall'Antonia, Tiago da Silva, Daniel Augusto de Souza, César Lincoln C. Mattos, and Diego Mesquita. Boosted gflownets: Improving exploration via sequential learning, 2026b.

Frederica Darema, David A. George, V. Alan Norton, and Gregory F. Pfister. A single-program-multiple-data computational model for EPEX/FORTRAN. *Parallel Computing*, 7(1):11–24, 1988. doi: 10.1016/0167-8191(88)90095-4.

Tristan Deleu and Yoshua Bengio. Generative flow networks: a markov chain perspective, 2023. URL https://arxiv.org/abs/2307.01422.

Tristan Deleu, António Góis, Chris Chinenye Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *UAI*, 2022.

Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint Bayesian inference of graphical structure and parameters with a single generative flow network. In *Advances in Neural Processing Systems (NeurIPS)*, 2023.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments, 2024. URL https://arxiv.org/abs/2402.10309.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

Mathias Drton and Thomas S. Richardson. Iterative conditional fitting for gaussian ancestral graph models. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pp. 130–137. AUAI Press, 2004.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, May 1998.

Edward I. George and Robert E. McCulloch. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993. doi: 10.2307/2290777.

Timofei Gritsaev, Nikita Morozov, Sergey Samsonov, and Daniil Tiapkin. Optimizing backward policies in gflownets via trajectory likelihood maximization, 2025. URL https://arxiv.org/abs/2410.15474.

Abhi Gupta, Polina Barabanshchikova, Vikas K Garg, Samuel Kaski, and Tommi Jaakkola. Divide-and-denoise: A game-theoretic method for fairly composing diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9voQUicsc2.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2nd edition, 2009.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2026. URL http://github.com/google/flax.

Matthew D. Hoffman, David M. Blei, and Francis Bach. Online learning for latent dirichlet allocation. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23, pp. 856–864. Curran Associates, Inc., 2010. URL https://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.

Edward J. Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, and et al. Amortizing intractable inference in large language models, 2023.

Rui Hu, Yifan Zhang, Zhuoran Li, and Longbo Huang. Beyond squared error: Exploring loss design for enhanced training of generative flow networks, 2026. URL https://arxiv.org/abs/2410.02596.

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

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Mohammad Emtiyaz Khan and Didrik Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models, 2018. URL https://arxiv.org/abs/1807.04489.

Hohyun Kim, Seunggeun Lee, and Min hwan Oh. Symmetry-aware gflownets, 2025a. URL https://arxiv.org/abs/2506.02685.

Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search gflownets. *arXiv preprint arXiv:2310.02710*, 2023.

Minsu Kim, Joohwan Ko, Taeyoung Yun, Dinghuai Zhang, Ling Pan, Woochang Kim, Jinkyoo Park, Emmanuel Bengio, and Yoshua Bengio. Learning to scale logits for temperature-conditional gflownets, 2024. URL https://arxiv.org/abs/2310.02823.

Minsu Kim, Sanghyeok Choi, Taeyoung Yun, Emmanuel Bengio, Leo Feng, Jarrid Rector-Brooks, Sungsoo Ahn, Jinkyoo Park, Nikolay Malkin, and Yoshua Bengio. Adaptive teachers for amortized samplers. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=BdmVgLMvaf.

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

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Aya Laajil, Abduragim Shtanchaev, Sajan Muhammad, Eric Moulines, and Salem Lahlou. Curriculum-augmented gflownets for mrna sequence generation, 2025. URL https://arxiv.org/abs/2510.03811.

Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-García, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18269–18300. PMLR, 2023.

Sida Li, Ioana Marinescu, and Sebastian Musslick. Gfn-sr: Symbolic regression with generative flow networks, 2023. URL https://arxiv.org/abs/2312.00396.

Jun S Liu and Jun S Liu. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

Sulin Liu, Peter J. Ramadge, and Ryan P. Adams. Generative marginalization models, 2024. URL https://arxiv.org/abs/2310.12920.

Francesco Locatello, Rajiv Khanna, Joydeep Ghosh, and Gunnar Ratsch. Boosting variational inference: an optimization perspective. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 464–472. PMLR, 09–11 Apr 2018.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, 2022.

Kanika Madan, Alex Lamb, Emmanuel Bengio, Glen Berseth, and Yoshua Bengio. Towards improving exploration through sibling augmented GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HH4KWP8RP5.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlownets. In *NeurIPS (NeurIPS)*, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Nikita Morozov, Ian Maksimov, Daniil Tiapkin, and Sergey Samsonov. Revisiting non-acyclic gflownets in discrete environments, 2025. URL https://arxiv.org/abs/2502.07735.

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

W. Neiswanger, C. Wang, and E. P. Xing. Asymptotically exact, embarrassingly parallel MCMC. In *UAI*, 2014.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. *arXiv preprint arXiv:2302.01687*, 2023a.

Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative augmented flow networks. In *International Conference on Learning Representations (ICLR)*, 2023b.

Mohit Pandey, Gopeshh Subbaraj, and Emmanuel Bengio. Gflownet pretraining with inexpensive rewards. *arXiv preprint arXiv:2409.09702*, 2024.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning, 2022. URL https://arxiv.org/abs/1712.00378.

Thomas Richardson and Peter Spirtes. Ancestral graph markov models. *The Annals of Statistics*, 30 (4):962–1030, 2002.

Robert W. Robinson. Counting labeled acyclic digraphs. In Frank Harary (ed.), *New Directions in the Theory of Graphs*, pp. 239–273. Academic Press, New York, 1973. ISBN 978-0-12-324255-6.

L. C. G. Rogers and David Williams. *Diffusions, Markov Processes, and Martingales: Volume 1, Foundations*. Cambridge Mathematical Library. Cambridge University Press, 2nd edition, 2000. ISBN 9780521775946.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 2021.

Jürgen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Networks*, 127:58–66, 2020.

Max W. Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International Conference on Machine Learning*, 2023.

Tiago Silva, Rodrigo Barreto Alves, Eliezer de Souza da Silva, Amauri H Souza, Vikas Garg, Samuel Kaski, and Diego Mesquita. When do GFlownets learn the right distribution? In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=9GsgCUJtic.

Tiago Silva, Amauri H Souza, Omar Rivasplata, Vikas Garg, Samuel Kaski, and Diego Mesquita. Generalization and distributed learning of GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=PJNhZoCjLh.

Tiago Silva, Esmeralda S. Whitammer, and Salem Lahlou. Path-dependent discrete amortized inference. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=viJvYPJHIj.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry Vetrov. Generative flow networks as entropy-regularized rl, 2024.

Daniil Tiapkin, Artem Agarkov, Nikita Morozov, Ian Maksimov, Askar Tsyganov, Timofei Gritsaev, and Sergey Samsonov. gfnx: Fast and scalable library for generative flow networks in jax, 2025. URL https://arxiv.org/abs/2511.16592.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, and Emmanuel Bengio others. Amortizing intractable inference in diffusion models for vision, language, and control, 2024. URL https://arxiv.org/abs/2405.20971.

Joseph D. Viviano, Omar G. Younis, Sanghyeok Choi, Victor Schmidt, Yoshua Bengio, and Salem Lahlou. torchgfn: A pytorch gflownet library, 2026. URL https://arxiv.org/abs/2305.14594.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR)*, 2019.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International conference on machine learning*. PMLR, 2018.

Xuan Yu, Xu Wang, Rui Zhu, Yudong Zhang, and Yang Wang. Exploring multiple high-scoring subspaces in generative flow networks, 2026a. URL https://arxiv.org/abs/2602.11491.

Xuan Yu, Xu Wang, Rui Zhu, Yudong Zhang, and Yang Wang. Partial gflownet: Accelerating convergence in large state spaces via strategic partitioning, 2026b. URL https://arxiv.org/abs/2602.11498.

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

Ming Yang Zhou, Zichao Yan, Elliot Layne, Nikolay Malkin, Dinghuai Zhang, Moksh Jain, Mathieu Blanchette, and Yoshua Bengio. PhyloGFN: Phylogenetic inference with generative flow networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Heiko Zimmermann, Fredrik Lindsten, Jan-Willem van de Meent, and Christian A Naesseth. A variational perspective on generative flow networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=AZ4GobeSLq.

A GFlowNets: Background & Challenges

We provide a detailed background on GFlowNets, and review current challenges in effectively training these samplers, explaining how they can be naturally addressed by our mixture framework.

A.1 Background

This section reviews prior work on learning GFlowNets for discrete distributions, following Malkin et al. (2023); Lahlou et al. (2023), to which we refer for a comprehensive overview.

GFlowNets. Let $\mathcal{X}$ be a finite set; we say $\mathcal{X}$ is a *compositional space* if there is a state graph (SG) $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ with vertices $\mathcal{S} \supset \mathcal{X}$ and edges $\mathcal{E}$ having the following properties.

1. $\mathcal{G}$ is a directed acyclic graph (DAG) with a single source, s_o, and with $\mathcal{X}$ as the set of sink vertices. We call s_o the *initial state*.
2. There is a (possibly non-unique) path from from each s_o to each $\mathcal{S} \setminus \{s_o\}$.

We further assume, for computational reasons, that the number of children of each $s \in \mathcal{S}$ on $\mathcal{G}$ and the maximum trajectory length in $\mathcal{G}$ are exponentially smaller than $|\mathcal{X}|$. (Otherwise, every finite set $\mathcal{X}$ would be compositional by considering $\mathcal{S} = \{s_o\} \cup \mathcal{X}$ and $\mathcal{E} = \{s_o \to x \colon x \in \mathcal{X}\}$). Our objective is to generate $x \in \mathcal{X}$ in proportion to a positive function $R \colon \mathcal{X} \to \mathbb{R}_+$.

Sampling as learning. Towards this objective, we define $p_F \colon \mathcal{S} \times (\mathcal{S} \cup \mathcal{X}) \to [0, 1]$ (resp. $p_B \colon (\mathcal{S} \cup \mathcal{X}) \times \mathcal{S} \to [0, 1]$) as the *forward* (resp. *backward*) policy on $\mathcal{G}$ representing the probability $p_F(s, s')$ of transitioning from s to a child s' of s. The reason for p_B's existence will become clearer shortly. We also define $s \rightsquigarrow s'$ as the set of trajectories in $\mathcal{G}$ from s to s'. Our objective is for the marginal probability of p_F over $\mathcal{X}$, when starting at s_o, to match R, i.e.,

$$m_F(x) \coloneqq \sum_{\substack{\tau \in s_o \rightsquigarrow x \\ \tau = \{(s_o, s_1), \ldots, (s_K, x)\}}} \prod_{(s,s') \in \tau} p_F(s, s') \coloneqq \sum_{\tau \in s_o \rightsquigarrow x} p_F(s_o, \tau) \propto R(x). \tag{8}$$

The trajectory length K in the equation above need not be the same for each τ. Importantly, we emphasize this equation merely integrates out from p_F intermediate variables from the joint distribution $(s_o, \ldots, x)$ over trajectories. Due to Equation (8)'s intractability, we estimate the summation above using p_B as a proposal for an importance-sampling-like scheme, that is, denoting $Z = \sum_{x \in \mathcal{X}} R(x)$,

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

with $p_B(x, \tau) = \prod_{(s,s') \in \tau} p_B(s', s)$. The condition ${}^{p_F(\tau|s_o)}/_{p_B(\tau|x)} = {}^{R(x)}/_{Z}$ is known as *trajectory balance* (TB) Malkin et al. (2022). To search for policies satisfying it, we parameterize p_F and (optionally) p_B as neural networks and solve the stochastic program

$$\min_{p_F, p_B} \mathcal{L}(p_F, p_B | \rho) \coloneqq \mathbb{E}_{\tau \sim \rho} \left(\log \frac{Z \cdot p_F(\tau | s_o)}{p_B(\tau | x) \cdot R(x)} \right)^2 \tag{9}$$

via stochastic gradient descent. As the reader may have noticed, p_B was introduced arbitrarily; in general, there are infinitely many pairs (p_F, p_B) satisfying the TB condition, a fact that will come up later in Section A.2 when we discuss mixtures of amortized samplers. Additional learning objectives, with distinct parameterizations and properties, have been proposed and extensively studied (Madan et al., 2022; Malkin et al., 2023; Tiapkin et al., 2024; Pan et al., 2023a; Hu et al., 2026). In practice, we let θ be the (joint) parameters of p_F, p_B, and $\log Z$, $\{\alpha_t\}_{t \geq 0}$ be a sequence of step sizes, and iteratively minimize

$$\theta^{(t)} = \min_{\theta} \theta^\intercal \nabla_{\theta^{(t-1)}} \hat{\mathcal{L}}(\theta | \theta^{(t-1)}) + \frac{1}{2\alpha_t} \|\theta - \theta^{(t-1)}\|_2^2$$

(or a preconditioned variant of it (Khan & Nielsen, 2018)), with $\hat{\mathcal{L}}(\theta | \theta^{(t-1)})$ as an Monte Carlo estimate of $\mathcal{L}(p_F, p_B | \rho^{(t-1)})$ when both p_F and p_B are instantiated according to θ and $\rho^{(t-1)}$ as a function of $\theta^{(t-1)}$ (Tiapkin et al., 2024; Kim et al., 2025b; Madan et al., 2025; Gritsaev et al., 2025; Dall'Antonia et al., 2026a). For example, the popular ϵ-greedy exploration (Bengio et al., 2021; Malkin et al., 2022; Pan et al., 2023b; Zhou et al., 2024; Kim et al., 2025a; Dall'Antonia et al., 2026b) uses $\rho^{(t-1)}(s, \cdot) = (1-\epsilon) \cdot p_F^{(t-1)}(s, \cdot) + \epsilon \cdot p_U(s, \cdot)$ with $p_U(s, \cdot)$ as the policy function which selects the children of s uniformly at random and $p_F^{(t-1)}(s, \cdot)$ as the p_F corresponding to $\theta^{(t-1)}$.

A.2 Challenges in training GFlowNets

To further motivate our work, we briefly describe current challenges in learning GFlowNets, and later argue they can be partly addressed through a mixture formulation of the sampling problem.

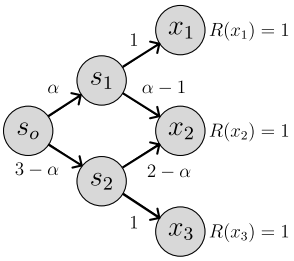


Figure 7: Any $\alpha \in [1, 2]$ solves the assignment problem in this state graph. The solution corresponding to uniform p_B is $\alpha = 1.5$.

The first challenge is that the optimization problem described in Equation (9) is *underdetermined*, as there is no pair (p_F, p_B) uniquely satisfying the TB condition in general (unless there is a single trajectory from s_o to each x, as in autoregressive sequence generation tasks Jain et al. (2022)). This is illustrated in Figure 7, which displays a family of p_F for which the marginal constraint in Equation (8) is satisfied, highlighting the existence of a *continuum* of solutions to the sampling problem.

Challenge A.1 (Underdetermination)**.** There are infinitely many pairs (p_F, p_B) for which TB is satisfied and the marginal of p_F on $\mathcal{X}$ matches R, and it is unclear how to choose between them.

A common choice, inspired by the maximum entropy principle (Jaynes, 2003), is to set $p_B(s, \cdot)$ as a uniform probability distribution over the parents of s in the SG (Shen et al., 2023). Alternatively, both p_F and p_B can be jointly learned (Gritsaev et al., 2025). However, deriving a p_B for which finding the corresponding p_F is easiest is not possible in general, and it is unclear which pair (p_F, p_B) does gradient-based learning induce. We illustrate this in Figure 8 for the HYPERGRID domain, showing the variability of $\mathrm{KL}(p_B(s, \cdot)||p_U(s, \cdot))$ averaged over states $s \in \mathcal{S} \setminus \{s_o\}$ during training for 5 independent runs (the highlighted curve represents per-step averages). As we can observe, the learned p_B vary drastically solely by modifying the pseudorandom seed for sample generation—while maintaining both the parameter initialization and optimizer unchanged.

A mixture model, as we will show, naturally addresses Challenge A.1 by learning a *family of backward policies*—hence avoiding strict commitment to a single, possibly unknown p_B. However, even when p_B is uniquely defined—e.g., in autoregressive sequence generation (Jain et al., 2022)—the underlying neural network might be incapable of finding an approximation to solution of the stochastic program in Equation (9). This problem, known as near state aliasing (Pardo et al., 2022), is particularly harmful for graph-structured domains (Silva et al., 2025a; Kim et al., 2025a).

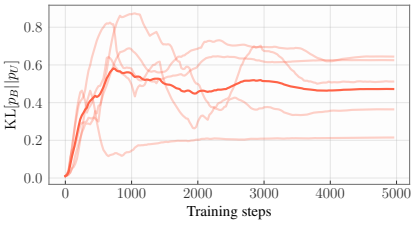


Figure 8: KL between p_B and p_U for the HYPERGRID domain.

Challenge A.2 ((Near) state aliasing)**.** A solution (p_F, p_B) to Equation (9) might not be (accurately) realizable by the chosen family of parametric models.

