Particle GFlowNets: Rethinking Generative Marginalization Models

Tiago da Silva[1] **Diego Mesquita**[2] **Salem Lahlou**[1]

[1]MBZUAI
[2]School of Applied Mathematics, Getulio Vargas Foundation

Abstract

Generative Marginalization Models (MaMs) have been recently introduced as efficient neural sampling models for any-order autoregressive modelling of discrete distributions. By learning both the marginal and conditional probabilities of a persistent-block Gibbs sampler, MaMs enable fast posterior evaluation with a single neural network forward pass. While prior work has considered MaMs to be distinct from Generative Flow Networks (GFlowNets), a well-established paradigm for inference in discrete stochastic models, we show that they are equivalent. Then, we also extend MaMs' sampling strategy to non-autoregressive generative processes. In particular, we describe an automatic criterion for full-state rejuvenation of the Gibbs sampler, derived from the Gelman-Rubin statistic, which plays a key role in speeding up learning convergence. Our experiments show that our method, called Particle GFlowNets, markedly accelerates training in large combinatorial spaces.

1 INTRODUCTION

Generative Flow Networks [GFlowNets; Bengio et al., 2021, 2023] have been used in recent years as flexible samplers for distributions over compositional, discrete objects such as sequences and sets. Notably, they were successfully applied to problems in computational biology [Jain et al., 2022, 2023a, Laajil et al., 2025], combinatorial optimization [Zhang et al., 2023a,b, Zhang and Cao, 2025], and Bayesian inference [Deleu et al., 2022, 2023], several of which seem not to be as easily solvable with other approaches. In fact, given a positive function $R(x)$ defined over a finite and combinatorial space, a GFlowNet can often generate independent samples from the corresponding probability distribution $\pi(x) \propto R(x)$ with remarkable accuracy [Shen et al., 2023]. To achieve this, GFlowNets learn a policy function for a Markov Decision Process (MDP) whose marginal distribution over terminal states matches $\pi(x)$ [Tiapkin et al., 2024]. However, training is frequently bottlenecked by the simulation of long, overlapping trajectories, requiring countless neural network forward passes for each gradient step. Generative Marginalization Models [MaMs; Liu et al., 2024] mitigate this issue by learning both the marginal and conditional distributions of a Gibbs sampler [Geman and Geman, 1984]. In doing so, each learning step requires a constant number of model evaluations—corresponding to a single Gibbs transition—regardless of the state space's size.

Contrarily to conventional understanding, we demonstrate that MaMs are equivalent to *permutation-conditioned* (PC) GFlowNets, which we introduce as a particular case of the well-established conditional GFlowNet framework [Bengio et al., 2023, Jain et al., 2023b]. Specifically, we show that MaMs' learning objective corresponds to the traditional detailed balance loss function [Bengio et al., 2023, Example 5] for PC-GFlowNets. With this in mind, the key contribution of MaMs is a computationally efficient Gibbs sampling algorithm that minimizes the number of neural network evaluations during training. Their core assumption, however, is that $R(x)$ is supported on a factorized domain (i.e., $\{1, \dots, K\}^d$), which limits MaMs' applicability in permutation-invariant (e.g., graph-structured) spaces [Liu et al., 2024, Section 3]. The central question we ask in this work is whether such an approach can be adapted to non-factorized domains. We answer it affirmatively.

For this, we propose *Particle (P) GFlowNets*. As in MaMs, a P-GFlowNet maintains a persistent Gibbs sampler over terminal states. Similarly to PC-GFlowNets, a P-GFlowNet optimizes a learning objective that enforces the detailed balance condition of the underlying Markov chain. In contrast to both methods, which were designed for autoregressive modelling, the reverse (backward) move for P-GFlowNets is non-deterministic, as the forward mapping is non-injective. To circumvent this, we use a stochastic reverse transition kernel, which can also be learned. Importantly, once

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:6366–6383.

trained, a P-GFlowNet can generate both independent or correlated samples via MDP simulation or Gibbs sampling, respectively. The crucial difference lies in the per-generation computational cost and the effective sample size of the generated samples, both of which are larger for independent sampling. Given a limited time budget, the optimal approach will likely need to be determined in a case-by-case basis.

In practice, we find that samples from P-GFlowNet's persistent Gibbs chain can become highly correlated during training, reducing the accuracy of gradient estimates and slowing down convergence. To address this limitation, we occasionally refresh the stochastic process based on the Gelman-Rubin statistics Gelman and Rubin [1992], Carpenter et al. [2017], also known as R-hat or $\hat{R}$, being above a certain threshold. We empirically show that this process, called *rejuvenation* in the probabilistic programming literature Lew et al. [2022a,b], significantly accelerates learning convergence.

Additionally, we evaluate P-GFlowNets on both established benchmarks and novel tasks for which relevant functionals can be efficiently computed and used for reliable performance assessment. Our experiments show that P-GFlowNets significantly accelerate training convergence when the cost of MDP simulation exceeds that of evaluating the target distribution, which is typical for large, long-horizon state spaces. In summary, our contributions are as follows.

1. We show MaMs can be represented as PC GFlowNets, and that their learning objective is equivalent to Bengio et al. [2023]'s expected detailed balance loss function.
2. We introduce P-GFlowNets. When compared to prior methods for GFlowNet training, P-GFlowNets asymptotically reduce the average number of forward passes per gradient step as the MDP's horizon increases.
3. We empirically demonstrate that P-GFlowNets significantly reduce wall-clock time for learning convergence.

The reader only interested in learning about P-GFlowNets may skip Section 3 and focus on Sections A, 4 and 5 instead.

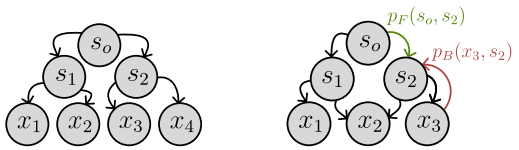
2 PRELIMINARIES

Notations. Let $\mathcal{X}$ be a discrete space. We assume $\mathcal{X}$ is *compositional*, i.e., each $x \in \mathcal{X}$ can be described as a collection of *components* $\mathcal{C}$, $x = \{c_1, \ldots, c_d\} \subseteq \mathcal{C}$, with $d \geq 1$ possibly depending on x. Often, we will consider $\mathcal{X} \coloneqq [K]^d \coloneqq \{1, \ldots, K\}^d$ for positive integers K and d, in which case we will refer to d as $\mathcal{X}$'s dimension. In this setting, $\mathcal{C} = [K] \times [d]$ and each $x = \{(i_1, 1), \ldots, (i_d, d)\}$ corresponds to the sequence $(i_1, \ldots, i_d) \in [K]^d$. We use these notations interchangeably, i.e., $x = \{(1,1),(0,2),(1,3)\}$ represents the sequence $(1,0,1)$, with $x_1 = 1$, $x_2 = 0$, and $x_3 = 1$. In particular, we write $x \equiv \{(i_j, j)\}_{j=1}^d$. In general, $\mathcal{X} \subseteq 2^{\mathcal{C}}$ is a subset of $\mathcal{C}$'s power set, $2^{\mathcal{C}}$.

Our objectives are to generate $x \in \mathcal{X}$ in proportion to a given positive measure in $\mathcal{X}$, the probability mass of which we will denote by $R\colon \mathcal{X} \to \mathbb{R}_+$, and to evaluate the marginal probability of subsets of x under R. We will call $\pi(x) \propto R(x)$ the normalized distribution. As in Bengio et al. [2021], Malkin et al. [2023], we refer to R as a *reward function*. Additionally, we will define the space $\mathcal{S}$ as

$$\mathcal{S} = \{s \in 2^{\mathcal{C}} \colon s \subset x \text{ for some } x \in \mathcal{X}\}. \tag{1}$$

Clearly, $\emptyset \in \mathcal{S}$. As in Bengio et al. [2021, 2023], we call $\mathcal{S} \cup \mathcal{X}$ the *state space*, and define $s_o \coloneqq \emptyset$ as the *initial state*. Similarly, we let $\mathcal{G} \coloneqq (\mathcal{S} \cup \mathcal{X}, \mathcal{E})$, $\mathcal{E} = \{(s, s') \colon \exists c \in \mathcal{C} \setminus s' \text{ such that } s' = s \cup \{c\}\}$, be the *state graph*. Intuitively, $s \to s'$ in $\mathcal{G}$ if s' differs from s by a single additional component. We say that the generative process is *autoregressive* when $\mathcal{G}$ is a tree rooted at s_o; see Figure 1.



(a) Autoregressive. (b) Non-autoregressive.

Figure 1: The state graph ($\mathcal{G}$) is a tree in an autoregressive setting (a). When many trajectories lead to the same state, such as $s_o \to (s_1, s_2) \to x_2$ in (b), $\mathcal{G}$ is a non-tree DAG.