In Section C, we show how our method alleviates it by establishing a relationship between Random Network Initialization (RNI) in Graph Neural Networks (GNNs) and mixture modelling. Even an universal approximator, however, may struggle to search for a compatible pair (p_F, p_B) simply due to the sheer size of the state space. The main reason for this is that, to accurately approximate the target distribution, the model has to reach its high-probability regions during training Kim et al. (2023; 2025b); Dall'Antonia et al. (2026a), an obstacle that typical ϵ-greedy strategies fail to overcome.

Challenge A.3 (Insufficient exploration)**.** Accurately learning (p_F, p_B) requires visiting the target R's high-probability regions during training, which are often diverse and sparsely distributed.

To address Challenge A.3, recent works have proposed using boosting-style techniques (Dall'Antonia et al., 2026b) and artificial curiosity-inspired Schmidhuber (2020) *exploratory* models—which are discarded after training. These approaches, however, increase the per-step computational cost of conventional samplers by severalfold, requiring the learning and evaluation of one Kim et al. (2025b); Madan et al. (2025); Dall'Antonia et al. (2026a) or many additional models Silva et al. (2025b); Dall'Antonia et al. (2026b) throughout learning and inference. In contrast, in Section 6, we showed improved exploration is achievable by a *mixture* of GFlowNets—without relying on additional, independently parameterized costly models.

B MIXTURE OF GFLOWNETS

We outline additional results on the learning of a mixture of GFlowNets: an extension of Madan et al. (2022)'s SubTB condition and the associated family of loss functions. We start with the former.

Proposition B.1 (SubTB). *For each $s \in \mathcal{S} \cup \mathcal{X}$, let $F(s, \mathrm{d}\gamma)$ be a measure on Γ such that $F(s, \mathrm{d}\gamma) = R(x, \mathrm{d}\gamma)$ for $x \in \mathcal{X}$. We call F a* flow function. *If, for $\tau \in s \rightsquigarrow s'$,*

$$F(s, \mathrm{d}\gamma) p_F^{(\gamma)}(s, \tau) = F(s', \mathrm{d}\gamma) p_B^{(\gamma)}(s', \tau),$$

then Equation (2) *is satisfied. In addition, for each $s \in \mathcal{S}$,*

$$Z(\mathrm{d}\gamma) = F(s_o, \mathrm{d}\gamma) \text{ and } F(s, \mathrm{d}\gamma) = \sum_{x \in \mathcal{X}} \sum_{\tau \in s \rightsquigarrow x} R(x, \mathrm{d}\gamma) p_B^{(\gamma)}(x, \tau).$$

We provide a demonstration for this and every other statement in the supplement in Section H, jointly with the statements in the main text. From Proposition B.1, the usual loss functions encountered in the GFlowNet literature can be readily derived.

Definition B.2 (SubTB & DB & TB losses). Let $Z(\cdot)$ and $F(s, \cdot)$ and $R(x, \cdot)$ be measures on Γ, as in Proposition B.1, for $s \in \mathcal{S}$ and $x \in \mathcal{X}$, and $p_F^{(\gamma)}$ and $p_B^{(\gamma)}$ be Γ-indexed policies. We concisely let $\mathfrak{g}_\Gamma = (Z, F, R, \{p_F^{(\gamma)}, p_B^{(\gamma)}\}_{\gamma \in \Gamma})$. Given $\tau = (s_o, \ldots, s_N)$ with $s_N = x \in \mathcal{X}$, we define $\tau_{i:j} = (s_i, s_{i+1}, \ldots, s_j)$ and

$$\mathcal{L}_{\mathrm{SubTB}}(\tau, \mathrm{d}\gamma; \mathfrak{g}_\Gamma) = \sum_{0 \le i < j \le N} \alpha_{ij} \left(\log \frac{F(s_i, \mathrm{d}\gamma) p_F^{(\gamma)}(s_i, \tau_{i:j})}{F(s_j, \mathrm{d}\gamma) p_B^{(\gamma)}(s_j, \tau_{j:i})} \right)^2$$

as the ***SubTB*** loss, with $(\alpha_{ij})_{i,j=0}^N$ as positive numbers with $\sum_{j>i} \alpha_{ij} = 1$. Similarly,

$$\mathcal{L}_{\mathrm{DB}}(\tau, \mathrm{d}\gamma; \mathfrak{g}_\Gamma) = \frac{1}{N} \sum_{0 \le i \le N-1} \left(\log \frac{F(s_i, \mathrm{d}\gamma) p_F^{(\gamma)}(s_i, s_{i+1})}{F(s_{i+1}, \mathrm{d}\gamma) p_B^{(\gamma)}(s_{i+1}, s_i)} \right)^2$$

is the ***DB*** loss. In conclusion, we let the ***TB*** loss be

$$\mathcal{L}_{\mathrm{TB}}(\tau, \mathrm{d}\gamma; \mathfrak{g}_\Gamma) = \left(\frac{Z(\mathrm{d}\gamma) p_F^{(\gamma)}(s_o, \tau)}{R(x, \mathrm{d}\gamma) p_B^{(\gamma)}(x, \tau)} \right)^2.$$

C CONTINUOUSLY INDEXED GFLOWNETS

In this section, we elaborate on the design of a continuously indexed mixture of GFlowNets. As we will show, this approach can drastically boost the expressivity of GNN-parameterized samplers (Kim et al., 2025a) without requiring expensive heuristics such as look-ahead search, as proposed in (Silva et al., 2025a). In doing so, we designedly use a *collapsed* mixing posterior distribution—$q(\mathrm{d}\gamma|x) = q(\mathrm{d}\gamma)$ for every $x \in \mathcal{X}$—an approach we refer to as *random features* (RFs) for GFlowNets. The reason for this is that either $Z(\mathrm{d}\gamma)$ or $q(\mathrm{d}\gamma|x)$ are generally intractable measures when Γ is uncountable, as we also demonstrate.

Continuous mixtures of GFlowNets. We henceforth consider $\Gamma = \mathbb{R}^d$ and ν as the Lebesgue measure, and we assume both $q_M(\mathrm{d}\gamma|x)$ (resp. $R(x, \mathrm{d}\gamma)$, $F(s, \mathrm{d}\gamma)$) and $p_M(\mathrm{d}\gamma)$ (resp. $Z(\mathrm{d}\gamma)$) are absolutely continuous with respect to it. As such, we write $R(x, \gamma)$ and $F(s, \gamma)$ and $Z(\gamma)$ for their corresponding densities. It is an obvious consequence of Proposition B.1 that the balance conditions can be written as a function of these densities instead of the previously introduced measures. We highlight this in the following corollary.

Remark C.1 (SubTB). When $\Gamma = \mathbb{R}^d$ and $F(s, \mathrm{d}\gamma)$ and $Z(\mathrm{d}\gamma)$ are absolutely continuous with respect to the base Lebesgue measure ν, the SubTB condition is

$$F(s, \gamma) p_F^{(\gamma)}(s, \tau) = F(s', \gamma) p_B^{(\gamma)}(s', \tau)$$

for each $s, s' \in \mathcal{S} \cup \mathcal{X}$ and $\tau \in s \rightsquigarrow s'$ and ν-almost surely for $\gamma \in \Gamma$.

As per Remark C.1, we learn $F(s, \gamma)$ as a real-valued neural network-parameterized positive function receiving both $s \in \mathcal{S}$ and $\gamma \in \Gamma$ as inputs; recall $F(x, \gamma) = R(x, \gamma)$ for $x \in \mathcal{X}$. The core issue in doing so, however, is that we may not be able to sample from the prior mixing distribution during inference when the function $Z(\gamma)$ is defined arbitrarily. Conversely, if we let $p_M(\gamma)$ have a simple form (e.g., Gaussian) and $q_M(\gamma|x)$ be flexibly learned, the latter distribution may collapse into the former. This can be understood via the following corollary of Proposition B.1.

Corollary C.2 (Relationship between p_M and q_M)**.** *Assume $p_M, q_M, p_F^{(\gamma)}, p_B^{(\gamma)}$ satisfy the SubTB condition in Remark C.1. Then,*

$$p_M(\gamma) = \sum_{x \in \mathcal{X}} \frac{R(x)}{Z} \cdot q_M(\gamma|x) \text{ and } q_M(\gamma|x) = \frac{Z \cdot p_M(\gamma) \cdot m_F^{(\gamma)}(x)}{R(x)}.$$

The first equation in Corollary C.2 shows that, if $q_M(\gamma|x)$ has a tractable, non-trivial form, then the solution for $p_M(\gamma)$ is a combinatorial mixture of all $q_\gamma(\gamma|x)$'s for $x \in \mathcal{X}$ weighted by $R(x)$, which is the distribution we are aiming to sample from in the first place. The second equation, on the other hand, suggests that if $p_F^{(\gamma)}(s, \cdot)$ is too flexible a function of γ and s—able to perfectly mimic the target $R(x)$—then the optimal form for $q_M(\gamma|x)$ reduces to $p_M(\gamma)$. As we also discussed in Section 5, reduction is not necessarily undesirable, as we will show later; however, we believe a x-independent q_M should be an algorithmic design decision rather than an implicit consequence of training. Should the goal be to have meaningfully different $p_F^{(\gamma)}$ for each γ with the objective of, for instance, improving exploration, Corollary C.2 suggests that sensible structural constraints should be imposed on the relationship between $p_F^{(\gamma)}$ and γ, a venue we pursued in Section 5.

Random Features (RFs) for GFlowNets. In this scenario, we describe the potential benefits of setting $q_M(\gamma|x) = p_M(\gamma)$ by connecting the ensuing framework with the theory of RFs for GNNs Abboud et al. (2021); Sato et al. (2021). We first recall the expressivity limitations of message passing GNN-parameterized GFlowNets in the following example.

Example C.3 (Limitations of GNN-based GFlowNets)**.** We consider the following state graph (Figure 9) with initial state s_o being followed by x_1 and x_2, each of them a 6-node graph. We notice that x_1 and x_2 correspond to x_o with additional edges (a, b) and (a, c), respectively, and we write $x_1 = s_o \cup \{(a, b)\}$ and $x_2 = s_o \cup \{(a, c)\}$. This is the simplest setting a GFlowNet can be applied to.

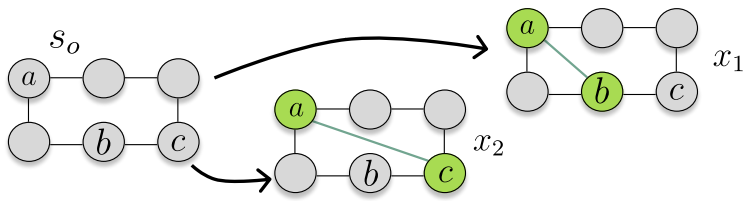


Figure 9: State graph for Example C.3.

As in (Silva et al., 2025a), we let $h\colon (\mathcal{G}, n) \mapsto \mathbb{R}^d$ be a 1-WL GNN Kipf & Welling (2016); Xu et al. (2019) yielding a representation for a node n in a graph $\mathcal{G}$ and $\sigma\colon \mathbb{R}^d \to \mathbb{R}$ be a learnable real-valued function. We parameterize the policy function according to

$$p_F(s, s \cup \{(n_1, n_2)\}) \propto \exp\{\sigma(h(s, n_1) + h(s, n_2))\} \tag{10}$$

for nodes n_1, n_2 such that there is no edge between n_1, n_2 in s, and 0 otherwise. Under these conditions, unless the target R satisfies $R(x_1) = R(x_2)$, there is no instantiation of p_F for which the marginal distribution of p_F over $\{x_1, x_2\}$ matches R—as h cannot distinguish node b from node c.

When augmenting h with random node-wise features γ, however, the policy function in Equation (10) acquires the capability of approximating *any target* with high-probability over the draw of γ. The intuition is that a RF-augmented 1-WL GNN will only be unable to separate nodes b and c in Figure 9 if their corresponding features match—an event of null probability since Γ is continuous.

Example C.4 (RFs boost the expressivity of GNN-based GFlowNets)**.** We let $h\colon (\mathcal{G}, n, \gamma_n) \mapsto \mathbb{R}^d$ be a RF-augmented 1-WL GNN with $\gamma_n \sim \mathcal{N}(0, 1)$. Then, if

$$p_F^{(\gamma)}(s, s \cup \{(n_1, n_2)\}) \propto \exp\{\sigma(h(s, n_1, \gamma_{n_1}) + h(s, n_2, \gamma_{n_2}))\} \tag{11}$$

as in Equation (10), it follows from (Abboud et al., 2021, Theorem 1) that for each $\epsilon > 0$ and $\delta \in (0, 1)$ there is a GNN h such that p_F approximates a given target with accuracy ϵ and probability $1 - \delta$ over the drawing of γ.

More broadly, we show in the next that a GFlowNet parameterized as in Equation (11) can arbitrarily approximate with high probability any given distribution defined on an edge-additive generative process for graph-structured objects. For this, we define $\mathrm{TV}(p, q)$ as the total variation distance between (possibly unnormalized) probability distributions p and q.

Proposition C.5 (RFs GNN-based GFlowNets are Universal Approximators)**.** *Let $\{(p_F^{(\gamma)}, p_B^{(\gamma)})\}_{\gamma\in\Gamma}$ be a GFlowNet augmented with RFs $\gamma \in \Gamma$ and defined over a generative process over graphs such that each transition corresponds to either adding an edge or stopping. We define the following hypothesis classes.*

1. $\mathcal{H}_{\mathrm{t}} = \{h\colon (\mathcal{G}, n, \gamma_n) \mapsto \mathbb{R}^d, \sigma\colon \mathbb{R}^d \to \mathbb{R}, d \in \mathbb{N}\}$ *containing 1-WL GNNs h and a real-valued function σ.*
2. $\mathcal{H}_{\mathrm{s}} = \{\xi\colon (\mathcal{G}, \{\mathbf{x}_n\colon n \in \mathcal{G}\}) \mapsto y\}$ *containing readout functions mapping a graph and its node embeddings to $y \in \mathbb{R}$.*

We parameterize $p_F^{(\gamma)}$ for edge addition, respectively, as in Equation (11)*, assigning also a probability proportional to $\exp\{\xi(s, \{h(s, n)\colon n \in s\})\}$ for interrupting the generation at $s \in \mathcal{S}$, and fix $p_B^{(\gamma)}$. Then, for any target distribution R, there is a $h \in \mathcal{H}_{\mathrm{t}}$ and a $\xi \in \mathcal{H}_{\mathrm{s}}$ such that the marginal distribution $m_F^{(\gamma)}$ satisfies $\mathrm{TV}(m_F^{(\gamma)}, R) \leq \epsilon$ with probability larger than $1 - \delta$ over the draw of γ.*

With this in mind, a thorough investigation of the benefits of RF GFlowNets is left for future research. An extensive analysis of how to usefully define p_M is of particular importance.

RFs & Stability. Besides improving the expressivity of GNN-based GFlowNets, we also demonstrate RFs impose an implicit Tikhonov regularization (Hastie et al., 2009) on the learning objective that mitigate the instability of p_B observed in Figure 8 in Section A.2. To understand this, we recall that, if $L(\theta)$ is a loss function associated with parameters θ and $\theta^\star$ is a local minimum of L,

$$L(\theta) = L(\theta^\star) + \frac{1}{2}(\theta - \theta^\star)^\top H(\theta^\star)(\theta - \theta^\star) + \mathcal{O}(\|\theta - \theta^\star\|^3)$$

with $H(\theta^\star) = \nabla^2_\theta L(\theta^\star)$ representing L's Hessian at $\theta^\star$. From this, we observe that $H(\theta^\star)$ characterizes the local geometry of L around $\theta^\star$; when $H(\theta^\star)$ is nearly singular, the loss surface around $\theta^\star$ is almost flat, a configuration that first-order gradient-based methods struggle to navigate. We next show that incorporating RFs into the policies of a GFlowNet has the effect of increasing the eigenvalues of the loss function's Hessian, reducing the geometry's flatness and—as we will empirically illustrate—reducing the functional variability of the learned (p_F, p_B) originating solely from parameter initialization. We start with the following lemma.

Lemma C.6 (Local Approximation for RF-augmented Loss)**.** *We define $\mathfrak{g}_\Gamma = \{p_F^{(\gamma)}, p_B^{(\gamma)}\}_{\gamma\in\Gamma}$ as a GFlowNet augmented with RFs γ. We assume γ is drawn from a zero-centered distribution p_M with covariance matrix $\sigma^2 I$ for some $\sigma > 0$. Then,*

$$\mathbb{E}_{\gamma\sim p_M}\left[\mathcal{L}_{\mathrm{TB}}(\tau, \gamma; \mathfrak{g}_\Gamma)\right] = \mathcal{L}_{\mathrm{TB}}(\tau, 0; \mathfrak{g}_\Gamma) + \frac{1}{2}\sigma^2 \cdot \mathrm{tr}\left(H_\gamma(\tau, 0)\right) + \mathcal{O}(\sigma^4), \tag{12}$$

with tr *being the trace operator, H_γ representing the Hessian of $\mathcal{L}_{\mathrm{TB}}(\tau, \gamma; \mathfrak{g}_\Gamma)$ with respect to γ, and $\mathcal{L}_{\mathrm{TB}}$ as in Definition B.2.*

Lemma C.6 follows from a Taylor approximation of the loss function around $\gamma = 0$ and the fact that the odd moments of γ null out. Notably, the quantity $\mathcal{L}_{\mathrm{TB}}(\tau, 0; \mathfrak{g}_\Gamma)$ corresponds to the loss of an unaugmented GFlowNet. By averaging τ out of Equation (12), we show that the trace term within it acts as a regularizer.

Proposition C.7 (RFs as Regularizers)**.** *Let $\mathfrak{g}_\Gamma$ be as in Lemma C.6 and $\mathfrak{g}_o$ be the GFlowNet corresponding to $\gamma = 0$. We define*

$$\mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_\Gamma) = \mathbb{E}_{\tau\sim\rho}\mathbb{E}_{\gamma\sim p_M}\left[\mathcal{L}_{\mathrm{TB}}(\tau, \gamma; \mathfrak{g}_\Gamma)\right],\ \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_o) = \mathbb{E}_{\tau\sim\rho}\left[\mathcal{L}_{\mathrm{TB}}(\tau, 0; \mathfrak{g}_\Gamma)\right],$$

and $\mathcal{L}_{\mathrm{Reg}}(\mathfrak{g}_\Gamma) = \mathbb{E}_{\tau\sim\rho}\left[\mathrm{tr}(H_\gamma(\tau, 0))\right]$ for an exploratory policy ρ. Also, let θ be the parameters of $\mathfrak{g}_\Gamma$, and let $H_{\mathrm{TB}} = \nabla^2_\theta \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_\Gamma)$, $H_o = \nabla^2_\theta \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_o)$, and $H_{\mathrm{Reg}} = \nabla^2_\theta \mathcal{L}_{\mathrm{Reg}}(\mathfrak{g}_\Gamma)$. In this setting,

$$H_{\mathrm{TB}} = H_o + \frac{\sigma^2}{2} \cdot H_{\mathrm{Reg}} + \mathcal{O}(\sigma^4). \tag{13}$$

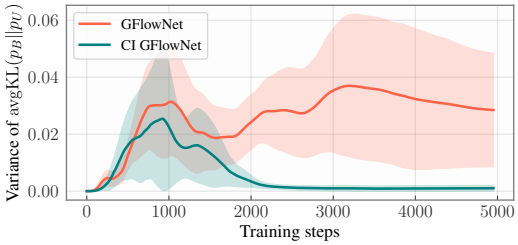


Figure 10: Random features act as a regularizer for GFlowNet training.

Importantly, Equation (13) suggests σ should be set to a small positive number to ensure the reminder term in $\mathcal{O}(\sigma^4)$ is negligible. To measure the effects of our method in practice, we calculate the variance of $\text{avgKL}(p_B||p_U) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\text{KL}[p_B(s,\cdot)||p_U(s,\cdot)]$, with p_U being an uniform backward policy. We train a continuously indexed (CI) GFlowNet with $\Gamma = \mathbb{R}^8$ and $\gamma \sim \mathcal{N}(0, 0.15^2 \cdot I)$ and a conventional GFlowNet for the HYPERGRID environment (see Section 6). As in Figure 8, we fix both the architecture and the parameter initialization of the neural networks underlying $p_F^{(\gamma)}$ and $p_B^{(\gamma)}$, and compute the per-step variance of the KL divergence by solely modifying the initial pseudorandom seed of the exploratory policy. We repeat this process across 5 independent runs. In conformity with Proposition C.7, we observe in Figure 10 that RFs have a stabilizing effect on the learning problem, smoothing the loss surface and increasing the model's robustness to the randomness of sample generation—as emphasized by the reduced variance of $\text{avgKL}(p_B||p_U)$.

D DISCRETELY INDEXED GFLOWNETS

We complement Section 5 by introducing further examples of how to partition the state space, besides the abstract modulo-based technique described in Remark 5.5.

Example D.1 (Length-based TC)**.** Let $d(x)$ be the trajectory-size, or *description-length*, for generating a $x \in \mathcal{X}$. Common GFlowNet applications such as sequence design (Laajil et al., 2025), variable selection, drug discovery (Bengio et al., 2021), and structure learning (Deleu et al., 2022) often have $d(x) \in [d_{\min}, d_{\max}]$ with both $d_{\min}$ and $d_{\max}$ known. Under this condition, we can define $\text{TC}(\mathcal{X}, K) = \{\mathcal{X}_k\}_{k=1}^K$ for $K \leq D := d_{\max} - d_{\min} + 1$ as

$$\mathcal{X}_k = \left\{ x \in \mathcal{X} \colon d_{\min}^{(k)} := d_{\min} + \frac{k-1}{K} \cdot D \leq d(x) < d_{\max}^{(k)} := d_{\min} + \frac{k}{K} \cdot D \right\}.$$

The policy $p_F^{(k)}$ can be simply implemented by masking out terminating transitions leading to x for which $d(x) \notin [d_{\min}^{(k)}, d_{\max}^{(k)})$.

Example D.2 (Hamming weight-based TC)**.** Let $\mathcal{X} = \{1, 0\}^d$, which is often the cases in applications such as bit sequences (Malkin et al., 2022) and Ising model simulation (Liu et al., 2024). We define $w(x) = \sum_{1\leq i\leq d} x_i$ as the *Hamming weight* of x and $\text{TC}(\mathcal{X}, K) = \{\mathcal{X}_k\}_{k=1}^K$ as

$$\mathcal{X}_k = \{x \in \mathcal{X} \colon w(x) \in [a_k, b_k)\}$$

with $a_1 = 0$, $b_K = d+1$, and $\bigcup_{k=1}^K [a_k, b_k) = [0, d+1)$. We implement $p_F^{(k)}$ by pruning transitions resulting in states for which the condition $w(x) \in [a_k, b_k)$ is unattainable.

The tasks above can be easily seen to fall into the symmetric-junta category we outlined in Proposition 5.7. Example D.1 straightforwardly follows from picking $\psi(s)$ to be equal to the number of components defining s. Example D.2, on the other hand, follows from letting s be represented as a subset of $\{1, 0\} \times [d]$ and defining $\psi(s) = \#(s \cap r)$ with the reference set $r = \{(1, 1), \ldots, (1, d)\}$ corresponding to a sequence containing only 1s. In either case, a tractable implementation for SC GFlowNets is admissible.

Implementation of SC GFlowNets. In addition, to further clarify the implementation of SC GFlowNets, we provide a high-level description of its training and inference in the pseudocode in Algorithms 1 and 2, respectively, alongside detailed and efficient computer code for reproducibility.

Algorithm 1 Training of SC GFlowNets

Require: $K, \mathbf{Z} = (Z_k)_{k\in[K]}$
Require: trainingIterations, $\epsilon > 0$, batchSize
Require: SC GFlowNet $\mathfrak{g}_{[K]}$
Require: getPolicies: $k, \mathfrak{g}_{[K]} \mapsto \{p_F^{(k)}, p_B^{(k)}\}$
  for _ in range(trainingIterations) **do**
    Samples $\leftarrow \{\}$
    parfor _ in range(batchSize) **do**
      $\mathbf{p}_Z \leftarrow (1-\epsilon)\cdot \frac{\mathbf{Z}}{\mathbf{1}_K^T\mathbf{Z}} + \epsilon \cdot \frac{\mathbf{1}_K}{K}$
      $k \sim \text{Categorical}(\mathbf{p}_Z)$
      $p_F^{(k)}, p_B^{(k)} \leftarrow \text{getPolicies}(k, \mathfrak{g}_{[K]})$
      # Sample trajectory from $p_F^{(k)}$
      $\tau \sim \epsilon$-greedy of $p_F^{(k)}(s_o, \cdot)$
      Samples $\leftarrow$ Samples $\cup \{(\tau, k)\}$
    end parfor
    loss $\leftarrow$ lossFunction(Samples, $\mathfrak{g}_{[K]}, \mathbf{Z}$)
    $\mathfrak{g}_{[K]}, \mathbf{Z} \leftarrow$ GradientStep(loss, $\mathfrak{g}_{[K]}, \mathbf{Z}$)
  end for

Algorithm 2 Inference with SC GFlowNets

Require: $K, \mathbf{Z} = (Z_k)_{k\in[K]}$
Require: SC GFlowNet $\mathfrak{g}_{[K]}$
Require: getPolicies: $k, \mathfrak{g}_{[K]} \mapsto \{p_F^{(k)}, p_B^{(k)}\}$
  $k \sim \text{Categorical}\left(\frac{\mathbf{Z}}{\mathbf{1}_K^T\mathbf{Z}}\right)$
  $p_F^{(k)}, p_B^{(k)} \leftarrow \text{getPolicies}(k, \mathfrak{g}_{[K]})$
  # Sample trajectory
  $\tau \sim p_F^{(k)}(s_o, \cdot)$
  # Get terminal state from τ
  $x \leftarrow \text{TerminalState}(\tau)$
  return x

E Revisiting Subgraph Asynchronous Learning (SAL)

This section and the next highlight how our framework can be instantiated to accommodate prior algorithms for learning GFlowNets. We start with SAL (Silva et al., 2025b). Similarly to SC GFlowNets, SAL defines a fixed-horizon covering (FHC)[4] of $\mathcal{X}$, referred to as $\text{FHC}(\mathcal{X}, K)$.

Definition E.1 (Fixed-Horizon Covering)**.** We call $\text{FHC}(\mathcal{X}, K) = \mathcal{X}_o \cup \{\mathcal{X}_k\}_{k=1}^K$ a *fixed-horizon covering* (FHC) if the following conditions are satisfied.

1. There are disjoint subsets $\{\mathcal{S}_k\}_{k=1}^K$ of $\mathcal{S}$ such that $\mathcal{X}_k$ contains all states in $\mathcal{X}$ reachable from some $s \in \mathcal{S}_k$.
2. Each $s \in \mathcal{S}_k$ is within a fixed-distance d to s_o, and all states within distance d of s_o are contained in some $\mathcal{S}_k$ (fixed-horizon).
3. $\mathcal{X}_o$ is the subset of $\mathcal{X}$ whose elements' to s_o to x is smaller than d.

We call *distance from s to s_o* the length of the shortest-path from s_o and s.

Drawing on Definition E.1, each partition enforces a so-called Amortized Trajectory Balance (ATB), defined as

$$F(s,k)p_F^{(k)}(s,\tau) = R(x)p_B(x,\tau) \text{ for all } k \in [K],\ s \in \mathcal{S}_k,\ x \in \mathcal{X}_k, \text{ and } \tau \in s \rightsquigarrow x, \tag{14}$$

and then $Z \cdot p_F^{(o)}(s,\tau') = F(s,k)p_B(s,\tau')$ and $Z \cdot p_F^{(o)}(s,\tau') = R(x)p_B(s,\tau')$ for $k \in [K]$, $s \in \mathcal{S}_k$, $x \in \mathcal{X}_o$, and $\tau \in s_o \rightsquigarrow s$ and $\tau' \in s_o \rightsquigarrow x$. In other words, each model first enforces a balance condition within its own partition according to a shared (uniform) backward policy p_B, and their learned flow functions $F(\cdot, k)$ are later used as the target distribution of the centralized model's ($p_F^{(o)}$) own learning objective.

We next show that a solution to SAL's amortized TB can be interpreted as a DI GFlowNet satisfying its own MTB in Definition 3.2. We do this by introducing a Doob h-transform (see Section 5.2) that maps $p_F^{(o)}$ and $p_F^{(k)}$ into a policy function that strictly goes through $\mathcal{S}_k$ when sampling from $\mathcal{X}_k$. With this in mind, we let

$$T_k^{(f)} = \min\{t\colon S_t^{(f)} \in \mathcal{S}_k\} \text{ and } T_k^{(b)} = \min\{t\colon S_t^{(b)} \in \mathcal{S}_k\}$$

[4]Silva et al. (2025b) called it a fixed-horizon *partition* (FHP); however, the underlying sets need not be disjoint, as the word *partition* suggests.

be the hitting times on $\mathcal{S}_k$ of the Markov chains introduced in Proposition 5.9, and

$$h_k(s) = \mathbb{P}_{\kappa_F}\left[T_k^{(f)} = \min_{k' \in [K]} T_{k'}^{(f)} < \infty \middle| s\right] \text{ and } g_k(s) = \mathbb{P}_{\kappa_B}\left[T_k^{(b)} = \min_{k' \in [K]} T_{k'}^{(b)} < \infty \middle| s\right]$$

be the corresponding probabilities that a Markov chain starting at $s \in \bar{\mathcal{S}}$ and following κ_F (resp. κ_B) arrives at $\mathcal{S}_k$ prior to arriving at any other $\mathcal{S}_{k'}$ with $k' \neq k$. We first show that both h_k and g_k are harmonic functions.

Lemma E.2. *Both h_k and g_k are harmonic functions on $\bar{\mathcal{S}} \setminus \mathcal{S}_k$ for $k \in [K]$.*

A core property required by the next Proposition E.4 is that both $h_k(s_o)$ and $g_k(x)$ for $x \in \mathcal{X}$ can be interpreted as probability distributions over $[K]$. As we emphasize next, this intuitively follows from the fact that $\{\mathcal{S}_k\}_{k=1}^K$ forms a boundary between s_o and the set $\mathcal{X}$ of terminal states.

Lemma E.3. *The functions h_k and g_k in Lemma E.2 satisfy*

$$\sum_{0 \leq k \leq K} h_k(s_o) = 1 \text{ and } \sum_{0 \leq k \leq K} g_k(x) = 1 \text{ for } x \in \mathcal{X}$$

with $h_o(s_o) = \mathbb{P}_{\kappa_F}\left[T_k^{(f)} = \infty \forall k \in [K] \middle| s_o\right]$ and $g_o(x) = \mathbb{P}_{\kappa_B}\left[T_k^{(b)} = \infty\, \forall k \in [K] \middle| x\right]$ referring to the case when the generative process terminates on $\mathcal{X}_o$ (see Definition E.1).

Building on Lemmas E.2 and E.3, we demonstrate in Proposition E.4 that SAL can be viewed as an instance of a DI GFlowNet by showing that a solution to Equation (14) can be uniquely represented as a solution to the mixture learning problem in Definition 3.2 with a specific choice of Z_k and $q(k|x)$. This suggests yet another form of designing a DI GFlowNet. It should nonetheless be noticed that, contrarily to SC GFlowNets, SAL is an inherently distributed algorithm—as neither h_k nor g_k in Lemma E.2 can be tractably computed in general.

Proposition E.4 (SAL as a DI GFlowNet)**.** *Let $\{p_F^{(k)}\}_{k=0}^K$ and p_B be the solutions to SAL's balance equations having d as the distance from s_o to $\{\mathcal{S}_k\}_{k=1}^K$. We define*

$$\tilde{p}_F^{(k)}(s, s') = \begin{cases} p_F^{(o)}(s, s') \cdot \frac{h_k(s')}{h_k(s)} \text{ if } \mathrm{dist}(s_o, s') \leq d, \\ p_F^{(k)}(s, s') \text{ otherwise}, \end{cases}$$

with dist *representing the shortest-path distance in the underlying state graph, and*

$$\tilde{p}_B^{(k)}(s, s') = \begin{cases} p_B(s, s') \text{ if } \mathrm{dist}(s_o, s') \leq d, \\ p_B(s, s') \cdot \frac{g_k(s')}{g_k(s)} \text{ otherwise}, \end{cases}$$

for $k \geq 1$ and $\tilde{p}_B^{(o)}(s, s') = p_B(s, s')$ for $k = 0$. Then, for $k \in \{0, \ldots, K\}$ and each trajectory τ from s_o to $x \in \mathcal{X}_k$ going through $\mathcal{S}_k$,

$$Z_k \tilde{p}_F^{(k)}(s_o, \tau) = R(x) q(k|x) \tilde{p}_B^{(k)}(x, \tau)$$

with $Z = \sum_{x \in \mathcal{X}} R(x)$ and $Z_k = Z \cdot h_k(s_o)$ and $q(k|x) = g_k(x)$.

In simpler terms, Proposition E.4 highlights SAL can be cast into our mixture learning framework by reframing $h_k(s_o)$ and $g_k(x)$ as our *prior* and *posterior* mixing distributions, an interpretation supported by the normalization property established in Lemma E.3. Notably, the central statements on the correctness of SAL in (Silva et al., 2025b) can be immediately derived from Proposition E.4, highlighting the generality of DI GFlowNets as a theoretical tool for devising GFlowNet training algorithms with proven correctness. The next section emphasizes this point further.

F Revisiting Boosted GFlowNets

This section is separated into two parts. The first one shows how Boosted GFlowNets can be understood through the lens of our abstract framework. The second one evaluates Boosted GFlowNets against our proposed SC GFlowNets, highlighting the computational amenability of the latter when compared to the former.

F.1 BOOSTED GFLOWNETS SATISFY THE MIXED TRAJECTORY BALANCE

As with SC GFlowNets and SAL, Boosted GFlowNets learn distinct samplers focusing on nearly-disjoint regions of the state space (Locatello et al., 2018; Dall'Antonia et al., 2026b). However, instead of relying on amortization (as in SC GFlowNets) or parallelization (SAL) over fixed structural primitives, Boosted GFlowNets adaptively learn a *residual target distribution* in a sequential fashion. Their connection to DI GFlowNets, which will be developed in this section, will reveal our framework's applicability in the context of *countably infinite* mixtures.