GFlowNets. A GFlowNet learns a *forward* policy function $p_F\colon \mathcal{S} \times (\mathcal{S} \cup \mathcal{X}) \to [0,1]$ such that $p_F(s, \cdot)$ is a probability distribution supported on the children of s in the state graph $\mathcal{G}$. By defining $s_o \rightsquigarrow x$ as the set of trajectories in $\mathcal{G}$ from s_o to $x \in \mathcal{X}$, we seek a function p_F satisfying

$$p_\top(x) \coloneqq \sum_{\tau \in s_o \rightsquigarrow x} p_F(s_o, \tau) \propto R(x), \tag{2}$$

with $\tau = (s_o, \ldots, s_{d-1}, x)$, $s_d \coloneqq x$, and (with a slight abuse of notation) $p_F(s_o, \tau) = \prod_{i=1}^d p_F(s_{i-1}, s_i)$; we refer to d as the trajectory's length. Equation (2) represents the marginal distribution over $\mathcal{X}$ induced by $p_F(s_o, \cdot)$. When the generative process is autoregressive, $s_o \rightsquigarrow x$ contains a single trajectory τ_x, and $p_\top(x) = p_F(s_o, \tau_x)$. Otherwise, the set $s_o \rightsquigarrow x$ might be intractably large; see Figure 1. Under these conditions, we introduce a *backward policy* p_B, which is a forward policy on the transposed state graph, and approximate Equation (2) via importance sampling,

$$p_\top(x) = \mathbb{E}_{\overleftarrow{\tau} \sim p_B(x, \cdot)} \left[\frac{p_F(s_o, \overrightarrow{\tau})}{p_B(x, \overleftarrow{\tau})} \right] \propto R(x),$$

in which we use $\overrightarrow{\tau}$ and $\overleftarrow{\tau}$ to distinguish the forward and backward trajectories. (We will drop this notation in the rest of the text for clarity). By letting $Z \coloneqq \sum_{x \in \mathcal{X}} R(x)$ be the partition function of our target distribution, the above equation may be re-arranged as (dividing both sides by $R(x)$)

$$\mathbb{E}_{\tau \sim p_B(x, \cdot)} \left[\frac{Z \cdot p_F(s_o, \tau)}{R(x) \cdot p_B(x, \tau)} \right] = 1. \tag{3}$$

The condition $Z \cdot p_F(s_o, \tau) = R(x) \cdot p_B(x, \tau)$ is known as *trajectory balance* [TB; Malkin et al., 2022]. In practice, $p_F(s, \cdot)$ is parameterized as a softmax neural network receiving s as input, and we optimize both p_F and Z via stochastic gradient descent on the objective function

$$\mathcal{L}_{\text{TB}}(p_F, p_B, Z) = \mathbb{E}_{\tau \sim p_E}\left[\left(\log \frac{Z \cdot p_F(s_o, \tau)}{R(x) \cdot p_B(x, \tau)}\right)^2\right],$$

with p_E as an exploratory (full-support) policy [see, e.g., Madan et al., 2025, Kim et al., 2025]. As the reader may have noticed, p_B was chosen arbitrarily; it can be either learned [e.g., Shen et al., 2023, Gritsaev et al., 2025] or fixed [e.g., Deleu et al., 2022, Zhou et al., 2024].

Alternatively, we commonly parameterize the *flow functions*

$$F(s) = Z \sum_{\tau \in s_o \rightsquigarrow s} p_F(s_o, \tau),$$

for $s \neq s_o$ and $F(s_o) = Z$, and define the *detailed balance* (DB) learning objective $\mathcal{L}_{\text{DB}}(p_F, p_B, F)$ as

$$\mathbb{E}_{\tau \sim p_E}\left[\sum_{1 \leq i \leq d} w_i \left(\log \frac{F(s_{i-1}) p_F(s_{i-1}, s_i)}{F(s_i) p_B(s_i, s_{i-1})}\right)^2\right], \quad (4)$$

with $\tau = (s_o, \ldots, s_d)$ as in Equation (2), $F(s_d) = R(s_d)$, and $w_i > 0$ as positive weights such that $\sum_{i=1}^{n} w_i = 1$. Common choices for w_i are either uniform $w_i = 1/n$ [Bengio et al., 2023] or $w_i \propto \exp\{\lambda i\}$ for $\lambda > 0$, the intuition being that states closer to $\mathcal{X}$ should be assigned with larger weights during training [Silva et al., 2025a]. Other learning objectives have also been studied [e.g., Madan et al., 2022, Zhang et al., 2023a]; see Section C for related works.

A *conditional* GFlowNet, on the other hand, models a family of distributions $R_\omega(x)$ indexed by a parameter $\omega \in \Omega$ that is also used as input for both the policy (p_F and p_B) and flow (F) functions. Prominent applications include multi-objective combinatorial optimization, in which Ω is a simplex and R_ω represents a weighted mixture of each objective according to ω [Jain et al., 2023b, Roy et al., 2023, Zhu et al., 2023, Laajil et al., 2025], and distributed learning, with each $\omega \in \Omega$ representing a distinct subgraph of the state graph and $R_\omega(x)$ simply referring to the restriction of a given $R(x)$ to the corresponding subset of $\mathcal{X}$ [Silva et al., 2025b].

MaMs. In the context of sampling, MaMs were designed by Liu et al. [2024] for fast autoregressive modelling of discrete models. Simply put, they concomitantly learn a marginal and conditional distributions, p_θ and p_ϕ, on $[K]^d$ satisfying both a *correctness* and *consistency* conditions. Given indices $\mathcal{I}, \mathcal{J} \subseteq \{1, \ldots, d\}$ such that $\mathcal{I} \subset \mathcal{J}$ and $|\mathcal{J} \setminus \mathcal{I}| = M$, we respectively define these conditions as

$$p_\theta(x) = \frac{R(x)}{Z} \text{ and } p_\theta(x_\mathcal{I}) p_\phi(x_\mathcal{J} | x_\mathcal{I}) = p_\theta(x_\mathcal{J}), \quad (5)$$

for $x \in \mathcal{X}$. As in Liu et al. [2024], we assume $M = 1$, although our analysis can be easily extended to $M > 1$ by interpreting a group of M adjacent variables as a single variable. As with GFlowNets, p_θ and p_ϕ are parameterized as neural networks. Also, a special symbol $\triangle$ is used to represent a marginalized variable—e.g., if $x = (x_1, \ldots, x_d)$, then $x_{\{1,2\}} = (x_1, x_2, \triangle, \ldots, \triangle)$—and Z is a learned parameter. Once trained, the probability $p_\theta(x_\mathcal{I})$ of any $x \in \mathcal{X}$ and any $\mathcal{I} \subseteq \{1, \ldots, d\}$ can be evaluated in a single neural network forward pass. To enforce Equation (5), we minimize

$$\text{KL}[p_\theta || \pi] + \lambda \text{ConsistencyError}(p_\theta, p_\phi, Z), \quad (6)$$

in which $\lambda > 0$, $\text{KL}[p_\theta || \pi] \coloneqq \mathbb{E}_{x \sim p_\theta}\left[\log \frac{p_\theta(x)}{\pi(x)}\right]$ is the Kullback-Leibler (KL) divergence between p_θ and π, and ConsistencyError is defined as

$$\mathbb{E}_{x \sim q} \mathbb{E}_m \mathbb{E}_\sigma \left(\log \frac{p_\theta(x_{\sigma(<m)}) \cdot p_\phi(x_m | x_{\sigma([m-1])})}{p_\theta(x_{\sigma([m])})}\right)^2,$$

with $m \sim \mathcal{U}[d]$, $\sigma \sim \mathcal{U}(S_d)$, q as any distribution over $\mathcal{X}$, $\mathcal{U}[d]$ as an uniform distribution over $[d] = \{1, \ldots, d\}$, (hard-coded) restriction $p_\theta((\triangle, \ldots, \triangle)) = Z$, and σ uniformly picked from the space of permutations, which we denote by

$$S_d = \{\sigma \colon [d] \to [d] \mid \sigma \text{ is bijective}\}. \quad (7)$$

To generate samples from p_θ, which are needed to estimate the KL divergence in Equation (6), Liu et al. [2024] implement a persistent-block Gibbs sampling scheme having p_ϕ as the transition kernel. As we explain next, under the consistency condition in Equation (5), the resulting Markov chain has the marginal p_θ as the stationary distribution.

Gibbs sampling. Originally designed by [Geman and Geman, 1984] to sample from the Gibbs distribution [Gibbs, 1902], and also known as Successive Substitution Sampling [SSS; Schervish, 2012], Gibbs sampling creates a Markov chain by iteratively modifying each coordinate of a state x according to the conditionals of the target distribution $\pi(x)$. That is, we iteratively select $i \in [d]$ and then $x'_i \sim \pi(x_i | x_{-i})$, with $x_{-i} = x_{\{1,\ldots,i-1,i+1,\ldots,d\}}$ as in Equation (5). This process may be pictured as $(x_1, x_2) \to (x'_1, x_2) \to (x'_1, x'_2)$ for two-dimensional distributions. When i is picked at random, the algorithm is known as *random scan* Gibbs sampler; otherwise, as *systematic scan* Gibbs sampler [Owen, 2013]. We will focus on the former. The reader is invited to notice that the resulting Markov chain is stationary with respect to the target distribution π.