To see this, we briefly recall the algorithmic design of Boosted GFlowNets. Let $\{\mathfrak{g}_k\}_{k\geq 1} \coloneqq \{(Z_k, p_F^{(k)}, p_B^{(k)})\}_{k\geq 1}$ be GFlowNets and, for $\tau \in s_o \rightsquigarrow x$, define

$$R^{(0)}(x,\tau) = 0 \text{ and } R^{(k)}(x,\tau) = R^{(k-1)}(x,\tau) + \frac{Z^{(k)} p_F^{(k)}(s_o,\tau)}{p_B^{(k)}(x,\tau)}$$

for $k \geq 2$. Intuitively, $R^{(k)}(x,\tau)$ represents an estimate of the accumulated probability mass on x by the first k models. An implicit assumption of Boosted GFlowNets, which is compatible with the observed empirical behavior of GFlowNets (Shen et al., 2023), is that each GFlowNet underestimates the total probability mass in any given region.

Assumption F.1 (Underallocation of probability mass)**.** After trained, we assume assume the first k-th GFlowNets satisfy $R^{(k)}(x,\tau) \leq R(x)$ for all x and $\tau \in s_o \rightsquigarrow x$.

Under Assumption F.1, each GFlowNet is trained by minimizing the *boosted trajectory balance (BTB)* loss function, which replaces $R(x)$ by the *residual target* $R(x) - R^{(K)}(x,\tau)$ with τ drawn from the corresponding backward policy $p_B^{(K)}(x,\cdot)$.

Definition F.2 (Boosted Trajectory Balance Condition & Loss)**.** Let $\{\mathfrak{g}_k\}_{k=1}^K$ be a sequence of GFlowNets with $\{\mathfrak{g}_k\}_{k=1}^{K-1}$ satisfying Assumption F.1. By defining

$$R^{(k)}(x) = \mathbb{E}_{\tau\sim p_B^{(k)}(x,\cdot)}\left[R^{(k)}(x,\tau)\right],$$

we call K-th *Boosted Trajectory Balance* (BTB) condition

$$Z_K \cdot p_F^{(K)}(s_o,\tau) = (R(x) - R^{(K-1)}(x)) \cdot p_B^{(K)}(x,\tau),$$

which is enforced through the loss function

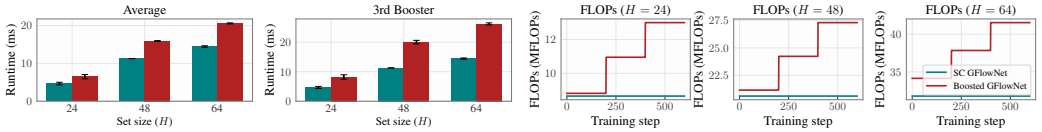
$$\mathcal{L}_{\text{BTB}}\left(\tau \middle| \{\mathfrak{g}_k\}_{k=1}^K\right) = \left(\log \frac{Z_K \cdot p_F^{(K)}(s_o,\tau)}{(R(x) - R^{(K-1)}(x)) \cdot p_B^{(K)}(x,\tau)}\right)^2,$$

which we refer to as *BTB loss*.

The intuition for the BTB loss is that, when the the first $K-1$ GFlowNets *collapse* onto a subset of the state space, the K-th GFlowNet will—through the influence of the residual target—mostly ignore such region. In practice, (Dall'Antonia et al., 2026b) proposes several techniques for stabilizing training when the quantity $R(x) - R^{(K)}(x)$ becomes overly small, to which the interested reader is referred to for further details; however, none of them are relevant to our discussion. It should be clear from Definition F.2 and Assumption F.1 that Boosted GFlowNets only sample from the target distribution when the trajectory-wise residual target vanishes and that the BTB loss can only be non-biasedly evaluated if we can compute $R^{(K-1)}(x)$ exactly. The latter is satisfied when $R^{(K-1)}(x,\tau) = R(x)$ for every τ in the support of p_F, and 0 otherwise; equivalently, when the on-policy expectation of the BTB loss under $p_F^{(K-1)}$ vanishes. This was the main condition for (Dall'Antonia et al., 2026b, Theorem 1). We explicitly state these behaviors in our next assumption.

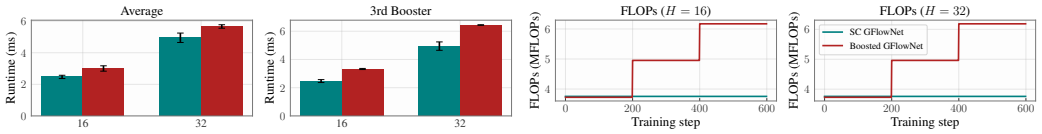
Assumption F.3 (Convergence of residuals & Collapse)**.** For all $x \in \mathcal{X}$ and $\tau \in s_o \rightsquigarrow x$, the series $\{R^{(k)}\}_{k\geq 1}$ satisfies $\lim_{k\to\infty} R^{(k)}(x,\tau) = R(x)$. Also, there is a subset $\mathcal{X}_k$ of $\mathcal{X}$ such that $R^{(k)}(x,\tau) = R^{(k)}(x)$ for each $\tau \in s_o \rightsquigarrow x$ and $x \in \mathcal{X}_k$, and $R^{(k)}(x,\tau) = 0$ otherwise.

We now show that Assumptions F.1 and F.3 allow us to characterize Boosted GFlowNets as a solution to the MTB equations in Definition 3.2 based on a countably infinite mixing distribution. Importantly, as with classical topic modelling (Blei et al., 2003; Hoffman et al., 2010) based on Dirichlet processes, we typically choose a K and truncate the prior mixing distribution to be supported on $[K]$. Dall'Antonia et al. (2026b) sets K to 2 or 3 in their experiments.



(a) Runtimes per training step. (b) FLOPs per training step.

Figure 11: SC GFlowNets run both faster (left) and more compute-efficiently (right) than Boosted GFlowNets for the SET GENERATION domain. Throughout this section, runtimes represent per-step averages over 3 independent runs, each with 600 iterations.



(a) Runtimes per training step. (b) FLOPs per training step.

Figure 12: SC GFlowNets are more compute-efficient than Boosted GFlowNets in the HYPERGRID domain, requiring substantially fewer FLOPs per gradient step during training.

Proposition F.4 (Boosted GFlowNets as DI GFlowNets)**.** *We define* $\{\mathfrak{g}_k\}_{k\geq 1} := \{(Z_k, p_F^{(k)}, p_B^{(k)})\}_{k\geq 1}$ *as GFlowNets satisfying Assumptions F.1 and F.3. By letting*

$$R_k(x) = R^{(k)}(x) - R^{(k-1)}(x) \text{ and } q(k|x) = \frac{R_k(x)}{R(x)},$$

then the following properties are satisfied.

1. $q(k|x)$ *is a probability distribution, i.e.,* $\sum_{k\geq 1} q(k|x) = 1$ *for each* $x \in \mathcal{X}$,
2. *The sequence* $\{\mathfrak{g}_k\}_{k\geq 1}$ *satisfies a MTB condition*

$$Z_k \cdot p_F^{(k)}(s_o, \tau) = R(x)q(k|x)p_B^{(k)}(x, \tau)$$

for $k \geq 1$, *with* $\sum_{k\geq 1} Z_k = Z$.

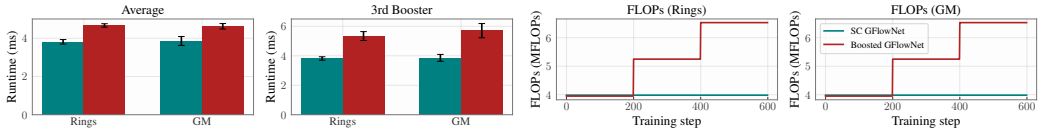
Once again, the correctness of Boosted GFlowNets can be directly inferred from Corollary 5.2 and Proposition F.4. However, contrarily to SC GFlowNets, which support both amortization and distributed training, Boosted GFlowNets can only be trained via sequential learning. Also, in contrast to both SC GFlowNets and SAL, Boosted GFlowNets cannot be trained with an near-embarrassingly parallel algorithm, primarily due to the unbounded support of the mixing distribution and the dependence of each model on its predecessors' residuals.

In addition, we notice SC GFlowNets are the most foundational instantiation of DI GFlowNets when compared to both SAL and Boosted GFlowNets: stratum-conditioning could be used for training each booster in Boosted GFlowNets and each sampler in SAL, but neither of these algorithms could be used to learn SC GFlowNets without sacrificing its monolithicness, which we deem to be one of its fundamental properties—a single model *can* be used for learning the entire mixture.

F.2 EXPERIMENTS

We now present several experiments to support our claim that (i) Boosted GFlowNets introduce a substantial computational overhead over the base model, which is not observed for SC GFlowNets, and (ii) SC GFlowNets often converge faster than Boosted GFlowNets.

Computational profile. We start by showcasing the cost of Boosted GFlowNets is substantially larger than of SC GFlowNets in terms of both runtime and FLOPs per training step for the SET GENERATION, HYPERGRID, and LAZY RANDOM WALK domains. To evaluate the latter, we use JAX's `cost_analysis` method for compiled functions (Bradbury et al., 2018). Results are displayed in Figures 11 to 13. Notably, SC GFlowNets are up to roughly $1.6\times$ faster than Boosted



(a) Runtimes per training step. (b) FLOPs per training step.

Figure 13: SC GFlowNets is far more efficient than Boosted GFlowNets in the LAZY RANDOM WALK domain (`Rings` corresponds to the RINGS target; `GM`, to GAUSSIAN MIXTURE).

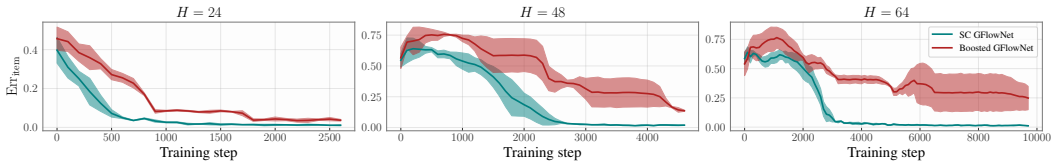


Figure 14: SC GFlowNets converge substantially faster than Boosted GFlowNets in the SET GENERATION domain, particularly for larger state spaces. See also Figure 3 for $\mathrm{Err}_{\mathrm{item}}$'s definition.

GFlowNets per step (for SET GENERATION), and execute up to 50% fewer FLOPs per training step. The reasons for this are clear: as discussed before, our method learns an *amortized policy* over the mixture's components, while boosting requires training independent neural networks in a sequential fashion, having a monotonically increasing cost in the number of *boosters* (components). Importantly, during our experiments, we proceeded as suggested by Dall'Antonia et al. (2026b) and allocated a prescribed number of iterations equally among a predefined number of boosters for training; automatic sampler elicitation remains an unresolved issue for Boosted GFlowNets.

Learning convergence. In addition to being more compute-efficient, SC GFlowNets often lead to faster learning convergence in the considered tasks. This is highlighted in Figures 14 to 16. The intuition for this is that, while both Boosted and SC GFlowNets enhance state space exploration by design, hence improving the rate of discovery of high-probability regions, our method does so by stably partitioning the state space, ensuring consistent, non-random exploration of specific subsets of it. In contrast, boosting is built on a more delicate approach that is is reliant upon accurate training of several independent neural networks, and can be biased—unless Assumptions F.1 and F.3 are satisfied.

G EXPERIMENTAL DETAILS & FURTHER EXPERIMENTS

This section extends empirical analysis in Section 6 with additional domains, target distributions, and TC's constructions. In addition, we provide further experimental details for reproducing our results. We also attach the computer code, written using JAX (Bradbury et al., 2018), we used for training and evaluating each of the described models on the considered tasks.

G.1 MORE EXPERIMENTS

Autoregressive sequence generation. We train a GFlowNet to reproduce the following generative process of token sequences $(t_1, \ldots, t_l)$ over a finite vocabulary,

$$t_1 \sim p_{\text{init}},\ l \sim p_{\text{length}},\ t_i | t_{i-1} \sim p_{\text{kernel}}(t_{i-1}, \cdot) \text{ for } i \in \{2, \ldots l\}. \tag{15}$$

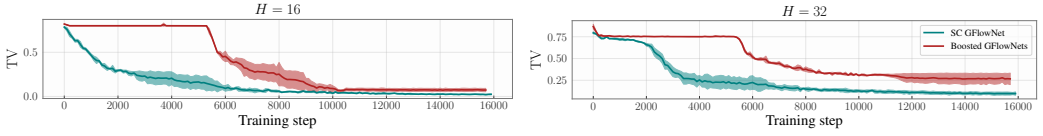
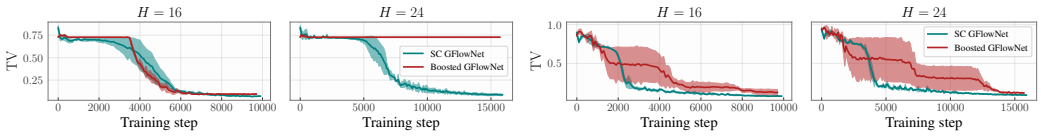
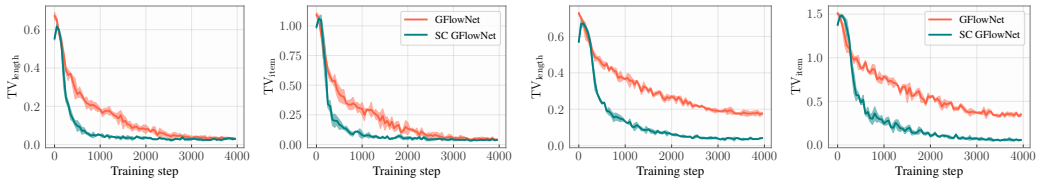


Figure 15: SC GFlowNets exhibit faster learning convergence when compared to Boosted GFlowNets, while also being far more compute-efficient (Figure 12).



(a) RINGS. (b) GAUSSIAN MIXTURE.

Figure 16: On top of being more computationally efficient (Figure 13), SC GFlowNets also learn a more accurate distributional approximation to the target relatively to Boosted GFlowNets in both the RINGS and GAUSSIAN MIXTURE variants of the LAZY RANDOM WALK domain.



(a) $L = 18$. (b) $L = 24$.

Figure 17: **SC GFlowNets significantly accelerate learning convergence** when approximating the generative process in Equation (15), with greater improvements being observed for larger state spaces. $\mathrm{TV}_{\mathrm{item}}$ and $\mathrm{TV}_{\mathrm{length}}$ are defined in Equation (17).

We let $[V] \coloneqq \{1, \ldots, V\}$ be the support of p_{init} (*vocabulary*) and $p_{\mathrm{kernel}}(t, \cdot)$ for $t \in [V]$ and $\{0\} \cup [L]$ be that of p_{length}. The target distribution associated with $\mathbf{t} = (t_1, \ldots, t_l)$ is

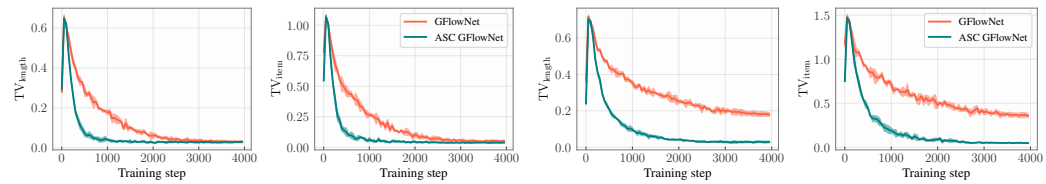
$$R(\mathbf{t}) = p_{\mathrm{init}}(t_1) p_{\mathrm{length}}(l) \prod_{2 \leq i \leq l} p_{\mathrm{kernel}}(t_{i-1}, t_i). \tag{16}$$

The GFlowNet is built upon a generative process starting at an empty sequence and iteratively deciding either to add a token from $[V]$ or to stop the generation, and the SC GFlowNet is designed according to Example D.1 with $K = 3$ partitions. Due to its simplicity, many quantities of interest from the model in Equation (16) can be tractably computed, including the marginal distribution over sequence lengths and the value of t_i for $i \in [L]$. We denote these by $\pi(l)$ and $\pi(i, v)$ for $i \in [L]$ and $v \in [V]$, respectively, and evaluate our model with

$$\mathrm{TV}_{\mathrm{length}} = \frac{1}{2} \sum_{0 \leq l \leq L} |\hat{\pi}(l) - \pi(l)| \text{ and } \mathrm{TV}_{\mathrm{item}} = \frac{1}{2} \max_{v \in [V]} \sum_{1 \leq i \leq L} |\hat{\pi}(i, v) - \pi(i, v)|, \tag{17}$$

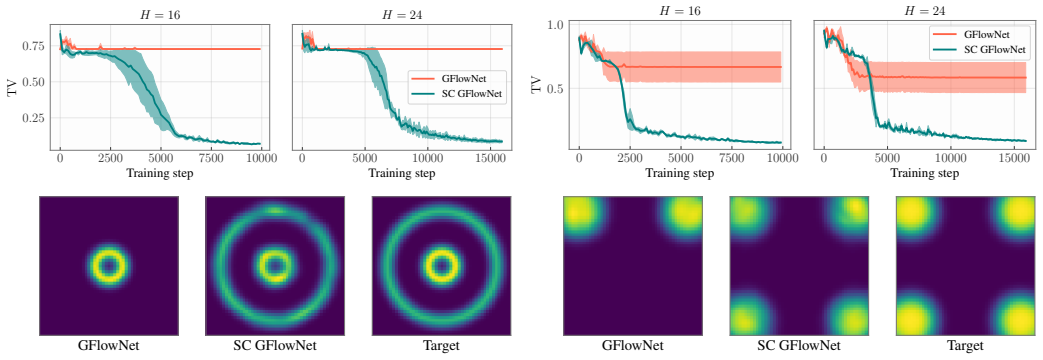
with $\hat{\pi}$ being a Monte Carlo estimate of π based on the learned sampler. We illustrate the evolution of these metrics throughout training in Figure 17. As we can observe, SC GFlowNets significantly improve learning convergence.

We also evaluate ASC GFlowNets on this domain, training $K = 3$ separate samplers in parallel on distinct subsets of the state space. We show the results in Figure 18. Again, we notice our partitioning technique significantly speeds up training.



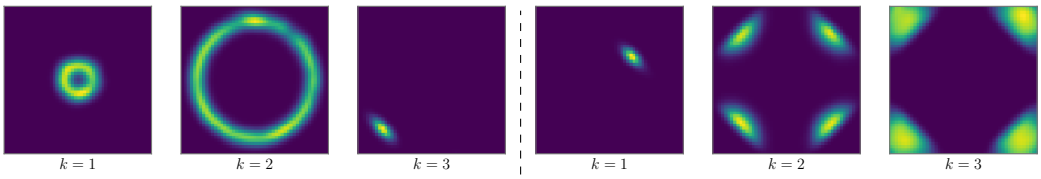
(a) $L = 18$. (b) $L = 24$.

Figure 18: Embarrassingly parallel counterpart of Figure 17, comparing ASC GFlowNet against the best of K randomly initialized and independently trained GFlowNets with matched computational cost. ASC GFlowNet accelerates learning convergence.



(a) RINGS. (b) GAUSSIAN MIXTURE.

Figure 20: **SC GFlowNets improve goodness-of-fit** for the LAZY RANDOM WALK task, exhaustively covering the target distribution's high probability regions even for an imperfect state space partitioning (as for the GAUSSIAN MIXTURE; see Figure 21).



(a) RINGS. (b) GAUSSIAN MIXTURE.

Figure 21: SC GFlowNet's marginal $m_F^{(k)}(s_o, x)$ (recall Equation (1)) in each $x \in \mathcal{X}$ for both the (a) RINGS and (b) GAUSSIAN MIXTURE target distributions of the LAZY RANDOM WALK domain.

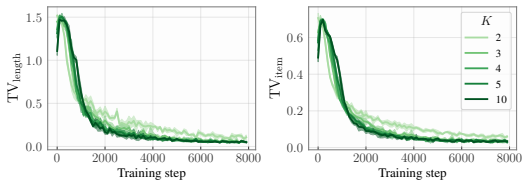


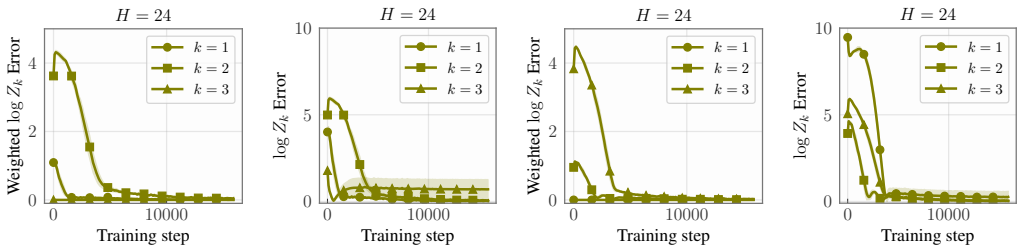
Figure 19: Accuracy of SC GFlowNets as a function of the number of components in the mixture.

Another question that comes up when designing SC GFlowNets is *how to select the number of partitions*. Figure 19 shows that, for this domain, there is no significant difference between choosing $K \in \{2, 3, 4, 5, 10\}$. This may not always be the case, however; see Figure 26. We discuss this further when describing the LAZY RANDOM WALK domain.

Importantly, the question of whether one should use ASC or SC GFlowNets depends primarily on the available computational resources. ASC GFlowNets, being massively parallelizable, have access to larger training sets and greater model capacity than any individual SC GFlowNet, being expected to both improve state space exploration and learning convergence, should the extra processing power be available. From an implementational perspective, as the samplers are architecturally identical, this parallelization can be directly attained by using tools designed to express single-program, multiple-data (SPMD) computational models (Darema et al., 1988), such as `jax.pmap`.

Lazy Random Walk. The state space for the LAZY RANDOM WALK domain is $\mathcal{S} = \{(\mathbf{x}, t) \colon \mathbf{x} \in \{-H, \dots, H\}^d \text{ and } 0 \leq t < T\}$ and $\mathcal{X} = \{(\mathbf{x}, T) \colon \mathbf{x} \in \{-H, \dots, H\}^d\}$ for given positive integers H, d, and T. The generative process starts at $s_o = (\mathbf{0}, 0)$, and each iteration adds either 1, -1, or 0 to a chosen coordinate of $\mathbf{x}$—as long as it remains within the boundary of $\mathcal{S}$—and increments t by one until $t = T$. As in (Dall'Antonia et al., 2026a; Silva et al., 2026), we consider $d = 2$, $H = 16$, and $T = 32$. Our target distributions, RINGS and GAUSSIAN MIXTURE, are shown in Figure 20.

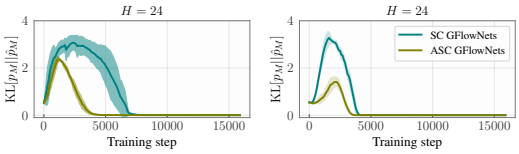
Notably, as the trajectory size is the same for each $(\mathbf{x}, T)$, the length-based partitioning of Example D.1 cannot be used for LAZY RANDOM WALK. Instead, we propose a uniform *distance*-based partitioning using the function $\psi(\mathbf{x}, T) = \|\mathbf{x}\|_1$ in Remark 5.5 with $K = 3$. We compute the TV distance between the learned and target distributions to measure a sampler's goodness-of-fit.



(a) RINGS. (b) RINGS. (c) GAUSSIAN MIXTURE. (d) GAUSSIAN MIXTURE.

Figure 22: Learned log-partition functions by ASC GFlowNets in the LAZY RANDOM WALK domain. We recall that, for RINGS, $k = 3$ is the *starved* component, containing negligible probability mass; for GAUSSIAN MIXTURE, it is $k = 1$. As expected, ASC GFlowNets learn the correct values (a, c); however, the approximation of $\log Z_k$ for regions with near-zero probability is imperfect (b, d), as sizeable deviations in $\log Z_k$ in this case have little effect on distributional accuracy.

Figure 20 shows both the model's accuracy throughout training and a snapshot of the learned distribution for both SC and a traditional GFlowNet. In addition, we present in Figure 21 each component $p_F^{(k)}$ for $k \in \{1, 2, 3\}$ of the SC GFlowNet. We notice that our mixture modelling framework both accelerates learning convergence and improves fitness—even when the chosen partitioning does not separate the target distribution's high probability regions, as in the GAUSSIAN MIXTURE example. The intuition for this is that the sampler will naturally avoid being pulled back to and bogged in previously visited modes during training when we constraint the future value of $\|\mathbf{x}\|_1$ at $t = T$. We also compare ASC to SC GFlowNets in Figure 24 for both examples; as expected, ASC GFlowNets find a distinctly better fit to the target—particularly in the case of RINGS, for which each component receives an easier target with connected high-probability regions.



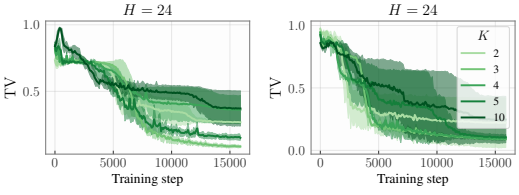
(a) RINGS. (b) GAUSSIAN MIXTURE.

Figure 23: KL between prior mixing distributions.

Importantly, due to having a tractably enumerable state space, the LAZY RANDOM WALK domain allows us to efficiently evaluate the log-partitions functions. This provides another road for evaluating our model. To see this, we denote by $\log \hat{Z}_k$ and $\log \hat{Z}$ the learned partition functions, and by $\log Z_k$ and $\log \hat{Z}$ their ground-truth values for $k \in [K]$. We consider the metrics

$$\mathrm{KL}[p_M||\hat{p}_M] \coloneqq \sum_{1 \le k \le K} \frac{Z_k}{Z} \cdot \log \frac{Z_k/Z}{\hat{Z}_k/\hat{Z}}, \left(\frac{Z_k}{Z} \cdot \left| \log \hat{Z}_k - \log Z_k \right| \right), \text{ and } \left(\left| \log \hat{Z}_k - \log Z_k \right| \right),$$

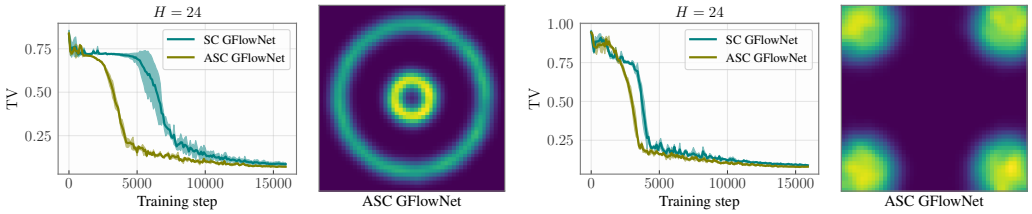
for $k \in [K]$, which respectively evaluate the accuracy of the learned prior mixing distribution and the relative and absolute errors of the learned partition functions. We separate relative and absolute metrics as, in practice, if a Z_k is negligible when compared to other $Z_{k'}$, small discrepancies in Z_k will result in large deviations for $\log \hat{Z}_k$. As an example, the distributions with logits $(1, 2, -10)$ and $(1.05, 1.99, -15)$ are effectively the same, despite the large difference in the values of the third component. We display these values in Figures 22 and 23, with a comparison between SC and ASC GFlowNets according to them in the latter.



(a) RINGS. (b) FOUR GAUSSIANS.

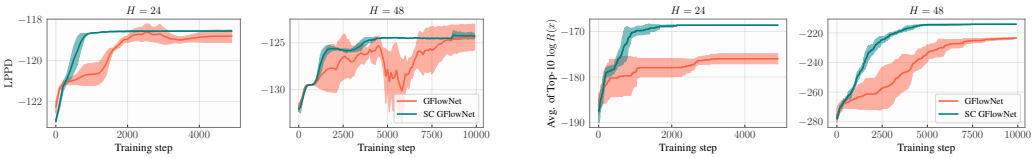
Figure 26: SC GFlowNets' distributional accuracy for distinct numbers of partitions.

We once again consider the problem of choosing K. Figure 26 highlights the dependence of SC GFlowNets's accuracy on K for the LAZY RANDOM WALK domain. Although SC GFlowNets improve upon a monolithic sampler for all evaluated choices of $K \ge 2$, there is also a noticeable difference between them. In view of this, we provide two separate suggestions for selecting this hyperparameter. The first is to always let $K = 3$ components,



(a) RINGS. (b) GAUSSIAN MIXTURE.

Figure 24: **ASC GFlowNets converge faster than their synchronous counterparts** in the LAZY RANDOM WALK task, achieving a more accurate approximation in fewer training iterations.



(a) Log-predictive posterior density of the observed data (LPPD).

(b) Average unnormalized target density of the 10 most likely encountered states.

Figure 25: **SC GFlowNets improve learning convergence** (left) and accelerate the discovery of high-probabiltiy states (right) for the Bayesian VARIABLE SELECTION task; H represents the number of variables (columns) in the dataset.

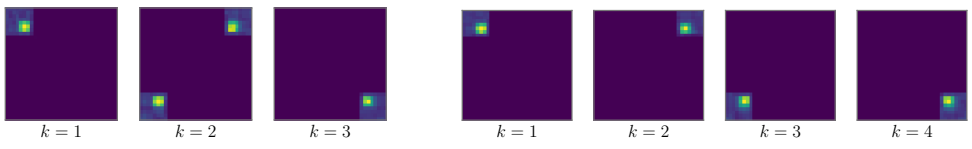
should this choice be compatible with the chosen TC; despite its simplicity, we found setting $K = 3$ to consistently achieve an accurate approximation to the target distributions we considered. The second is to rely upon standard hyperparameter selection protocols, such as carrying out short training runs and measuring the sampler's performance afterwards, to pick K. With this in mind, future research should investigate principled approaches for algorithmic picking of K. Nonetheless, a useful intuition to have is that the complexity of the function learned by our component-wise amortized policy increases with K, due to which we advise against setting too large a value for the number of components in SC GFlowNets.

The main takeaway from these experiments, however, is that designing an effective covering for SC GFlowNets *does not require* fine-grained prior knowledge about the shape of the target distribution.

Variable Selection. We consider the task of Bayesian variable selection (George & McCulloch, 1993). Given a data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ with $x^{(i)} \in \mathbb{R}^H$ and $y^{(i)} \in \mathbb{R}$, we let $p(y|x, S, \beta) = \mathcal{N}(y|\beta_S^T x_S, \sigma^2)$ be defined according to a linear model with observational noise σ and coefficients $\beta \in \mathbb{R}^H$ and features $S \subseteq [H]$. We here denote by x_S the vector $(x^{(i)})_{i \in S}$ composed of the indices i in S, with a similar definitiong for β_S. In this setting, we train a sampler to draw from the marginal posterior distribution over S, i.e.,

$$R(S) \coloneqq p(S|\mathcal{D}) = \int_{\mathbb{R}^H} p(S, \beta|\mathcal{D}) \mathrm{d}\beta,$$

with $p(S, \beta|\mathcal{D})$ being the usual Bayesian posterior for S and β. We notice $p(S|\mathcal{D})$ can be computed in closed form up to a normalizing constant independent of S for a linear model. After training, we



(a) Length-based TC. (b) Octant-based TC.

Figure 27: Component-wise learned distributions for both considered SC GFlowNets in Figure 5.

evaluate the samplers according to the predictive posterior distribution of the observed data, as it is usually done in Bayesian modelling, and to the average unnormalized log-posterior of the 10 most likely variable sets, a common approach for assessing exploration effectiveness in the GFlowNet literature, e.g., (Kim et al., 2024; 2023; 2025b). We show both metrics in Figure 25, with the SC GFlowNet being designed as in the SET GENERATION domain. Notably, our mixture model improves both learning and the rate at which modes are encountered during training.

G.2 EXPERIMENTAL DETAILS

We provide further details for how each domain was implemented, including model architecture, mixture-encoding details, optimizer, training iterations, and batch size per gradient step. We ran all our experiments in a MacBook Pro M4 with 16 GB of RAM, and we implemented our models using Jax's Flax library (Bradbury et al., 2018; Heek et al., 2026). All reported results, including error bars, represent averages and standard deviations computed over three independent runs.

In every experiment, we used the Muon optimizer (Jordan et al., 2024) for the neural network's parameters, and Adam (Kingma & Ba, 2014) for the $\log Z$ and $\{\log Z_k\}_{k\in[K]}$ parameters. Unless stated otherwise, we parameterized the policies p_F and p_B of the GFlowNet using an MLP with two 64-dimensional layers,

$$p_F(s,\cdot) = \mathrm{Softmax}\left(\mathrm{MLP}(\iota(s)) \odot \mathbf{m}(s)\right),$$

with $\iota(s)\colon \mathcal{S} \to \mathbb{R}^{64}$ being a learned vector embedding for the state, and $\mathbf{m}(s) \in \{1, -\infty\}^f$ being masking function that ensures the sampler is constrained to the target region (either $\mathcal{X}$ or $\mathcal{X}_k$, in the case of SC GFlowNets); f is the maximum number of children a given state s has in the underlying state graph. We implement a similar representation for p_B, using a shared encoder and simply changing the output layer and the mask. In its centralized form, SC GFlowNets are implemented as

$$p_F^{(\gamma)}(s,\cdot) = \mathrm{Softmax}(\mathrm{MLP}(\iota(s) \oplus \iota_\gamma(\gamma)) \odot \mathbf{m}(s,\gamma)). \tag{18}$$

Again, $\mathbf{m}(s,\gamma)$ is a masking function, which depends on γ in the case of SC GFlowNets but not in the case of RF GFlowNets; $\oplus$ represents concatenation. To ensure matching computational costs and similar parameter counts for both SC GFlowNets and GFlowNets, we let $\iota\colon \mathcal{S} \to \mathbb{R}^{64-d_\gamma}$ and $\iota_\gamma\colon \Gamma \to \mathbb{R}^{d_\gamma}$, with d_γ being the dimension of the learned representation for γ. For SC GFlowNets, ι_γ is a learned dictionary from $[K]$ to $\mathbb{R}^{d_\gamma}$. For RF GFlowNets, ι_γ is a linear layer. We set $d_\gamma = 8$ in all experiments, which we found to be consistently effective. ASC GFlowNets, on the other hand, was trained with K identical copies of Equation (18); as each model learns to sample from a distinct $\mathcal{X}_k$, information about k was not required to be explicitly encoded into the neural network.

Set generation. We considered sets of size $\{24, 48, 64\}$, and trained each sampler for 3000, 5000, 10000 iterations, respectively, using a batch size of 64 trajectories.