In the context of learning and energy-based models [Hinton, 2002, Tieleman, 2008, Liu et al., 2024], a *persistent-block* Gibbs sampler extends the canonical algorithm by (i) grouping variables into blocks, each of which is updated in a single step, and (ii) maintaining the Markov chain's state across gradient updates of the underlying neural sampling model. We notice that, in this case, the Markov chain ceases to be stationary until training converges.

3 RETHINKING MAMS AS GFLOWNETS

We now show that MaMs may be interpreted as a conditional GFlowNet (Section 3.1). In a nutshell, the conditioning space $\Omega := S_d$ will be the space of permutations defined in Equation (7), and we will demonstrate the consistency loss function is equivalent to the weighted DB loss in Equation (4) in expectation (Section 3.2).

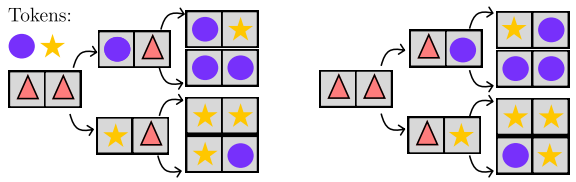
3.1 PERMUTATION-CONDITIONED GFLOWNETS

To understand the connection between MaMs and GFlowNets, we first introduce a family of *permutation-conditioned* (PC) GFlowNets for autoregressive modelling. Again, let $\mathcal{X} = [K]^d$. We define a learnable *state-embedding function* $\psi\colon \mathcal{S} \to \mathbb{R}^h$ and weights $\{\mathbf{W}_k\}_{k=1}^K$ such that $\mathbf{W}_k \in \mathbb{R}^{h\times d}$ for a given dimension h. Then, recall from Section 2 that the state space $\mathcal{S}\cup\mathcal{X}$ is simply a subset of the power set of $\mathcal{C} := [K] \times [d]$. As such, the policy function evaluated at $s^{(i)} := \{(k_1, \sigma(1)), \ldots, (k_i, \sigma(i))\} \in \mathcal{S}$ and conditioned on a permutation $\sigma \in S_d$ is defined as

$$p_F^\sigma(s^{(i)}, s^{(i+1)}) \propto \exp\{\mathbf{w}_{k,\sigma(i+1)}^T \psi(s^{(i)})\} \tag{8}$$

for $s^{(i+1)} = s^{(i)} \cup \{(k, \sigma(i+1))\}$ and $k \in [K]$. Intuitively, p_F^σ fills up a sequence up to a prescribed size (d, in this case) according to the ordering imposed by σ. We illustrate this in Figure 2. Given a σ and a $s^{(i+1)} \in \mathcal{S}$, there is only one state $s^{(i)}$ such that the transition $s^{(i)} \to s^{(i+1)}$ has positive probability under $p_F^\sigma(s_i, \cdot)$. Hence, the only choice for the backward policy abiding by the TB condition in Equation (3) is $p_B^\sigma(s^{(i+1)}, \cdot) = \delta_{s^{(i)}}$ for $s^{(i)} = \{(k_j, \sigma(j))\}_{j=1}^i$ and $s^{(i+1)} = s^{(i)} \cup \{(k_{i+1}, \sigma(i+1))\}$ with $(k_j)_{j=1}^{i+1} \in [K]^{i+1}$; $\delta_{s^{(i)}}$ is the Dirac delta at $s^{(i)}$, i.e., $\delta_{s^{(i)}}(s) = 1$ if $s = s^{(i)}$ and 0 otherwise. Thus, as p_B plays no significant role in learning, we set it aside from most of our analysis until Section 4. Our main claim in this section is that there a bijective correspondence between MaMs and PC GFlowNets.

Claim 3.1. *For each MaM, there is a unique and functionally equivalent PC GFlowNet; see Proposition 3.2.*



(a) $\sigma(1) = 1, \sigma(2) = 2$. (b) $\sigma(1) = 2, \sigma(2) = 1$.

Figure 2: **A PC GFlowNet fills up a sequence according to a given permutation** (σ). After trained, it supports fast evaluation of any-subset marginals. ($\triangle$ means no token.)

Clearly, the conditioning permutation σ only affects the navigation of $\mathcal{S}$, as both $\mathcal{X}$ and $R\colon \mathcal{X} \to \mathbb{R}_+$ remain unchanged under σ (except for re-labeling). Interestingly, however, the permutation-conditioned flow function F^σ satisfying the DB condition in Equation (4) can be exactly interpreted as the marginal probability distribution of a sequence under R. This result, outlined below, is crucial in bridging the gap between GFlowNets and MaMs.

Proposition 3.1. *Let F^σ be a flow function abiding by the DB condition of a permutation-conditioned GFlowNet, i.e.,*

$$F^\sigma(s)p_F^\sigma(s, s') = F^\sigma(s')p_B^\sigma(s', s) \tag{9}$$

and $F^\sigma(x) = R(x)$ for $x \in \mathcal{X}$. Then, denoting by $s^{(i)} = \{(k_1, \sigma(1)), \ldots, (k_i, \sigma(i))\}$ for $i \le d$,

$$F^\sigma(s^{(i)}) = \sum_{x\in\mathcal{X}\colon\, x_{\sigma([i])}=s^{(i)}_{\sigma([i])}} R(x) \text{ and } F^\sigma(s_o) = \sum_{x\in\mathcal{X}} R(x), \tag{10}$$

in which $x_{\sigma([i])} = s^{(i)}_{\sigma([i])}$ means that $x \in [K]^d$ agrees with $s^{(i)}$ on the first i coordinates according to the permutation σ, i.e., $x_{\sigma(1)} = s^{(i)}_{\sigma(1)}, \ldots, x_{\sigma(i)} = s^{(i)}_{\sigma(i)}$. In other words, $\frac{F^\sigma(s^{(i)})}{F^\sigma(s_o)}$ is the marginal distribution of $s^{(i)}$ under R.

As in Section 2, the state $s^{(i)}$ in the Proposition 3.1 can be written as $s^{(i)}_j = k_j$ for $j \in \sigma([i])$ and $s^{(i)}_j = \triangle$ otherwise.

When learning a conditional GFlowNet, we often augment the input of the neural networks parameterizing both F^σ and p_F^σ with σ [Bengio et al., 2023, Zhang et al., 2023a]. As MaMs avoid such augmentation, we ask: is it needed for PC GFlowNets? The following corollary shows that, when states s^1 and s^2 differ solely by the permutation σ, the optimal flow function in Equation (9) does not depend on σ.

Corollary 3.1. *Let $\sigma_1, \sigma_2 \in S_d$ be permutations, and define $s^{1,(i)} = \{(k_1, \sigma_1(1)), \ldots, (k_i, \sigma_1(i))\}$ and $s^{2,(i)} = \{(k_1, \sigma_2(1)), \ldots, (k_i, \sigma_2(i))\}$. Then, if i satisfies $\sigma_1([i]) = \sigma_2([i])$, i.e., σ_1 and σ_2 coincide on $[i] := \{1, \ldots, i\}$,*

$$F^{\sigma_1}(s^{1,(i)}) = F^{\sigma_2}(s^{2,(i)}).$$

To evaluate $p_F^\sigma(s^{(i)}, s^{(i+1)})$, however, information about $\sigma(i+1)$ is needed. As we show below, σ_1 and σ_2 coincide on $[i+1]$, then $p_F^{\sigma_1}(s^{(i)}, s) = p_F^{\sigma_2}(s^{(i)}, s)$ for each $s \in \mathcal{S}\cup\mathcal{X}$.

Corollary 3.2. *Let $\sigma_1, \sigma_2 \in S_d$. Define $s^{1,(i)}$ and $s^{2,(i)}$ as in Corollary 3.1. Assume $\sigma_1(\le i+1) = \sigma_2(\le i+1)$. Then,*

$$p_F^{\sigma_1}(s^{1,(i)}, \cdot) = p_F^{\sigma_2}(s^{2,(i)}, \cdot),$$

i.e., if $s^{1,(i+1)} \equiv s^{2,(i+1)}$ correspond to the same object in $([K] \cup \{\triangle\})^d$, $p_F^{\sigma_1}(s^{1,(i)}, s^{1,(i+1)}) = p_F^{\sigma_2}(s^{2,(i)}, s^{2,(i+1)})$.

In practice, we parameterize $p_F^\sigma(s^{(i)}, \cdot)$ as a neural network returning a matrix $\mathbf{P} = \mathbb{R}_+^{K\times d}$ with $\mathrm{sum}(\mathbf{P}) = 1$ representing the probability of each variable k being in the ith position, $\mathbf{P}_{k,i}$, and mask out all columns except the $\sigma(i+1)$-th one, which results in Equation (8). Drawing on the derivations above, Claim 3.1 is only a matter of bookkeeping.