Hypergrid. We trained the models for $H \in \{16, 32\}$, both of which over 16000 iterations and a batch size of 64 trajectories per step. The reward function is the same as in (Malkin et al., 2022), i.e.,

$$\log R(\mathbf{x}) = \log\left(\frac{1}{2}\prod_{i=1}^{d}\mathbb{1}(a_i > 0.5) + 2\prod_{i=1}^{d}\mathbb{I}(0.6 < a_i < 0.8) + \rho\right), \tag{19}$$

with $\rho = 10^{-3}$ and $a_i = 2 \cdot \mathbf{x}_i \cdot (H-1)^{-1} - 1$.

Ancestral Graphs. We considered two distinct tasks under the ANCESTRAL GRAPHS domain. The first, considered also in (da Silva et al., 2026), considered graphs with $H \in \{15, 20\}$ nodes and used the BIC for scoring each AG,

$$R(x) = -\mathrm{BIC}(x) = 2\log\hat{p}(\mathcal{D}|x) - |x|_E \log n - 2|x|_E \log|x|_N, \tag{20}$$

in which $\hat{p}$ is the maximum likelihood of $\mathcal{D}$ given x, computed via Drton & Richardson (2004)'s Iteratively Reweighted Least Squares (IRLS) algorithm, $|x|_N$ is the number of nodes (H) in x, $|x|_E$, the number of edges, and $\mathcal{D}$ was drawn from a randomly parameterized linear Gaussian structural equation model with H variables. We used the same data generative process as in (da Silva et al., 2026).

The second task consisted of simulating the Erdős Renyi model, whose unnormalized density is defined as

$$R(x) = \alpha^{|x_E|}.$$

We restrict the GFlowNet's generative process to directed acyclic graphs (DAGs) by masking out bidirected edges. Under this condition, both the partition function $Z_H(\alpha)$ of $R(x)$ and the probability that there is an edge between each pair of nodes (the same for all of them, due to symmetry) can be computed in closed form and compared against that learned by our sampler. We notice that these quantities are actually related, and both of them can be computed efficiently by adapting Robinson's formula (Robinson, 1973), which we quickly review. Fix any pair (a, b) of nodes. Let $\mathcal{X}_H$ be the space of all DAGs with H nodes. The probability that there is an edge from a to b in the graph is

$$E_H(\alpha) = \frac{\sum_{x\in\mathcal{X},(a\to b)\text{ in }x} \alpha^{|x_E|}}{Z_H(\alpha)} \text{ with } Z_H(\alpha) = \sum_{x\in\mathcal{X}_H} \alpha^{|x_E|}.$$

To compute the latter, pick any subset of size d of nodes, which will be deemed roots of the DAG, for $d \in \{1, \ldots, H\}$; in a DAG, there is always at least one. There are $\binom{H}{d}$ ways of choosing them. From them, we may pick each of the $H - d$ remaining nodes to decide whether there is an edge between them and the root nodes, totaling a maximum of $d(H - d)$ edges. Given this choice, the remaining $H - d$ nodes may form any DAG. In our summation, the corresponding term is

$$m(d, \alpha) = \binom{H}{d} \cdot Z_{H-d}(\alpha) \cdot \sum_{0\le k\le d(H-d)} \binom{d(H-d)}{k} \alpha^k = \binom{H}{d} \cdot Z_{H-d}(\alpha) \cdot (1+\alpha)^{d(H-d)}.$$

However, summing this expression over $d \in \{1, \ldots, H\}$ over-counts certain graphs; e.g., the edgeless graph is counted several times when $k = 0$ for each d. More comprehensively, let $R_{H,d}$ be the set of all DAGs having exactly d nodes as roots, and let $c(d, \alpha) = \sum_{x\in R_{H,d}} \alpha^{|x|_E}$; our objective is to compute $\sum_{1\le d\le H} c(d, \alpha)$. However, the expression above evaluates

$$m(d, \alpha) = \sum_{d\le j\le H} \binom{j}{d} \cdot c(j, \alpha),$$

since $R_{H,j}$ appears $\binom{j}{d}$ times when evaluating $m(d, \alpha)$; to see this, fix a set $\mathrm{n} = \{n_1, \ldots, n_j\}$ of j nodes, and notice that our computation above considers the case in which every $n \in \mathrm{n}$ is root each time we pick any d-sized subset of n, i.e., $\binom{j}{d}$ times, for $d \le j$, as our computation considers DAGs with *at least* d roots. We can circumvent this issue by instead computing

$$\begin{aligned}
\sum_{1\le d\le H} (-1)^{d+1} m(d, \alpha) &= \sum_{1\le d\le H} \sum_{d\le j\le H} (-1)^{d+1} \cdot \binom{j}{d} \cdot c(j, \alpha) \\
&= \sum_{1\le j\le H} c(j, \alpha) \cdot \sum_{1\le d\le j} (-1)^{d+1} \cdot \binom{j}{d} \\
&= \sum_{1\le j\le H} c(j, \alpha) \cdot \left(1 - \sum_{0\le d\le j} (-1)^d \cdot \binom{j}{d}\right) \\
&= \sum_{1\le j\le H} c(j, \alpha) \left(1 - (1-1)^j\right) = \sum_{1\le j\le H} c(j, \alpha).
\end{aligned}$$

In conclusion,

$$Z_H(\alpha) = \sum_{1\le d\le H} (-1)^{d+1} \cdot \binom{H}{d} \cdot Z_{H-d}(\alpha) \cdot (1+\alpha)^{d(H-d)}.$$

This can be computed recursively via the boundary condition $Z_1(\alpha) = 1$. To evaluate $E_H(\alpha)$, we notice the probability of existing an edge from a to b is equal, by symmetry, to the expected number of edges divided by the total number of edges. On the other hand, the number of edges follows the exponential family $|x|_E \mapsto \exp\{|x|_E \log\alpha - \log Z_H(\alpha)\}$ with cumulant $\log Z_H(\alpha)$; as a consequence, its expected value can be computed as the derivative of the cumulant,

$$\mathbb{E}_x[|x|_E] = \frac{\mathrm{d}\log Z_H(\alpha)}{\mathrm{d}\log\alpha} = \alpha \cdot \frac{\mathrm{d}\log Z_H(\alpha)}{\mathrm{d}\alpha},$$

which can be computed efficiently through our previously discussed recursive formula. The probability of there being an edge between any pair of nodes under Erdős-Renyi's model is then

$$E_H(\alpha) = \frac{1}{H(H-1)} \cdot \alpha \cdot \frac{\mathrm{d} \log Z_H(\alpha)}{\mathrm{d}\alpha}.$$

During training, we compare $E_H(\alpha)$ with the edge-wise probabilities learned by our sampler, estimated via Monte Carlo with 2048 samples; the average L_1 between both quantities is the value reported in Figure 6, called $\mathrm{Marginal\,Edge\,Error}$ and evaluated as

$$\mathrm{Marginal\,Edge\,Error} = \frac{2}{H(H-1)} \sum_{n_1, n_2 \in [H], n_1 \neq n_2} |E_H(\alpha) - \hat{\pi}(n_1, n_2)|$$

with $\hat{\pi}(n_1, n_2)$ being the referred Monte Carlo estimate.

Autoregressive sequence generation. We set p_{length} to an uniform distribution and both p_{init} and p_{kernel} to randomly sampled distribution with logits drawn according a standard multivariate normal model of dimension $L \in \{18, 24\}$. We train the samplers for 8000 iterations in both cases.

Lazy Random Walk. We consider the follow distributions for the RINGS and GAUSSIAN MIXTURE targets, respectively,

$$\log R_{\text{Rings}}(\mathbf{x}) = \log \left(\sum_{k=1}^{K} w_k \exp \left(-\frac{1}{2} \left(\frac{\|\mathbf{x}\|_2 - r_k}{s} \right)^2 \right) \right),$$

with $K = 2$, $r_1 = 0.2$, $r_2 = 0.8$, $w_1 = 0.6$, $w_2 = 0.4$, and $s = 0.05$, and

$$\log R_{\text{GAUSSIANMIXTURE}}(\mathbf{x}) = \left[\log \left(\sum_{j=1}^{4} \exp \left(-\frac{1}{2} \left(\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|_2^2}{\sigma} \right)^2 \right) \right) - \log \left(4\pi\sigma\sqrt{2\pi} \right) \right],$$

with $\sigma = 0.1$ and $\mu_1 = (-0.8, -0.8)$, $\mu_2 = (-0.8, 0.8)$, $\mu_3 = (0.8, -0.8)$, and $\mu_4 = (0.8, 0.8)$. In both cases, we normalize $\mathbf{x}$ to $[-1, 1]$ through the transformation $\mathbf{x} \mapsto H^{-1}\mathbf{x}$.

Variable Selection. We consider $H \in \{24, 48\}$ and train both models for 5000 and 10000 iterations, respectively, and we used a batch size of 64 trajectories per training step.

H PROOFS

We provide detailed proofs for each theoretical result in both the main text and the supplement.

H.1 PROOF OF PROPOSITION 3.3

As each $\mathfrak{g}_\gamma$ abides by the MTB condition, we observe that

$$Z(\mathrm{d}\gamma) \cdot p_F^{(\gamma)}(s_o, \tau) = R(x, \mathrm{d}\gamma) \cdot p_B^{(\gamma)}(x, \tau)$$

for each $\gamma \in \Gamma$ and trajectory τ from s_o to x. We then infer that, by marginalizing $\tau \in s_o \rightsquigarrow x$,

$$Z(\mathrm{d}\gamma) \sum_{\tau \in s_o \rightsquigarrow x} p_F(s_o, \tau) = R(x, \mathrm{d}\gamma) \underbrace{\sum_{\tau \in s_o \rightsquigarrow x} p_B^{(\gamma)}(x, \tau)}_{=1} = R(x, \mathrm{d}\gamma).$$

In addition, the left-hand side of the above equation is exactly $m_F^{(\gamma)}(x)$ (by definition). Hence,

$$Z(\mathrm{d}\gamma) m_F^{(\gamma)}(x) = R(x, \mathrm{d}\gamma).$$

This shows the second equality in Proposition 3.3. By summing the above equation over $x \in \mathcal{X}$ and noticing that $m_F^{(\gamma)}$ is a probability measure on $\mathcal{X}$, we obtain

$$Z(\mathrm{d}\gamma) = Z(\mathrm{d}\gamma) \underbrace{\sum_{x \in \mathcal{X}} m_F^{(\gamma)}(x)}_{=1} = \sum_{x \in \mathcal{X}} R(x, \mathrm{d}\gamma).$$

This demonstrates the first equality in Proposition 3.3.

H.2 Proof of Corollary 5.2

Under the notation of Proposition 3.3, we observe that

$$R_k(x) = R(x, \{k\}) = R(x)q_k(x) = R(x)N(x)^{-1}$$

if $x \in \mathcal{X}_k$, and $R(x, \{k\}) = 0$ otherwise. This proves the second equality in Corollary 5.2. The first equality can be derived by recalling that $Z_k \coloneqq \sum_{x \in \mathcal{X}_k} R_k(x)$ and hence

$$Z_k = \sum_{x \in \mathcal{X}_k} \frac{R(x)}{N(x)}.$$

This is exactly the first equality in Corollary 5.2.

H.3 Proof of Proposition 5.6

We demonstrate there are numbers $\{a_k, b_k\}_{k \in [K]}$ and a function ψ for which the problem

$$\exists s' \in 2^{\mathcal{C} \setminus s} \text{s.t. } s \cup s' \in \mathcal{X} \text{ and } \psi(s \cup s') \in [a_k, b_k)$$

is equivalent to `SubsetSum`. We assume $\mathcal{C} = [C]$ for some $C \in \mathbb{N}$. We first notice that since ψ is modular, i.e.,

$$\psi(s \cup \{c\}) = \psi(s) + \psi(\{c\})$$

for each $c \in \mathcal{C} \setminus s$, we may write

$$\psi(s) = \psi(\emptyset) + \sum_{c \in s} \psi(\{c\}).$$

As such, we define $\eta_c = \psi(\{c\}) \geq 0$ for conciseness and assume $\psi(\emptyset) = 0$ without loss of generality. In this case, $\psi(s)$ can be computed in linear time in the size of s. We also assume η_c are positive integers and fix $s = \emptyset$. Under these conditions, our decision problem in Equation (6) becomes

$$\exists s \in 2^{\mathcal{C}} \colon \sum_{c \in s} \eta_c \in [a_k, b_k).$$

Let there be a k for which $b_k = a_k + 1$ with both a_k and b_k positive integers; as η_c are also integers, the above sum can only be satisfied when

$$\sum_{c \in s} \eta_c = a_k$$

for some subset s of $\mathcal{C}$. This is exactly the subset sum problem (with positive inputs), which is NP-complete (Cormen et al., 2009). In particular, deciding upon the reachability of $\mathcal{X}_k$ from a given $s \in \mathcal{S}$ is NP-complete in general.

H.4 Proof of Proposition 5.7

We show the decision problem in Equation (6) can be efficiently solved when ψ is a symmetric junta by relying upon the fact that, since $\mathcal{C}$ is finite, there are only so many values realizable by ψ, all of which can be quickly verified in polynomial time.

Fix any $s \in \mathcal{S}$ with $n = \#(s \cap r)$. We notice $n \leq \#(\mathcal{C} \cap r)$. Also, $\psi(s) = \xi(n)$ by definition. Our decision problem then becomes

$$\exists s' \in 2^{\mathcal{C} \setminus s} \colon \psi(s \cup s') \in [a_k, b_k).$$

As s and s' are disjoint, $\#((s \cup s') \cap r) = \#(s \cap r) + \#(s' \cap r)$. Similarly, $\#(s' \cap r) \leq \#(\mathcal{C} \cap r) - n$. And not only this, but $\#(s' \cap r)$ can also assume *any value* within $\{0, \ldots, \#(\mathcal{C} \cap r) - n\}$ as $s' \in 2^{\mathcal{C} \setminus s}$. As such, deciding whether the above problem is solvable is equivalent to deciding whether there is a $m \in \{0, \ldots, \#(\mathcal{C} \cap r) - n\}$ for which

$$\xi(n + m) \in [a_k, b_k)$$

which can (by definition) be executed efficiently as ψ is assumed to be efficiently evaluable in polynomial time.

To see that the result remains true if we replace $2^{\mathcal{C}}$ by the set of bounded multisubsets of $\mathcal{C}$, we define

$$\mathcal{M}(B) = \{\{\{c_1, \ldots, c_k\}\} \colon c_1, \ldots, c_k \in \mathcal{C} \text{ and } k \leq B\}$$

as the set of multisubsets of $\mathcal{C}$ of size up to B. We emphasize the operator $s \mapsto \#s$ counts repeated entries as distinct in s. We assume B is polynomial on $\#\mathcal{C}$. In this case, the above argument applies by replacing $2^{\mathcal{C}}$ with $\mathcal{M}(B)$ and noticing that it is enough to verify whether, given $s \in \mathcal{C}$, there is a $m \in \{0, \ldots, B - \#(s \cap r)\}$ for which $\xi(\#(s \cap r) + m) \in [a_k, b_k)$, which can again be done efficiently.

H.5 Proof of Proposition 5.9

Let $\{S_t^{(f)}\}_{t \geq 1}$ be the Markov chain defined by κ_F, and let L be the maximum trajectory length in the GFlowNet's state graph. For conciseness, we will denote by $\mathbb{P}_{\kappa_F}[\cdot]$ the probability measure corresponding to the case in which the forward Markov chain starts s_o, i.e., $\mathbb{P}_{\kappa_F}[\cdot|s_o]$. It is clear that $T_{\mathcal{X}} \leq L$ and, hence, $\mathbb{P}_{\kappa_F}[T_{\mathcal{X}} < \infty|s] = 1$ for any s. We now proceed to show that $S_{T_{\mathcal{X}}}^{(f)}$ is distributed according to the target $R(x)$. We start by noticing by

$$\mathbb{P}_{\kappa_F}[S_{T_{\mathcal{X}}}^{(f)} = x] = \sum_{1 \leq t \leq L} \mathbb{P}_{\kappa_F}[T_{\mathcal{X}} = t] \mathbb{P}_{\kappa_F}\left[S_{T_{\mathcal{X}}}^{(f)} = x | T_{\mathcal{X}} = t\right].$$

and, by the definition of conditional probabilities,

$$\mathbb{P}_{\kappa_F}\left[S_{T_{\mathcal{X}}}^{(f)} = x | T_{\mathcal{X}} = t\right] = \frac{\sum_{(s_1, \ldots, s_{t-1}, x) \in \bar{S}^t} \kappa_F(s_{t-1}, x) \cdot \prod_{1 \leq i \leq t-2} \kappa_F(s_i, s_{i+1})}{\sum_{x' \in \mathcal{X}} \sum_{(s_1, \ldots, s_{t-1}, x') \in \bar{S}^t} \kappa_F(s_{t-1}, x') \cdot \prod_{1 \leq i \leq t-2} \kappa_F(s_i, s_{i+1})}.$$

This equation follows from the definition of $T_{\mathcal{X}}$ and the Markov property. In addition, if there is a (s_i, s_{i+1}) for which the transition $s_i \to s_{i+1}$ is not in the underlying GFlowNet's state graph, the probability of $(s_o, \ldots, s_{t-1}, x)$ is zero by design (as κ_F matches p_F); hence, the above ratio can be written as

$$\frac{\sum_{\tau \in s_o \rightsquigarrow x, |\tau| = t} p_F(s_o, \tau)}{\sum_{x' \in \mathcal{X}} \sum_{\tau' \in s_o \rightsquigarrow x', |\tau'| = t} p_F(s_o, \tau')},$$

with $|\tau| = t$ stating that the length (number of states) of the trajectory τ is t. A similar reasoning shows that

$$\mathbb{P}_{\kappa_F}[T_{\mathcal{X}} = t] = \sum_{x' \in \mathcal{X}} \sum_{\tau \in s_o \rightsquigarrow x', |\tau| = t} p_F(s_o, \tau).$$

Thus,

$$\mathbb{P}_{\kappa_F}[S_{T_{\mathcal{X}}}^{(f)} = x] = \sum_{1 \leq t \leq L} \sum_{\tau \in s_o \rightsquigarrow x, |\tau| = t} p_F(s_o, \tau) = \sum_{\tau \in s_o \rightsquigarrow x} p_F(s_o, \tau) = m_F(x).$$

When the GFlowNet abides a balance condition, this implies

$$\mathbb{P}_{\kappa_F}\left[S_{T_{\mathcal{X}}}^{(f)} = x\right] = \frac{R(x)}{\sum_{x \in \mathcal{X}} R(x)},$$

as stated by Proposition 5.9.

Correspondingly, we observe that

$$\mathbb{P}_{\kappa_B}\left[T_{\{s_o\}} < \infty | x\right] = 1$$

since each trajectory starting at x must eventually reach s_o, and the maximum distance between s_o and any $x \in \mathcal{X}$ is L. This shows the second part of Proposition 5.9.

H.6 Proof of Lemma 5.10

We show the function h_k defined as

$$h_k(s) = \mathbb{P}_{\kappa_F}\left[T_k = \min_{k'} T_{k'} | s\right]$$

for $s \in \mathcal{S}$ is harmonic. For this, we observe that

$$\begin{aligned}\mathbb{E}_{s' \sim \kappa_F(s,\cdot)}\left[h_k(s')\right] &= \sum_{s' \in \bar{\mathcal{S}}} \kappa_F(s,s') \mathbb{P}_{\kappa_F}\left[T_k = \min_{k'} T_{k'} | s'\right] \\ &= \sum_{s' \in \bar{\mathcal{S}}} \kappa_F(s,s') \mathbb{P}_{\kappa_F}\left[T_k < \infty | s'\right] \\ &= \mathbb{P}_{\kappa_F}[T_k < \infty | s] = \mathbb{P}_{\kappa_F}\left[T_k = \min_{k'} T_{k'} | s\right] = h_k(s).\end{aligned}$$

The first equation follows from definition. The second and fourth equations are a consequence of the fact that $\mathcal{X}_k$ is an absorbing set of the Markov chain dictated by κ_F and hence that $\mathcal{X}_k$ being arrived at before $\mathcal{X}_{k'}$ for $k \neq k'$ implies that T_k is the only finite hitting time among $\{T_k\}_{k\in[K]}$. The third equation follows from the law of total probability, and the fifth equation follows from the definition of h_k. This demonstrates h_k is an harmonic function for $k \in [K]$.

The fact that $h_k(x) = 1$ for $x \in \mathcal{X}_k$ and $h_k(x) = 0$ for $x \in \mathcal{X} \setminus \mathcal{X}_k$ follows from the definition of h_k as the probability of the Markov process reaching at $\mathcal{X}_k$ before any $\mathcal{X}_{k'}$ for $k' \neq k$.

Similarly, $h_k(s_o)$ is the probability that κ_F arrives at $\mathcal{X}_k$ before arriving at any other $\mathcal{X}_{k'}$ for $k' \neq k$; as the underlying GFlowNet is assumed to abide by any of its balance conditions, the probability of the former is proportional to $Z_k = \sum_{x\in\mathcal{X}_k} R(x)$, and of the latter, proportional to $Z = \sum_{x\in\mathcal{X}} R(x)$. This shows

$$h_k(s_o) = \frac{\sum_{x\in\mathcal{X}_k} R(x)}{\sum_{x\in\mathcal{X}} R(x)} = \frac{Z_k}{Z}.$$

We also provide a direct derivation of this fact in Section H.7.

H.7 Proof of Proposition 5.11

We show that the marginal distribution of the transformed kernel $\kappa_F^{(k)}$, as defined in Theorem 5.9, matches the restriction of $R(x)$ to $\mathcal{X}_k$ up to a normalizing constant.

As before, we denote by $\mathbb{P}_{\kappa_F}[\cdot]$ the probability measure $\mathbb{P}_{\kappa_F}[\cdot|s_o]$ corresponding to the scenario in which the forward Markov chain starts at s_o. We first notice that, since h_k is harmonic,

$$\sum_{s' \in \bar{\mathcal{S}}} \kappa_F^{(k)}(s,s') = \sum_{s' \in \bar{\mathcal{S}}} \kappa_F(s,s') \cdot \frac{h_k(s')}{h_k(s)} = \frac{h_k(s)}{h_k(s)} = 1.$$

We then demonstrate that, under $\kappa_F^{(k)}$, the probability of any path that arrives at $\mathcal{X}_m$ before $\mathcal{X}_k$ for a $m \neq k$ is zero. To see this, let $(s_o, s_1 \ldots, s_{t-1}, x)$ for $x \notin \mathcal{X}_k$; then, $h_k(x) = 0$ and there is a smallest index $i \geq 1$ for which $h_k(s_i) = 0$. (By the definition of κ_F, $h_k(s_o) > 0$ for each $k \in [K]$). The transition from s_{i-1} to s_i under $\kappa_F^{(k)}$ thus has probability zero. This shows that, under $\kappa_F^{(k)}$, the Markov chain arrives at $\mathcal{X}_k$ before any $\mathcal{X}_m$ for $m \neq k$ with probability one. In particular, dropping the superscript (k) from T_m for clarity,

$$\mathbb{P}_{\kappa_F^{(k)}}\left[T_k < \infty = T_m\right] = 1.$$

This is Proposition 5.11's first statement.

To show that

$$\mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k}^{(k)} = x\right] = \frac{R(x)}{\sum_{x\in\mathcal{X}_k} R(x)},$$

we proceed as in the demonstration of Proposition 5.9. Since

$$\begin{aligned}\mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k}^{(k)} = x\right] &= \sum_{1\leq t\leq L} \mathbb{P}_{\kappa_F^{(k)}}[T_k = t] \mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k}^{(k)} = x | T_k = t\right] \\ &= \sum_{1\leq t\leq L} \mathbb{P}_{\kappa_F^{(k)}}\left[T_k = t\right] \frac{\sum_{\tau\in s_o \rightsquigarrow x, |\tau|=t} \kappa_F^{(k)}(s_o,\tau)}{\sum_{x'\in\mathcal{X}_k} \sum_{\tau'\in s_o\rightsquigarrow x', |\tau'|=t} \kappa_F^{(k)}(s_o,\tau')},\end{aligned}$$

in which we define $\kappa_F^{(k)}(s_o, \tau) = \prod_{(s,s')\in\tau} \kappa_F^{(k)}(s, s')$, and $h_k(x) = 1$ and $h_k(s_o) > 0$ for each $k \in [K]$ and $x \in \mathcal{X}_k$, a telescoping product shows that

$$\prod_{(s,s')\in\tau} \kappa_F^{(k)}(s, s') = \frac{1}{h_k(s_o)} \prod_{(s,s')\in\tau} \kappa_F(s, s') = \frac{1}{h_k(s_o)} \cdot \kappa_F(s_o, \tau) = \frac{1}{h_k(s_o)} \cdot p_F(s_o, \tau)$$

for any $\tau \in s_o \rightsquigarrow x$ with $x \in \mathcal{X}_k$. As a consequence,

$$\frac{\sum_{\tau\in s_o\rightsquigarrow x, |\tau|=t} \kappa_F^{(k)}(s_o, \tau)}{\sum_{x'\in\mathcal{X}_k} \sum_{\tau'\in s_o\rightsquigarrow x', |\tau'|=t} \kappa_F^{(k)}(s_o, \tau')} = \frac{\sum_{\tau\in s_o\rightsquigarrow x, |\tau|=t} p_F(s_o, \tau)}{\sum_{x'\in\mathcal{X}_k} \sum_{\tau'\in s_o\rightsquigarrow x', |\tau'|=t} p_F(s_o, \tau')}.$$

The rest proceeds as in Proposition 5.9. In fact, as

$$\mathbb{P}_{\kappa_F^{(k)}}[T_k = t] = \sum_{x\in\mathcal{X}_k} \sum_{\tau\in s_o\rightsquigarrow x} \kappa_F^{(k)}(s_o, \tau) = \frac{1}{h_k(s_o)} \sum_{x\in\mathcal{X}_k} \sum_{\tau\in s_o\rightsquigarrow x} p_F(s_o, \tau)$$

and

$$h_k(s_o) = \mathbb{P}_{\kappa_F}[T_k < \infty | s_o] = \sum_{x\in\mathcal{X}_x} \sum_{\tau\in s_o\rightsquigarrow x} p_F(s_o, \tau),$$

we deduce that

$$\mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k}^{(k)} = x\right] = \frac{\sum_{1\le t\le L} \sum_{\tau\in s_o\rightsquigarrow x, |\tau|=t} p_F(s_o, \tau)}{\sum_{x\in\mathcal{X}_x} \sum_{\tau\in s_o\rightsquigarrow x} p_F(s_o, \tau)}.$$

When the GFlowNet abides by a balance condition, this expression can be rewritten as

$$\mathbb{P}_{\kappa_F^{(k)}}\left[S_{T_k}^{(k)} = x\right] = \frac{R(x)}{\sum_{x\in\mathcal{X}_k} R(x)}.$$

This demonstrates the second statement in Proposition 5.11.

H.8 Proof of Proposition B.1

We show that, if our mixture model satisfies the defined SubTB condition, then it samples from the target distribution. We then show write $F(s, \mathrm{d}\gamma)$ in terms of the well-known quantities $Z(\mathrm{d}\gamma)$ and $R(x, \mathrm{d}\gamma)$.

The first statement can be derived directly from the fact that, since the condition is satisfied for all trajectories τ between reachable pairs s, s', it is in particular satisfied by *complete* trajectories going from s_o to $x \in \mathcal{X}$. This implies the trajectory balance is satisfied and, by (Malkin et al., 2022, Theorem 1), the marginal distribution of $p_M(\mathrm{d}\gamma)p_F^{(\gamma)}$ over x matches $R(x, \mathrm{d}\gamma)$. By integrating γ out, we conclude that Equation (2) is satisfied.

To see that $Z(\mathrm{d}\gamma) \coloneqq \sum_{x\in\mathcal{X}} R(x, \mathrm{d}\gamma)$, we notice that

$$F(s_o, \mathrm{d}\gamma)p_F^{(\gamma)}(s_o, \tau) = R(x, \mathrm{d}\gamma)p_B^{(\gamma)}(x, \tau)$$

for all $x \in \mathcal{X}$ and all trajectories $\tau \in s_o \rightsquigarrow x$. In particular, marginalizing out both τ and x (in sequence), we infer (as in Proposition 3.3) that

$$F(s_o, \mathrm{d}\gamma) = \sum_{x\in\mathcal{X}} R(x, \mathrm{d}\gamma) \coloneqq Z(\mathrm{d}\gamma).$$

In a similar way, as $F(s, \mathrm{d}\gamma)p_F^{(\gamma)}(s, \tau) = R(x, \mathrm{d}\gamma)p_B^{(\gamma)}(x, \tau)$, we directly infer that

$$F(s, \mathrm{d}\gamma) = \sum_{x\in\mathcal{X}} \sum_{\tau\in s\rightsquigarrow x} R(x, \mathrm{d}\gamma)p_B^{(\gamma)}(x, \tau)$$

with the convention that $\sum_{\tau\in\emptyset} \eta_\tau = 0$ for any trajectory-indexed real sequence $\{\eta_\tau\}_\tau$. This is exactly Proposition B.1's second statement.

H.9 Proof of Corollary C.2

We derive the relationship between the prior and posterior mixing distributions when they both satisfy the SubTB condition. It is clear that, since p_M, q_M, $p_F^{(\gamma)}$, and $p_B^{(\gamma)}$ abide by SubTB, they also abide by TB. In particular,

$$p_M(\gamma)p_F^{(\gamma)}(s_o,\tau) = \frac{R(x)}{Z}q_M(\gamma|x)p_B^{(\gamma)}(x,\tau).$$

As in Proposition 3.3, we marginalize over $\tau \in s_o \rightsquigarrow x$ to infer that

$$p_M(\gamma)\cdot m_F^{(\gamma)}(x) = \frac{R(x)}{Z}\cdot q(\gamma|x);$$

we then derive the second equation from Corollary C.2 by isolating $q(\gamma|x)$. Similarly, the first equation can be obtained by marginalizing over $x \in \mathcal{X}$ and recalling $m_F^{(\gamma)}(x)$ is a probability distribution over $\mathcal{X}$, i.e.,

$$p_M(\gamma) = p_M(\gamma)\sum_{x\in\mathcal{X}} m_F^{(\gamma)}(x) = \sum_{x\in\mathcal{X}}\frac{R(x)}{Z}q_M(\gamma|x).$$

H.10 Proof of Proposition C.5

Our proof has two steps. We first show that, if a model achieves an error of at most ϵ in terms of TV distance to the true policy (which is unique since we are fixing the backward policy for each γ), then the TV between the marginal distribution over $\mathcal{X}$ is bounded as a monotonically decreasing and continuous function of ϵ. Then, using Abboud et al. (2021, Theorem 1), which characterizes the universality of GNNs with random features, and the intermediate value theorem for the prior bound, we infer that our sampler can approximate any distribution with high-probability.

To start with, we temporarily simplify our notation of $p_F^{(\gamma)}$ as p, ommiting the dependence on γ. We also refer to the true solution to the chosen balance equations as $p^\star$. Our GNN approximates the distribution $p(s,\cdot)$; however, we are interested in the difference

$$\left|\sum_{\tau\in s_o\rightsquigarrow x}\prod_{(s,s')\in\tau} p(s,s') - \prod_{(s,s')\in\tau} p^\star(s,s')\right|$$

for each $x \in \mathcal{X}$. We start with the following lemma.

Lemma H.1. *Let $a_1,\dots,a_n$ and $b_1,\dots,b_n$ be sequences in $[0,1]$ such that $|a_i - b_i| \le \delta$ for some $\epsilon > 0$. Then,*

$$\left|\prod_{1\le i\le n} a_i - \prod_{1\le i\le n} b_i\right| \le n\epsilon.$$

Proof. We proceed by induction. This clearly holds for $n = 1$. To build intuition, we also show for $N = 2$. Notice that

$$|a_1a_2 - b_1b_2| = |(a_1-b_1)a_2 + b_1(a_2-b_2)| \le a_2|a_1-b_1| + b_1|a_2-b_2| \le 2\epsilon$$

since $a_1, a_2, b_1, b_2 \in [0,1]$. Assume the identity holds for n. Then,

$$\left|\prod_{1\le i\le n+1} a_i - \prod_{1\le i\le n+1} b_i\right| = \left|(a_{n+1}-b_{n+1})\prod_{1\le i\le n} a_i + b_{n+1}\cdot\left(\prod_{1\le i\le n} a_i - \prod_{1\le i\le n} b_i\right)\right|$$
$$= \epsilon + n\epsilon = (n+1)\epsilon.$$

by the induction hypothesis. This shows our desired bound. □

As there is a realization of our GNN for which $\mathrm{TV}\left(p(s,\cdot), p^\star(s,\cdot)\right) \le \epsilon$ for any $s \in \mathcal{S}$, we infer that $|p(s,s') - p^\star(s,s')| \le 2\epsilon$. Under this condition, we let T be the maximum number of trajectories from s_o to x for $x \in \mathcal{X}$ and L to be the largest trajectory length from s_o to $x \in \mathcal{X}$. By Abboud et al.

(2021, Theorem 1), there are $h \in \mathcal{H}_t$ and $\xi \in \mathcal{H}_s$ for which $p(s,\cdot)$ approximates $p^\star(s,\cdot)$ with error at most $\epsilon' = \frac{\epsilon}{2|\mathcal{X}|TL}$ and with probability $1 - \frac{\delta}{|\mathcal{X}|}$ over the realization of γ. The above difference in marginals can then be bounded as, with probability $1 - \frac{\delta}{|\mathcal{X}|}$ over the realization of γ,

$$\begin{aligned}
&\left| \left(\sum_{\tau \in s_o \rightsquigarrow x} \prod_{(s,s') \in \tau} p(s,s') - \prod_{(s,s') \in \tau} p^\star(s,s') \right) \right| \\
&\leq \left(\sum_{\tau \in s_o \rightsquigarrow x} \left| \prod_{(s,s') \in \tau} p(s,s') - \prod_{(s,s') \in \tau} p^\star(s,s') \right| \right) \\
&\leq \left(1 - \frac{\delta}{T}\right) (2TL) \frac{\epsilon}{2|\mathcal{X}|TL} = \frac{\epsilon}{|\mathcal{X}|}.
\end{aligned}$$

In other words, denoting by $m_F^{(\gamma)}$ and by $\pi(x)$ the estimated and true marginals for the learned and target models,

$$|m_F^{(\gamma)}(x) - \pi(x)| \leq \frac{\epsilon}{|\mathcal{X}|}$$

with probability $1 - \frac{\delta}{|\mathcal{X}|}$ over γ. Through an union bound[5] for $x \in \mathcal{X}$, we infer that

$$\mathrm{TV}\left(m_F^{(\gamma)}, \pi\right) = \frac{1}{2} \sum_{x \in \mathcal{X}} \left| m_F^{(\gamma)}(x) - \pi(x) \right| \leq \frac{\epsilon}{2}$$

with probability $1 - \delta$ over the sampling of γ. This demonstrates that RF GFlowNets are universal approximators of distributions over graphs when using the kind of edge-additive process we described in Proposition C.5.