Notation mapping. We recall a MaM is characterized by a marginal p_θ and conditional p_ϕ distributions. Also, for each $s = (k_1, \ldots, k_d)$ with $k_j \in [K]$ or $k_j = \triangle$ (i.e., k_j is marginalized), as described in Section 2, there is a permutation σ and an index i for which $s \equiv \{(j, \sigma(j))\}_{j=1}^{i}$ in the PC GFlowNet's state graph. By Corollary 3.1, we can unambiguously define $F^\sigma(s) \coloneqq p_\theta(s)$. Similarly, let $\mathcal{J} \supset \mathcal{I}$ be subsets of $\{1, \ldots, d\}$, and define $I = |\mathcal{I}|$. (Recall $|\mathcal{J} \setminus \mathcal{I}| = 1$). As before, given $x \in \mathcal{X}$, there is a permutation σ such that $\sigma([I]) = \mathcal{I}, \sigma([I+1]) = \mathcal{J}$, and

$$x_\mathcal{I} \equiv \{(x_{\sigma(j)}, \sigma(j))\}_{j=1}^{I} \text{ and } x_\mathcal{J} \equiv \{(x_{\sigma(j)}, \sigma(j))\}_{j=1}^{I+1}. \tag{11}$$

Under Corollary 3.2, $p_F^\sigma(x_\mathcal{I}, \cdot)$ does not depend on $\sigma(j)$ for $j > I + 1$. Again, we can thus define $p_\phi(x_\mathcal{J}|x_\mathcal{I}) = p_F^\sigma(x_\mathcal{I}, x_\mathcal{J})$ without ambiguity. Our central result, stated below, establishes that there is a unique PC GFlowNet for each MaM satisfying both the distributional (i.e., $p_\theta(x) = \pi(x)$ for $x \in \mathcal{X}$) and consistency conditions, and vice-versa.

Proposition 3.2 (MaMs are PC GFlowNets)**.** *Let (p_θ, p_ϕ) be a consistent MaM satisfying $p_\theta(x) = \pi(x)$. Then, there is a unique PC GFlowNet such that $F^\sigma(x_\mathcal{J}) = p_\theta(x_\mathcal{J})$ and $p_F^\sigma(x_\mathcal{I}, x_\mathcal{J}) = p_\phi(x_\mathcal{J}|x_\mathcal{I})$ for each triplet $\mathcal{I}, \mathcal{J}, \sigma$ as in Equation* (11)*. This PC GFlowNet satisfies the DB condition in Equation* (9)*. Conversely, a PC GFlowNet satisfying DB induces a unique consistent MaM such that $p_\theta(x) = \pi(x)$.*

From an operational viewpoint, both MaMs and PC GFlowNets allow for the evaluation of any-subset marginals with a single neural network forward pass in discrete models. In fact, the only lingering difference between MaMs and PC GFlowNets is their differing learning objectives.

As we discuss next, however, the estimator used for MaM's loss function is biased unless the consistency condition holds. Hence, it might be unsuited for training. We instead derive a novel unbiased estimator that preserves MaMs' constant number of forward passes per gradient step.

3.2 REVISITING MAMS' OBJECTIVE

At a basic level, MaMs aim to minimize the KL divergence between p_θ and π under the consistency constraint, i.e.,

$$\min_{p_\theta} \mathrm{KL}[p_\theta || \pi] \text{ such that } p_\theta(x_\mathcal{I}) p_\phi(x_\mathcal{J}|x_\mathcal{I}) = p_\theta(x_\mathcal{J})$$

for each set of indices $\mathcal{I}, \mathcal{J}$ as in Proposition 3.2. As directly enforcing p_θ to be consistent with p_ϕ is computationally unfeasible, a penalty function—the $\mathrm{ConsistencyError}$—is introduced instead; recall Equation (6). A central question, however, remains unanswered: how to generate samples from p_θ to estimate the objective function $\mathrm{KL}[p_\theta||\pi]$?

When the consistency constraint is satisfied, a persistent Gibbs sampling scheme using p_ϕ as the transition kernel would realize a Markov chain ergodic with respect to p_θ, the samples of which could be used for estimating $\mathrm{KL}[p_\theta||\pi]$. However, p_θ is only approximately consistent with p_ϕ. Thus, the generated samples would produce biased estimates of the KL objective. Based on the connection between MaMs and PC GFlowNets in Proposition 3.2, we propose instead minimizing the consistency-only objective below.

Definition 3.1. We let $p_\theta(x_\mathcal{I}) = \frac{F_\theta(x_\mathcal{I})}{Z}$ for given parametric function F_θ and $x \in \mathcal{X}$ and indices $\mathcal{I} \subseteq [d]$ with $Z \coloneqq F(x_\emptyset)$ learnable. The consistency-only objective $\mathcal{L}_{\mathrm{CO}}$ is

$$\mathbb{E}_{x\sim q, m, \sigma}\left(\log \frac{F_\theta(x_{\sigma([m-1])}) p_\phi(x_{\sigma(m)}|x_{\sigma([m-1])})}{F_\theta(x_{\sigma([m])})}\right)^2,$$

in which $\sigma \sim \mathcal{U}(S_d)$, $m \sim \mathrm{Cat}(\mathbf{w})$ for $\mathbf{w} \geq 0$ and $\sum_{i=1}^d \mathbf{w}_i = 1$, and q is a full-support distribution over $\mathcal{X}$. Additionally, we restrict $F_\theta(x) = R(x)$ for $x \in \mathcal{X}$.

The only difference between $\mathcal{L}_{\mathrm{CO}}$ and MaMs' consistency error is that we enforce $F_\theta(x) = R(x)$ for $x \in \mathcal{X}$ instead of minimizing $\mathrm{KL}[p_\theta||\pi]$. Clearly, when $\mathcal{L}_{\mathrm{CO}}$ is globally minimized, $F_\theta(x_\emptyset) = \sum_{x\in\mathcal{X}} R(x)$, which allows for direct evaluation of p_θ. Moreover, the $\mathcal{L}_{\mathrm{CO}}$ corresponds to the DB loss $\mathcal{L}_{\mathrm{DB}}$ in Equation (4) with a specific weight scheme $(w_i)_{i=1}^d$. In contrast to $\mathcal{L}_{\mathrm{DB}}$, however, a Monte Carlo approximation of $\mathcal{L}_{\mathrm{CO}}$ requires a constant number of forward passes with respect to the state space's dimension.

Proposition 3.3. *Let (F, p_F^σ) be a PC GFlowNet. (By Corollary 3.1, F does not depend on σ). Let p_E^σ be an exploratory policy such that the marginal of $p_E^\sigma(s_o, \cdot)$ over $\mathcal{X}$ matches the distribution q defined in Definition 3.1. Then,*

$$\mathop{\mathbb{E}}_{\substack{\sigma\sim\mathcal{U}(S_d),\\ \tau\sim p_E^\sigma(s_o,\cdot)}}\left[w_i \sum_{1\leq i\leq d}\left(\log \frac{F(s_{i-1}) p_F^\sigma(s_i|s_{i-1})}{F(s_i)}\right)^2\right] =$$

$$\mathop{\mathbb{E}}_{\substack{\sigma\sim\mathcal{U}(S_d),\ x\sim q,\\ i\sim\mathrm{Cat}(\mathbf{w})}}\left[\left(\log \frac{F(x_{\sigma([i-1])}) p_F^\sigma(x_{\sigma([i-1])}, x_{\sigma([i])})}{F(x_{\sigma([i])})}\right)^2\right],$$

with $\tau = (s_i)_{i=0}^d$, $s_d = x$, and $x_{\sigma([i])} = \{(x_j, \sigma(j))\}_{j=1}^i$.

Proposition 3.3 establishes that MaMs' $\mathrm{ConsistencyError}$ corresponds to a transition-wise estimator of the conventional DB loss function for conditional GFlowNets. After thoroughly outlining the connection between PC GFlowNets and MaMs, we ask: how can we leverage our results to improve GFlowNet training? We investigate this next.

4 PARTICLE GFLOWNETS

We extend the persistent-block Gibbs sampling scheme to non-autoregressive generative processes, such as set generation, in which multiple trajectories may lead to the same object. As with MaMs, this reduces the number of forward passes from $\mathcal{O}(d)$ to $\mathcal{O}(1)$ per gradient step. We also develop an automatic criterion for refreshing the Gibbs chain, which improves exploration and accelerates convergence.

Particle GFlowNets. We recall from Section 2 that a compositional object x is represented as a collection of components from a set $\mathcal{C}$, i.e., $x = \{c_1, \ldots, c_d\}$. In this scenario, a policy function induces a probability distribution over a subset of $\mathcal{C}$ conditioned on a state $s \in 2^{\mathcal{C}}$. For PC GFlowNets, $\mathcal{C} = [K] \times [d]$ and $p_F^{\sigma}(s^{(i)}, \cdot)$ induces a distribution over $[K] \times \{\sigma(i+1)\} \subset \mathcal{C}$, as in Proposition 3.1. We henceforth assume that each $x \in \mathcal{X}$ is naturally represented by exactly d components, i.e., $\mathcal{X} \subseteq \binom{\mathcal{C}}{d}$. This is often the case for usual applications and benchmarks in the GFlowNet literature, such as set generation Jang et al. [2024], phylogenetic inference [Zhou et al., 2024], design of mRNA sequences [Laajil et al., 2025], causal discovery [Silva et al., 2026], and certain combinatorial optimization tasks [Zhang et al., 2023b].

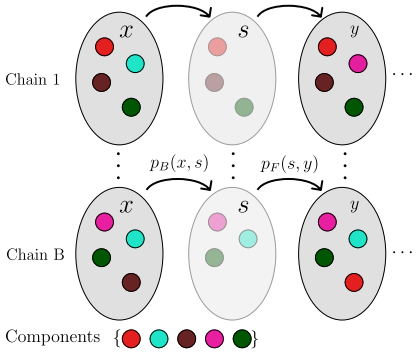


Figure 3: **A transition $x \to y$ for P-GFlowNets'** persistent Gibbs sampler. At each iteration, we replace a component of the current state x and evaluate the loss function in Equation (12) by averaging over the B stochastic processes. This requires $\mathcal{O}(1)$ forward passess, regardless of d.

To build intuition, consider Figure 3. We start by generating samples $\{x_t^b\}_{b=1}^B \subseteq \mathcal{X}$ from an untrained GFlowNet. Each x_b can be thought of as a d-sized subset of $\mathcal{C}$, for instance, $x_t^b = \{c_1^{t,b}, \ldots, c_d^{t,b}\}$. At each iteration, we remove a component $c_i^{t,b}$ from x_t^b according to $p_B(x_t^b, \cdot)$, resulting in s_t^b. We then choose a $c \in \mathcal{C}$ according to the forward policy $p_F(s_t^b, \cdot)$ and attach it to s_t^b, generating $x_{t+1}^b \coloneqq s_t^b \cup \{c\}$. In conclusion, we may update each model via a gradient step on

$$\hat{\mathcal{L}}_{\mathrm{C}} \coloneqq \frac{1}{B} \sum_{1 \le b \le B} \left(\log \frac{F(s_t^b) p_F(s_t^b, x_{t+1}^b)}{R(x_{t+1}^b) p_B(x_{t+1}^b, s_t^b)} \right)^2 .$$

Upon repetition, this produces a coupled Markov chain $\{\{x_t^b\}_{b=1}^B\}_{t \ge 1}$ satisfying the DB condition for terminal (x_t^b) and near-terminal (s_t^b) states. However, independent sampling via MDP simulation can only be achieved if the DB condition is satisfied for every intermediate state. In view of this, we also sample $s_t^{b,k} \sim p_B^k(x_t^b, \cdot)$ by removing k components from x_t^b; p_B^k denotes p_B's k-fold composition. Then, we select $s_{t+1}^{b,k} \sim p_F(s_t^{b,k}, \cdot)$ and compute

$$\hat{\mathcal{L}}_{\mathrm{I}} \coloneqq \frac{1}{B} \sum_{1 \le b \le B} \left(\log \frac{F(s_t^{b,k}) p_F(s_t^{b,k}, s_{t+1}^{b,k})}{F(s_{t+1}^{b,k}) p_B(s_{t+1}^{b,k}, s_t^{b,k})} \right)^2 .$$

In practice, we pick $k \sim \mathrm{Cat}(\mathbf{w})$ with $\mathbf{w} \in \mathbb{R}^d$ as the probabilities of a truncated Poisson distribution with average equal to $\log d$. As suggested by Proposition 3.3, this choice provides larger weight to near-terminal states, which has been shown to be beneficial [Silva et al., 2025a], while ensuring coverage of the entire state graph. We then define

$$\hat{\mathcal{L}}_{\mathrm{P}} = \hat{\mathcal{L}}_{\mathrm{C}} + \hat{\mathcal{L}}_{\mathrm{I}} \tag{12}$$

as our loss function. We refer to a model trained by minimizing $\hat{\mathcal{L}}_{\mathrm{P}}$ as a *Particle (P) GFlowNet*. Notably, we show below that the Markov chain $\{x_t\}_{t \ge 1}$ described above is ergodic with respect to the target π when the P-GFlowNet abides by the DB condition. This ensures P-GFlowNets support both correlated and independent sampling, the choice of which to use being a trade off between computational cost with statistical efficiency.

Proposition 4.1. *Let (p_F, p_B, F) be a P-GFlowNet abiding by the detailed balance condition. Define $\{x_t\}_{t \ge 1}$ as the Markov chain with the transition kernel depicted in Figure 3. Then, $\{x_t\}_{t \ge 1}$ is ergodic with respect to π.*

Importantly, our persistent Gibbs chain might suffer from inadequate state space exploration due to the structural similarity of adjacent states. To mitigate this issue, we occasionally refresh the process with fresh independent samples from the current policy $p_F(s_o, \cdot)$. This approach is discussed next.

Chain rejuvenation. We use the $\hat{R}$ metric to decide when to restart our stochastic process. Following standard statistical practice, we define $\hat{R} > 1.1$ as our condition for rejuvenation [Carpenter et al., 2017]. To understand this, recall that the $\hat{R}$ measures the discrepancy between the within- and inter-chain variances. When samples are independently generated, $\hat{R} \approx 1$. A large $\hat{R}$ indicates that the diversity of our batched stochastic process is significantly larger than that of each individual sequence, which suggests inefficient state space exploration. To account for the non-stationarity of our Gibbs sampler, we use Gelman's split $\hat{R}$ metric.

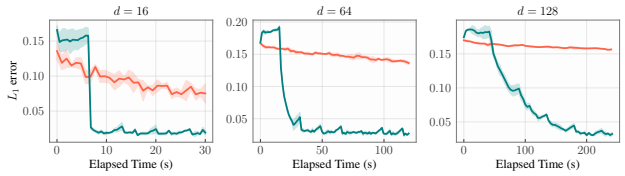
As we are dealing with discrete state spaces, however, we cannot directly compute meaningful variances. Instead, we use the last layer embeddings of the forward policy's neural network to compute the split $\hat{R}$ metric. As we show in the following section, the proposed criterion notoriously improves learning convergence and exploration.

5 EXPERIMENTS

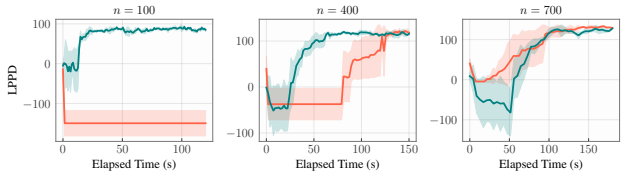
Our experimental campaign addresses the following research questions (RQs) regarding P-GFlowNets.

RQ1 Under which conditions do P-GFlowNets improve convergence relative to a standard GFlowNet?

RQ2 How effective is our approach for chain rejuvenation?



(a) Set generation with log-additive rewards.



(b) Bayesian variable selection.

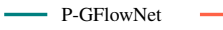


Figure 4: **P-GFlowNets often converge faster to the target distribution**. As the MDP horizon d grows (a), the relative speed up of our method increases. In contrast (b), our method grows more effective as the reward query cost decreases relative to the sampling cost—i.e., as the number n of samples for likelihood evaluation becomes smaller.

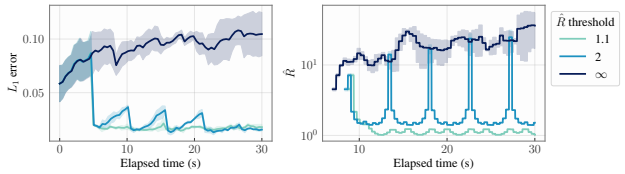


Figure 5: **Chain rejuvenation** critically accelerates learning. Left: Equation (13). Right: $\hat{R}$ throughout training.

As P-GFlowNets reduce the number of neural network forward passes for sample generation, we expect it to reduce training time when learning is bottlenecked by policy evaluation (RQ1). This is the case when trajectories are long, and sampling is expensive, and reward queries are cheaper than multiple model inferences. Our experiments confirm this intuition (see Figures 4 and 6).

We also show that chain rejuvenation does not only improve state space exploration, but significantly accelerates learning convergence (RQ2). All in all, our observations position P-GFlowNets as a compute-efficient and principled algorithm for GFlowNet training. We provide further details and computer code for our experiments in the supplement.