H.11 Proof of Lemma C.6

We first expand $\mathcal{L}_{\mathrm{TB}}$ around $\gamma = 0$ and then take the expectation of the corresponding expansion over $\gamma \sim p_M$. To start with, let $L(\gamma) \coloneqq \mathcal{L}(\tau, \gamma; \mathfrak{g}_\Gamma)$; in this case,

$$L(\gamma) = L(0) + \gamma^T \nabla_{\gamma=0} L(\gamma) + \frac{1}{2} \gamma^T H_\gamma(\tau, 0) \gamma + \mathcal{O}(\sigma^3).$$

Since, for any matrix A,

$$\mathbb{E}_{\gamma \sim p_M}[\gamma] = 0 \text{ and } \mathbb{E}_{\gamma \sim p_M}\left[\gamma^T A \gamma\right] = \sigma^2 \cdot \mathrm{tr}(A)$$

as $\gamma^T A \gamma = \mathrm{tr}(\gamma^T A \gamma) = \mathrm{tr}(A \gamma \gamma^T)$ for any matrix A and tr is a linear operator, and in general any odd moment of γ has nulls out, we infer that

$$\mathbb{E}_{\gamma \sim p_M}[L(\gamma)] = L(0) + \frac{1}{2} \sigma^2 \cdot \mathrm{tr}\left(H_\gamma(\tau, 0)\right) + \mathcal{O}(\sigma^4),$$

which is the statement of Lemma C.6 after replacing L with $\mathcal{L}_{\mathrm{TB}}$. This characterizes the local behavior of the (population) loss in the learning objective of CI GFlowNets.

H.12 Proof of Proposition C.7

We derive from Lemma C.6 that

$$\mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_\Gamma) = \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_o) + \frac{1}{2} \cdot \sigma^2 \cdot \mathbb{E}_{\tau \sim \rho}\left[\mathrm{tr}\left(H_\gamma(\tau, 0)\right)\right] + \mathcal{O}(\sigma^4) = \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_o) + \frac{1}{2} \cdot \sigma^2 \cdot \mathcal{L}_{\mathrm{reg}}(\mathfrak{g}_\Gamma) + \mathcal{O}(\sigma^4).$$

As taking the derivative is a linear operation, $f \in \mathcal{O}(\sigma^4)$ implies $\nabla_\theta^2 f \in \mathcal{O}(\sigma^4)$ for any twice differentiable function $f \colon \Theta \to \mathbb{R}$ on a differentiable manifold Θ. In particular, we infer that, by

[5] We briefly recall that, if $P_1, \ldots, P_n$ are events with probability at least $1 - \delta$, then $\cap_i P_i$ is an event with probability at least

$$\mathbb{P}[\cap_i P_i] = 1 - \mathbb{P}[\cup_i P_i^c] \geq 1 - \sum_i \mathbb{P}[P_i^c] \geq 1 - n\delta.$$

evaluating the Hessian from both members of the above equation with respect to the parameters θ of the underlying neural network,

$$H_{\mathrm{TB}} = H_o + \frac{\sigma^2}{2} \cdot H_{\mathrm{reg}} + \mathcal{O}(\sigma^4),$$

with $H_{\mathrm{TB}} = \nabla^2_\theta \mathcal{L}_{\mathrm{TB}}(\mathfrak{g}_\Gamma)$, $H_o = \nabla^2_\theta \mathcal{L}_o(\mathfrak{g}_\Gamma)$, and $H_{\mathrm{reg}} = \nabla^2_\theta \mathcal{L}_{\mathrm{reg}}(\mathfrak{g}_\Gamma)$.

This in particular shows that the random features act as a form of *spectral regularization*, shifting the eigenvalues of H_o by $\frac{\sigma^2}{2} \cdot \mathrm{eig}(H_{\mathrm{reg}})$, with $\mathrm{eig}(H_{\mathrm{reg}})$ being the vector of eigenvalues of H_{reg}.

H.13 Proof of Lemma E.2

We show both functions h_k and g_k are harmonic on $\bar{\mathcal{S}} \setminus \mathcal{S}_k$. Let $s \in \bar{\mathcal{S}} \setminus \mathcal{S}_k$. Then,

$$\begin{aligned}
\mathbb{E}_{s' \sim \kappa_F(s,\cdot)}\left[h_k(s')\right] &= \sum_{s' \in \bar{\mathcal{S}}} \kappa_F(s,s') \cdot \mathbb{P}_{\kappa_F}\left[T_k^{(f)} = \min_{k'} T_{k'}^{(f)} < \infty | s'\right] \\
&= \sum_{s' \in \bar{\mathcal{S}}} \kappa_F(s,s') \mathbb{P}_{\kappa_F}\left[T_k^{(f)} < \infty | s'\right] \\
&= \mathbb{P}_{\kappa_F}\left[T_k^{(f)} < \infty | s\right] = h_k(s).
\end{aligned}$$

The first equation follows from the definition of h_k; the second and third equations follow from the fact that the distance from $s \in \mathcal{S}_k$ to s_o in the FHC in Definition E.1 is constant for $k \in [K]$ and, hence, if the Markov chains goes through $\mathcal{S}_k$ it will not go through any $\mathcal{S}_{k'}$ for $k' \neq k$, . A similar argument can be carried out for $g_k(s)$ and $T_k^{(b)}$.

H.14 Proof of Lemma E.3

We show that $h_k(s_o)$ and $g_k(x)$ can be interpreted as probability distributions over $k \in \{0, \ldots, K\}$. The intuition is that the Markov chain has to either hit one of $\mathcal{S}_k$ when starting at s_o, or end up in $\mathcal{X}_o$, in which case none of $\mathcal{S}_k$ are hit for $k \in [K]$; similarly, if $x \in \mathcal{X}_o$, the Markov chain following κ_B and starting at x is absorbed by s_o without going through any of the $\mathcal{S}_k$ for $k \in [K]$.

We start with $h_k(s_o)$. By Definition E.1, the events

$$\left\{T_1^{(f)} < \infty, S_1^{(f)} = s_o\right\}, \ldots, \{T_K^{(f)} < \infty, S_1^{(f)} = s_1\}, \{T_k^{(f)} = \infty \text{ for } k \in [K], S_1^{(f)} = s_o\}$$

are mutually joint. In addition, $T_k^{(f)} < \infty$ implies $T_{k'}^{(f)} = \infty$ almost surely for $k' \neq k$. As a consequence, these events also cover the space of possibilities. Hence, by the law of total probability,

$$\left(\sum_{1 \leq k \leq K} \mathbb{P}_{\kappa_F}\left[T_k^{(f)} < \infty | S_1^{(f)} = s_o\right]\right) + \mathbb{P}_{\kappa_F}\left[T_k^{(f)} = \infty \text{ for } k \in [K] | S_1^{(f)} = s_o\right] = 1.$$

This sum is exactly $h_o(s_o) + \sum_{k \in [K]} h_k(s_o)$, which demonstrates the first statement of Lemma E.3.

We proceed similarly for g_k. As noticed, if $x \in \mathcal{X}_o$, then $T_k^{(b)} = \infty$ for $k \in [K]$. Otherwise, as s_o is the only absorbing state of a Markov chain dictated by κ_B in $\bar{\mathcal{S}}$ has to eventually go through some of the $\mathcal{S}_k$ when starting at x. This means that the events

$$\{T_1^{(b)} < \infty\}, \ldots, \{T_K^{(b)} < \infty\}, \{T_k = \infty \text{ for } k \in [K]\}$$

are mutually disjoint when the Markov chain starts at $x \in \mathcal{X}$ and cover the space of possibilities. As with h_k, this implies

$$g_o(x) + \sum_{1 \leq k \leq K} g_k(x) = 1,$$

which is the second statement in Lemma E.3.

H.15 Proof of Proposition E.4

We show that, if the policies $p_F^{(k)}$ and $p_B^{(k)}$ satisfy the ATB condition from Silva et al. (2025b), then the policies $\tilde{p}_F^{(k)}$ and $\tilde{p}_B^{(k)}$ satisfy the MTB condition in Definition 3.2.

We first notice that, for any trajectory τ from s_o to $x \in \mathcal{X}_o$,

$$\tilde{p}_F^{(o)}(s_o, \tau) = \frac{h_o(x)}{h_o(s_o)} p_F^{(o)}(s_o, \tau) = \frac{1}{h_o(s_o)} \cdot p_F^{(o)}(s_o, \tau),$$

a fact that can be derived from a telescoping product and $h_o(x) = 1$ since for a Markov chain starting at $x \in \mathcal{X}_o$ and following κ_F it is true that $T_k^{(f)} = \infty$ (as none of $\mathcal{S}_k$ is reachable from $x \in \mathcal{X}_o$, which is an absorbing state). By the ATB,

$$p_F^{(o)}(s_o, \tau) = \frac{R(x)}{Z} p_B^{(o)}(x, \tau) = \frac{R(x)}{Z} \tilde{p}_B^{(o)}(x, \tau),$$

as $p_B^{(o)}(s, s') = \tilde{p}_B^{(o)}(s, s')$ for states s, s' within a distance of d of s_o; since p_B does not increase the distance to s_o, and $x \in \mathcal{X}_o$ is by definition within a distance d of s_o, it holds that $p_F^{(o)}(x, \tau) = \tilde{p}_F^{(o)}(x, \tau)$ for any $\tau \in s_o \rightsquigarrow x \in \mathcal{X}_o$. Similarly, for any trajectory going from s_o to $x \in \mathcal{X}_k$ through some $s \in \mathcal{S}_k$ (if the trajectory goes through $s \in \mathcal{S}_k$, it must eventually end up in $\mathcal{X}_k$ by definition),

$$\tilde{p}_F^{(k)}(s_o, \tau) = p_F^{(o)}(s_o, \tau_1) \cdot \frac{h_k(s)}{h_k(s_o)} \cdot p_F^{(k)}(s, \tau_2) = \frac{1}{h_k(s_o)} \cdot p_F^{(o)}(s_o, \tau_1) \cdot p_F^{(k)}(s, \tau_2),$$

with τ_1 and τ_2 being the portions of τ before and after $s \in \mathcal{S}_k$, respectively, and $h_k(s) = 1$ since $s \in \mathcal{S}_k$, and the Markov chain can only visit one of $\mathcal{S}_k$ for $k \in [K]$ since there is no positive-probability trajectory between them. Also, by the ATB,

$$p_F^{(o)}(s_o, \tau_1) p_F^{(k)}(s, \tau_2) = \frac{R(x)}{Z} p_B^{(k)}(s_o, \tau_2) \cdot p_B^{(o)}(s_o, \tau_1).$$

As before, for τ_1, it holds that $p_B^{(o)}(x, \tau_1) = \tilde{p}_B^{(o)}(x, \tau_1)$. For τ_2, on the other hand, a telescoping product shows that

$$p_B^{(k)}(x, \tau) = \tilde{p}_B^{(k)}(x, \tau) \cdot \frac{g_k(x)}{g_k(s)} = g_k(x) \cdot \tilde{p}_B^{(k)}(x, \tau).$$

In particular,

$$\tilde{p}_F(s_o, \tau) = \frac{R(x)}{Z} \cdot \frac{g_k(x)}{h_k(s_o)} \cdot \tilde{p}_B^{(k)}(x, \tau),$$

i.e., defining $Z_k = Z \cdot h_k(s_o)$ and $q(k|x) = g_k(x)$, which is a valid probability distribution by Lemma E.3, we infer that

$$Z_k \tilde{p}_F^{(k)}(s_o, \tau) = R(x) q(k|x) \tilde{p}_B(x, \tau)$$

for $k \in [K]$, $k = 0$, and every τ from s_o to $x \in \mathcal{X}_k$ through $\mathcal{S}_k$. This shows the solution to SAL is also a solution to the MTB condition in Definition 3.2.

H.16 Proof of Proposition F.4

We show that the solution $\{\mathfrak{g}_k\}_{k \geq 1}$ for the Boosted GFlowNets learning problem abides by the MTB under Assumptions F.1 and F.3.

We first demonstrate that $q(k|x)$ is a probability distribution over $k \geq 1$. To see this, we notice that

$$\sum_{1 \leq k \leq K} q(k|x) = \sum_{1 \leq k \leq K} \frac{R_k(x)}{R(x)} = \sum_{1 \leq k \leq K} \frac{R^{(k)}(x) - R^{(k-1)}(x)}{R(x)} = \frac{R^{(K)}(x)}{R(x)},$$

as $R^{(o)}(x) = 0$ by design. By Assumption F.1, $q(k|x) \geq 0$; by Assumption F.1, the limit of the above equation as $K \to \infty$ is

$$\lim_{K \to \infty} \frac{R^{(K)}}{R(x)} = \frac{R(x)}{R(x)} = 1.$$

This shows $q(k|x)$ corresponds to a probability distribution over $k \geq 1$.

We now prove that the learned policies abide by the MTB condition in Definition 3.2 having $q(k|x)$ as the posterior mixing distribution. By Assumption F.3,

$$R^{(k)}(x) = R^{(k)}(x, \tau) \coloneqq R^{(k-1)}(x) + \frac{Z_k \cdot p_F^{(k)}(s_o, \tau)}{p_B^{(k)}(x, \tau)},$$

i.e.,

$$p_B^{(k)}(x, \tau) \cdot \left(R^{(k)}(x) - R^{(k-1)}(x)\right) = Z_k \cdot p_F^{(k)}(s_o, \tau).$$

Since $R^{(k)}(x) - R^{(k-1)}(x) = R_k(x) = R(x) \cdot q(k|x)$ by definition,

$$Z_k \cdot p_F^{(k)}(s_o, \tau) = R(x)q(k|x) \cdot p_B^{(k)}(x, \tau).$$

This is Proposition F.4's second statement. It should also be clear that, since

$$Z_k = \sum_{x \in \mathcal{X}} R_k(x) = \sum_{x \in \mathcal{X}} R(x)q(k|x),$$

our prior verification shows that

$$\sum_{k \geq 1} Z_k = \sum_{k \geq 1} \sum_{x \in \mathcal{X}} R(x)q(k|x) = \sum_{x \in \mathcal{X}} R(x) \sum_{k \geq 1} q(k|x) = \sum_{x \in \mathcal{X}} R(x) = Z.$$

We can interchange the summations due to the fact that $\sum_{k \geq 1} q(k|x) = 1$ and $\mathcal{X}$ is finite. These derivations jointly highlight Boosted GFlowNets can be interpreted as DI GFlowNets having countably infinite components.

I Related works

Generative Flow Networks have been introduced in (Bengio et al., 2021; 2023) as generative models for discrete, compositional spaces such as graphs and text. They have since been applied to diverse problems, including phylogenetic inference (Zhou et al., 2024), robust scheduling (Zhang et al., 2023), Bayesian structure learning (Deleu et al., 2022; 2023), LLM fine-tuning (Hu et al., 2023; Venkatraman et al., 2024), symbolic regression (Li et al., 2023), quantitative finance (Chen et al., 2026), and drug discovery (Bengio et al., 2021; Laajil et al., 2025; Pandey et al., 2024), and have been popularized by publicly released software libraries (Tiapkin et al., 2025; Viviano et al., 2026). From a theoretical perspective, their connection to variational inference (Malkin et al., 2023) and reinforcement learning (Tiapkin et al., 2024) and Markov chains (Deleu & Bengio, 2023) has been well-established, and variants operating on cyclic graphs (Brunswic et al., 2024; Morozov et al., 2025) have also been recently considered. As with SC GFlowNets, recent works have proposed modifying the state space of the GFlowNet for improved learning, e.g., (Boussif et al., 2024; Yu et al., 2026b; Silva et al., 2025b), however, their focus lies either on improving the exploratory policy, an issue which is orthogonal to our investigations, or on developing strictly distributed algorithms requiring the training of independent models. Broadly, mixture models have been widely studied in variational and approximate inference, for instance, in semi-implicit variational inference methods (Yin & Zhou, 2018), path-sampling for Monte Carlo methods (Gelman & Meng, 1998; Liu & Liu, 2001), and in continuously indexed normalizing flows (Cornish et al., 2021; Caterini et al., 2021), the terminology of which we borrow for describing our framework. Correspondingly, recent works have proposed partitioning (binning) the continuous component of the state space of both GFlowNets (Zhou et al., 2024) and diffusion probabilistic models (Gupta et al., 2026), the outcome of which is a coarse mixture approximation to a continuous density.