Set generation with log-additive rewards. Our first task consists of generating fixed-size subsets of a given set $\mathcal{C} = \{1, \dots, K\}$. Let $\mathcal{X} \coloneqq \{s \subseteq \mathcal{C} \colon |s| = d\}$ be the space of d-sized subsets of $\mathcal{C}$. Similarly, $\mathcal{S} \coloneqq \{s \subseteq \mathcal{C} \colon |s| < d\}$, and $s_o = \emptyset$ be the initial state. Given a utility function $u \colon \mathcal{C} \to \mathbb{R}_+$, we define the target distribution $R \colon \mathcal{X} \to \mathbb{R}_+$ as

$$\log R(x) = \sum_{c \in x} \log u(c).$$

This function has several important features. First, despite being factorizable into x's components, it cannot be arbitrarily well-approximated by a mean-field variational approximation. Second, the function $R(x)$ can be naturally extended to arbitrary d (state graph's diameter) and K (state graph branching's factor) with $d \leq K$. Third, if $\mathbf{1}_s$ denotes s's indicator function, both the partition function $Z \coloneqq \sum_{x \in \mathcal{X}} R(x)$ and the marginals $p_c \coloneqq \frac{1}{Z} \sum_{s \in \mathcal{X}} R(x) \cdot \mathbf{1}_s(c)$ for $c \in \mathcal{C}$ can be efficiently computed in $\mathcal{O}(K \cdot d^2)$ through the following dynamic programming algorithm, which may be of independent interest for GFlowNet evaluation. Let $\text{first}_k(S)$ denote the first k elements of any $S \subseteq \mathcal{C}$ in a fixed order, and define

$$Z^c_{k,i} = \sum_{\substack{s \subseteq \text{first}_k(\mathcal{C} \setminus \{c\}) \\ |s| = i}} \prod_{e \in s} u(e) \text{ and } Z_{k,i} = \sum_{\substack{s \subseteq \text{first}_k(\mathcal{C}) \\ |s| = i}} \prod_{e \in s} u(e)$$

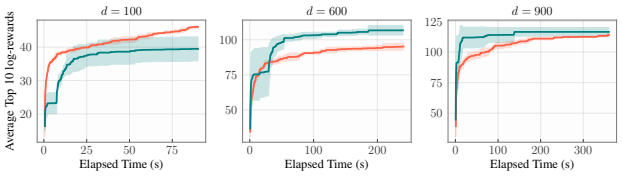
for $k \in \{1, \dots, K\}$ and $i \in \{1, \dots d\}$. It should be clear that if $P^c_{k,i} = u(c) \cdot \frac{Z^c_{k-1,i}}{Z_{k,i}}$ then $p_c = P^c_{K,d}$. Also, $Z_{k+1,i} = u(k+1) \cdot Z_{k,i-1} + Z_{k,i}$, with a similar recurrence equation for $Z^c_{k,i}$ obtained by replacing $u(k+1)$ by the appropriate $(k+1)$-th element of $\mathcal{C} \setminus \{c\}$. As such, we gauge the accuracy of a trained GFlowNet on this task by drawing N independent states $\{\{x_1, \dots, x_N\}\} \subseteq \mathcal{X}$ and computing

$$\frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} |p_c - \hat{p}_c| \text{ with } \hat{p}_c = \frac{1}{N} \sum_{1 \leq n \leq N} \mathbf{1}_{x_n}(c). \qquad (13)$$

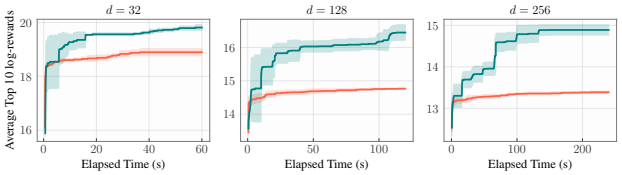
We consider $(d, K) \in \{(32, 64), (64, 128), (128, 256)\}$ and $\log u(c) \sim \mathcal{N}(0, 1)$ drawn from a standard Gaussian for $c \in \mathcal{C}$. Then, we track the reduction in the above metric in terms of wall-clock time in Figure 4. As d increases, the runtime gap between P-GFlowNet and a standard GFlowNet widens.

Additionally, Figure 5 shows our rejuvenation approach is paramount for speeding up training. There, we consider $(d, K) = (64, 128)$ and the conditions $\hat{R} > \alpha$ for $\alpha = 1.1$ (default), $\alpha = 2$, and $\alpha = \infty$ (i.e., no rejuvenation).

Bayesian variable selection. When reward query costs offset the sampling overhead, we expect that allocating more compute per sample should speed up learning convergence. Indeed, this has been empirically observed by Madan et al. [2025], Kim et al. [2025], Dall'Antonia et al. [2026]. To understand how this behavior affects the training of P-GFlowNets, we consider the problem of Bayesian variable selection [George and McCulloch, 1993] with progressively larger sample sizescorresponding to increasingly expensive-to-evaluate posterior distributions. Let $\mathbf{X} \in \mathbb{R}^{n \times K}$ and $\mathbf{y} \in \mathbb{R}^n$ be a dataset with n samples and K-dimensional features, and denote by $\mathbf{X}_F \in \mathbb{R}^{n \times |F|}$ the column-filtered data with $F \subseteq \{1, \dots, K\}$. Also, let $\mathbf{I}_n$ (resp. $\mathbf{I}_{|F|}$) is n-dimensional (resp. $|F|$-dimensional) identity matrix, $F \sim \text{Multinomial}(\{1, \dots, K\}, \pi)$ indicates each F is sampled with probability $\pi^{|F|}(1 - \pi)^{K - |F|}$ for



(a) Ising model.



(b) Bit sequences.

P-GFlowNet GFlowNet

Figure 6: **P-GFlowNets improve exploration.** We show the average log-reward of the 10 most rewarding states found in training. As in Figure 4, the runtime gains from P-GFlowNets as the MDP horizon (d) grows.

$\pi \in [0, 1]$. We then consider the linear model

$$\mathbf{y}|\beta, F \sim \mathcal{N}(\mathbf{X}_F\beta_F, \eta^2\mathbf{I}_n) \text{ with}$$
$$\beta_F|F \sim \mathcal{N}(0, \nu^2\mathbf{I}_{|F|}) \text{ and } F \sim \text{Multinomial}([K], \pi).$$

We assume $\pi, \eta > 0$, and $\nu > 0$ are known. Under these conditions, the marginal posterior distribution over the set F is

$$\log R(F) = \log \mathcal{N}(\mathbf{y}|0, \nu^2\mathbf{X}_F\mathbf{X}_F^T + \eta^2\mathbf{I}_n) + |F| \log \frac{\pi}{1-\pi},$$

with $\mathcal{N}(\mathbf{y}|\mu, \Sigma)$ denoting the density of a Gaussian distribution with mean μ and covariance Σ. Clearly, evaluating $R(F)$ costs $\mathcal{O}(K \cdot n^2)$. To assess a GFlowNet, we evaluate the log-predictive posterior density (LPPD) of a held-out dataset $(\mathbf{X}^\star, \mathbf{y}^\star)$ throughout training. Given samples $\{\{F_1, \ldots, F_N\}\}$, we approximate the LPPD as

$$\text{LPPD} = \log \frac{1}{N} \sum_{1 \le n \le N} p(\mathbf{y}^\star|\mathbf{X}^\star, \mathbf{y}, F),$$

with $p(\mathbf{y}^\star|\mathbf{y}, F) = \mathcal{N}(\mathbf{y}^\star|\mathbf{X}_F\mu_F, \mathbf{X}_F^\star\Sigma_F\mathbf{X}^T + \eta^2\mathbf{I}_n)$ and $\mu_F = \eta^{-2}\Sigma_F\mathbf{X}_F^T\mathbf{y}$ and $\Sigma_F = (\eta^{-2}\mathbf{X}_F^T\mathbf{X}_F + \nu^{-2}\mathbf{I}_{|F|})^{-1}$ as β_F posterior's mean and covariance given F, respectively. In practice, each row of $\mathbf{X}_F$ is independently sampled from $\mathcal{N}(\mathbf{0}, \Gamma)$, with $\mathbf{0} \in \mathbb{R}^K$ and $\Gamma \in \mathbb{R}^{K \times K}$ and $\Gamma_{ij} = \gamma^{|i-j|}$ for $\gamma = 0.8$. Under this model, $\mathbf{X}$'s columns are highly correlated, and the posterior distribution over F is multimodal.

Notably, Figure 4 shows that P-GFlowNets consistently outperform a standard GFlowNet when n small. However, as n grows, the posterior evaluation cost outpaces that of trajectory sampling, and the computational benefits from our model dwindle. As explained above, this confirms our initial assumptions regarding P-GFlowNets.

We next consider the tasks of Ising model simulation, also present in [Liu et al., 2024], and of bit generation, a common testbed for GFlowNets [e.g., Viviano et al., 2023]. For each example, $\mathcal{X}$ is the space of d-sized sequences with elements from $\{-1, 1\}$ and $\{1, 0\}$, respectively. As is standard practice Malkin et al. [2022], Pan et al. [2023], Kim et al. [2025], we evaluate a model by measuring the average log reward for the top 10 most valuable samples found throughout training.

Ising model. Simply put, let $\mathbf{J} \in \mathbb{R}^{d \times d}$ and $\mathbf{h} \in \mathbb{R}^d$. We define an energy function as $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\mathbf{J}\mathbf{x} - \mathbf{h}^T\mathbf{x}$, and $p(x) \propto \exp\{-E(x)/\beta\}$ as the probability of a configuration $x \in \{-1, 1\}^d$ under a temperature $\beta > 0$. In Figure 6, we consider $d \in \{100, 300, 900\}$. We observe that P-GFlowNets drastically improve exploration when trajectories are long and sampling is consequently expensive.

Bit sequences. As in Malkin et al. [2022], Tiapkin et al. [2024], we let $\mathcal{M} \subseteq \{1, 0\}^d$ be a set of *modes*, and define $\beta \log R(x) = 1 - \min_{m \in \mathcal{M}} \rho(m,x)/d$, with $\rho(x, m)$ as edit distance between x and m and $\beta > 0$ is a temperature parameter. We consider $d \in \{64, 128, 256\}$ and $\beta = 1/20$. As we consistently observed in prior experiments, Figure 6 shows P-GFlowNets significantly speed up the discovery of high-reward states as the MDP horizon grows.

6 DISCUSSION

We showed that MaMs [Liu et al., 2024], which were previously thought to be distinct from GFlowNets, can be seen as an instantiation of a conditional GFlowNet using a persistent Gibbs sampler for exploration during training. Based on this, we also demonstrated this strategy generalizes beyond autoregressive modelling, for which MaMs were originally designed, while maintaining its computational benefits. Our experiments highlighted that the resulting method, called Particle GFlowNets, significantly accelerated learning convergence in terms of wall-clock time when compared against conventional GFlowNet training algorithms.

From a broader perspective, our work (esp. Propositions 3.3 and 4.1) strengthens the connection between GFlowNets and Markov chain methods, which was also formally studied by Deleu and Bengio [2023]. This raises several questions. How to optimally decide when to rejuvenate the persistent Gibbs sampler? As noted in Silva et al. [2025a], diagnosing GFlowNets is strikingly difficult; can we draw inspirations from the MCMC literature to properly assess the distributional accuracy of GFlowNets? Successful Markov samplers, such as Langevin dynamics-based methods [Welling and Teh, 2011, Girolami and Calderhead, 2011], rely on simulating a latent dynamics for each transition of the underlying stochastic process; is such a technique extensible to GFlowNets, and when does it accelerate training? We believe these to be interesting directions for future research.

Acknowledgements

DM acknowledges the support of the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) (SEI-260003/020348/2025, SEI-260003/020694/2025) and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (404336/2023-0, 305692/2025-9, 445170/2024-7).

References

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

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.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 2017.

Sanghyeok Choi, Sarthak Mittal, Víctor Elvira, Jinkyoo Park, and Nikolay Malkin. Reinforced sequential monte carlo for amortised sampling, 2025. URL https://arxiv.org/abs/2510.11711.

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

Tristan Deleu and Yoshua Bengio. Generative flow networks: a markov chain perspective, 2023.

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.

Walter M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20(4):406–416, 1971.

Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 1992.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6), November 1984. doi: 10.1109/tpami.1984.4767596.

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

Josiah Willard Gibbs. *Elementary Principles in Statistical Mechanics: Developed with Especial Reference to the Rational Foundation of Thermodynamics*. Charles Scribner's Sons, New York, 1902. Reprinted by Dover Publications (1960) and others.

Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2011.

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.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14 (8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL https://doi.org/10.1162/089976602760128018.

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

Edward J. Hu, Nikolay Malkin, Moksh Jain, Katie Everett, Alexandros Graikos, and Yoshua Bengio. Gflownet-em for learning compositional latent variable models, 2023b. URL https://arxiv.org/abs/2302.06576.

Rui Hu, Yifan Zhang, Zhuoran Li, and Longbo Huang. Beyond squared error: Exploring loss design for enhanced training of generative flow networks, 2024. 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.

Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2023a.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective GFlowNets. In *International Conference on Machine Learning (ICML)*, 2023b.

Hyosoon Jang, Minsu Kim, and Sungsoo Ahn. Learning energy decompositions for partial inference in GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024.

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/.

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, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search gflownets, 2024. URL https://arxiv.org/abs/2310.02710.

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*, 2025. URL https://openreview.net/forum?id=BdmVgLMvaf.

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*, pages 18269–18300. PMLR, 2023.

Elaine Lau, Nikhil Vemgal, Doina Precup, and Emmanuel Bengio. Dgfn: Double generative flow networks, 2023. URL https://arxiv.org/abs/2310.19685.

Alexander K. Lew, Monica Agrawal, David Sontag, and Vikash K. Mansinghka. Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming, 2022a. URL https://arxiv.org/abs/2007.11838.

Alexander K. Lew, Marco Cusumano-Towner, and Vikash K. Mansinghka. Recursive monte carlo and variational inference with auxiliary variables, 2022b. URL https://arxiv.org/abs/2203.02836.

Sulin Liu, Peter J Ramadge, and Ryan P Adams. Generative marginalization models. In *International Conference on Machine Learning (ICML)* , 2024.

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.

Idriss Malek, Aya Laajil, Abhijith Sharma, Eric Moulines, and Salem Lahlou. Loss-guided auxiliary agents for overcoming mode collapse in gflownets, 2025. URL https://arxiv.org/abs/2505.15251.

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.

Sean P Meyn and Richard L Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, Cambridge, 2nd edition, 2009. Cambridge Mathematical Library.

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. In *International Conference on Machine Learning (ICML)*, 2023.

Ling Pan, Moksh Jain, Kanika Madan, and Yoshua Bengio. Pre-training and fine-tuning generative flow networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Julien Roy, Pierre-Luc Bacon, Christopher Pal, and Emmanuel Bengio. Goal-conditioned gflownets for controllable multi-objective molecular design. *arXiv preprint arXiv:2306.04620*, 2023.

Mark J Schervish. *Theory of statistics*. Springer Science & Business Media, 2012.

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.

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.

Tiago Silva, Bruna Bazaluk, Eliezer da Silva, António Góis, Salem Lahlou, Dominik Heider, Samuel Kaski, Diego Mesquita, and Adele Ribeiro. Expert-aided causal discovery of ancestral graphs. *SSRN*, 01 2026. doi: 10.2139/ssrn.6074306.

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

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, Alexandre Adam, Jarrid Rector-Brooks, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. 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. *arXiv e-prints*, pages arXiv–2305, 2023.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011.

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

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial optimization problems with gflownets. In *NeurIPS (NeurIPS)*, 2023b.

Ni Zhang and Zhiguang Cao. Hybrid-balance GFlownet for solving vehicle routing problems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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.

Mingyang Zhou, Zichao Yan, Elliot Layne, Nikolay Malkin, Dinghuai Zhang, Moksh Jain, Mathieu Blanchette, and Yoshua Bengio. Phylogfn: Phylogenetic inference with generative flow networks, 2023.

Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Chang-Yu Hsieh, Tingjun Hou, and Jian Wu. Sample-efficient multi-objective molecular optimization with gflownets, 2023.

Particle GFlowNets: Rethinking Generative Marginalization Models (Supplementary Material)

Tiago da Silva[1] **Diego Mesquita**[2] **Salem Lahlou**[1]

[1]MBZUAI
[2]School of Applied Mathematics, Getulio Vargas Foundation

A SUMMARY & PSEUDOCODE

Table 1: Summary of our main theoretical results.

Result	Name	Statement
Prop. 3.1	Flow = Marginal	The flow at any partial state equals the sum of rewards over all completions consistent with that state.
Cor. 3.1	σ-Independence of Flow	The optimal flow does not depend on the variable ordering σ used to construct the DAG.
Cor. 3.2	σ-Independence of Policy	The forward policy is likewise permutation-independent, so the network need not receive σ as input.
Prop. 3.2	MaMs $\equiv$ PC-GFlowNets	Consistent MaMs and PC-GFlowNets satisfying detailed balance are equivalent.
Prop. 3.3	Consistency Error = DB Loss	The MaM consistency error is an unbiased, $O(1)$-cost estimator of the weighted detailed balance training loss.
Prop. 4.1	P-GFlowNet Ergodicity	The persistent Markov chain induced by a P-GFlowNet satisfying detailed balance converges to $\pi \propto R$.

The main purpose of our work is to show that MaMs and GFlowNets, previously thought to be distinct, are the same—with MaM differing from conventional GFlowNet implementations solely in how states are sampled during training. Building on this equivalence, we introduce P-GFlowNets, demonstrating its correctness and computational efficiency as a sampling model for discrete, compositional spaces. In this context, Table 1 summarizes our central claims regarding the equivalence between P-GFlowNets and MaMs (Propositions 3.1 and 3.2, Corollaries 3.1 and 3.2) and the correctness of P-GFlowNets (Propositions 3.3 and 4.1). We also provide a pseudocode description of P-GFlowNets in Algorithms 1 and 2.

Algorithm 1: Transition kernel for Algorithm 2.

Procedure TransitionKernel(x)
  /* One step of the persistent Gibbs chain (used to produce $y^{(b)}$ in $\hat{\mathcal{L}}_C$) */
  Sample $s \sim p_B(\cdot \mid x)$; // remove one component via backward policy
  Sample $x' \sim p_F(\cdot \mid s)$; // add one component via forward policy
  return x';

Algorithm 2: P-GFlowNet Training Loop

Input: Reward function R, batch size B, GR threshold $\hat{R}_{\text{thr}}$, weight vector $\mathbf{w}$ (truncated Poisson, mean $\log d$)
Output: Trained parameters $\theta = (p_F, p_B, F)$

/* Initialisation */

Sample particles $\{x^{(b)}\}_{b=1}^{B} \sim p_F(\cdot \mid s_o)$ via MDP simulation;

while not converged **do**

/* Complete loss $\hat{\mathcal{L}}_C$ (terminal / near-terminal transitions) */

for $b = 1$ **to** B **do**

$s^{(b)} \sim p_B(\cdot \mid x^{(b)})$; // remove one component

$y^{(b)} \sim p_F(\cdot \mid s^{(b)})$; // add one component

$\hat{\mathcal{L}}_C \leftarrow \frac{1}{B}\sum_{b=1}^{B}\left(\log \frac{F(s^{(b)})\,p_F(s^{(b)}, y^{(b)})}{R(y^{(b)})\,p_B(y^{(b)}, s^{(b)})}\right)^2$;

/* Intermediate loss $\hat{\mathcal{L}}_I$ (interior transitions, depth sampled from $\mathbf{w}$) */

for $b = 1$ **to** B **do**

Sample depth $k^{(b)} \sim \mathrm{Cat}(\mathbf{w})$; // truncated Poisson, mean $\log d$

$s^{(b,k)} \sim p_B^{k^{(b)}}(\cdot \mid x^{(b)})$; // remove k components

$s'^{(b,k)} \sim p_F(\cdot \mid s^{(b,k)})$; // add one component

$\hat{\mathcal{L}}_I \leftarrow \frac{1}{B}\sum_{b=1}^{B}\left(\log \frac{F(s^{(b,k)})\,p_F(s^{(b,k)}, s'^{(b,k)})}{F(s'^{(b,k)})\,p_B(s'^{(b,k)}, s^{(b,k)})}\right)^2$;

/* Gradient step */

$\hat{\mathcal{L}}_P \leftarrow \hat{\mathcal{L}}_C + \hat{\mathcal{L}}_I$;

$\theta \leftarrow \theta - \eta \nabla_\theta \hat{\mathcal{L}}_P(\theta)$;

/* Particle update via Gibbs step */

for $b = 1$ **to** B **do**

$x^{(b)} \leftarrow y^{(b)}$; // advance persistent chain

/* Rejuvenation (Gelman–Rubin criterion) */

Compute split $\hat{R}$ from last-layer embeddings of p_F over the 512 most recent states;

if $\hat{R} > \hat{R}_{\text{thr}}$ **then**

Resample all particles: $x^{(b)} \sim p_F(\cdot \mid s_o)$ for $b = 1, \ldots, B$;

return θ;

B PROOFS

B.1 PROOF OF PROPOSITION 3.1

We first show that $F^\sigma(s_o)$ equals the partition function of Z and, in particular, does not depend on σ. To see this, notice that Equation (9) implies

$$F^\sigma(s_o) p_F^\sigma(s_o, \tau_{\sigma,x}) = F^\sigma(x) := R(x) \tag{14}$$

for the only trajectory $\tau_{\sigma,x}$ starting at s_o and finishing at $x \in \mathcal{X}$ with positive probability under p_F^σ (as explained earlier, $p_B^\tau(s, \cdot)$ is either 1 or 0). Also, it should be clear that

$$\sum_{x \in \mathcal{X}} p_F^\sigma(s_o, \tau_{\sigma,x}) = \sum_{\tau \in s_o \rightsquigarrow \mathcal{X}} p_F^\sigma(s_o, \tau) = 1,$$

as there is only one trajectory from s_o to each x and, separating the sum according to the terminal state $x \in \mathcal{X}$ of each τ, this is exactly $\sum_{x\in\mathcal{X}} p_\top(x)$ for the $p_\top$ introduced in Equation (2). Then, when we sum Equation (14) over $x \in \mathcal{X}$,

$$F^\sigma(s_o) = \sum_{x\in\mathcal{X}} R(x) \coloneqq Z.$$

This shows $F^\sigma(s_o)$ does not depend on σ. Similarly, to show that Equation (10) is satisfied, we proceed by induction in i. For this, let $\mathrm{T}(\sigma, s^{(i)}) = \{x \in \mathcal{X} \colon x_{\sigma([i])} = s^{(i)}_{\sigma([i])}\}$. Then, for $i = d$, $s^{(d)} \in \mathcal{X}$, $\mathrm{T}(\sigma, s^{(d)}) = \{s^{(d)}\}$ and $F^\sigma(s^{(d)}) = R(x)$ by definition. Assume, for $i < d$, that $F(s^{(i+1)}) = \sum_{x\in\mathrm{T}(\sigma,s^{(i+1)})} R(x)$ for each $(i+1)$-sized object $s^{(i+1)}$. Then, summing over the support of $p^\sigma_F(s_i, \cdot)$ in the DB condition in Equation (9), we observe that

$$F^\sigma(s^{(i)}) = \sum_{k\in[K]} F^\sigma(s^{(i)} \cup \{(k, \sigma(i+1))\}).$$

Clearly, $s^{(i),k} = s^{(i)} \cup \{(k, \sigma(i+1))\}$ corresponds to an $(i+1)$-sized sequence. Also, the sets $\mathrm{T}(\sigma, s^{(i),k})$ are disjoint for $k \in [K]$ as, if $x \in \mathrm{T}(\sigma, s^{(i),k}) \cap \mathrm{T}(\sigma, s^{(i),k'})$, then $x_{\sigma(i+1)} = k$ and $x_{\sigma(i+1)} = k'$. Additionally, $\mathrm{T}(\sigma, s^{(i)}) = \bigcup_{k\in[K]} \mathrm{T}(\sigma, s^{(i),k})$ (see Figure 2). By induction,

$$\begin{aligned} F^\sigma(s^{(i)}) &= \sum_{k\in[K]} F^\sigma(s^{(i),k}) \\ &= \sum_{k\in[K]} \sum_{x\in\mathrm{T}(\sigma,s^{(i),k})} R(x) = \sum_{x\in\mathrm{T}(\sigma,s^{(i)})} R(x). \end{aligned}$$

In particular,

$$\frac{F^\sigma(s^{(i)})}{F^\sigma(s_o)} = \sum_{x\in\mathrm{T}(\sigma,s^{(i)})} \frac{R(x)}{Z},$$

which is exactly the marginal distribution of $s^{(i)}$ under R.

B.2 PROOF OF COROLLARY 3.1

This follows from Proposition 3.1. In fact, notice that $\mathrm{T}(\sigma_1, s^{1,(i)}) = \mathrm{T}(\sigma_2, s^{2,(i)})$, as if $x \in \mathrm{T}(\sigma_1, s^{1,(i)})$, then $x_{\sigma([i])} = s^{1,(i)}_{\sigma([i])} = s^{2,(i)}_{\sigma([i])}$, and so $x \in \mathrm{T}(\sigma_2, s^{2,(i)})$, and vice-versa. As a consequence,

$$\begin{aligned} F^{\sigma_1}(s^{1,(i)}) &= \sum_{x\in\mathrm{T}(\sigma_1,s^{1,(i)})} R(x) \\ &= \sum_{x\in\mathrm{T}(\sigma_2,s^{2,(i)})} R(x) = F^{\sigma_2}(s^{2,(i)}). \end{aligned}$$

B.3 PROOF OF COROLLARY 3.2

This follows from Corollary 3.1. As both $p^{\sigma_1}_F$ and $p^{\sigma_2}_F$ satisfy the DB condition, for $j \in \{1, 2\}$,

$$p^{\sigma_j}_F(s^{j,(i)}, s^{j,(i+1)}) = \frac{F^{\sigma_j}(s^{j,(i+1)})}{F^{\sigma_j}(s^{j,(i)})},$$

with $s^{j,(i+1)} = s^{j,(i)} \cup \{(k, \sigma_j(i+1))\}$ for some $k \in [K]$, and zero otherwise. As $\sigma_1(\leq i+1) = \sigma_2(\leq i+1)$, Corollary 3.1 ensures that $F^{\sigma_1}(s^{1,(i)}) = F^{\sigma_2}(s^{2,(i)})$ and $F^{\sigma_1}(s^{1,(i+1)}) = F^{\sigma_2}(s^{2,(i+1)})$, and then

$$p^{\sigma_1}_F(s^{1,(i)}, \cdot) = p^{\sigma_2}_F(s^{2,(i)}, \cdot).$$

B.4 PROOF OF PROPOSITION 3.2

This proposition follows directly from Corollaries 3.1 and 3.2, and the notational mapping discussed in Section 3.1. That said, we provide a comprehensive demonstration below.

($\implies$) Let (p_θ, p_ϕ) be a consistent MaM satisfying $p_\theta(x) = \pi(x)$ for all x. Then, define a PC-GFlowNet (p_F^σ, F^σ) as follows. For each $\mathcal{J} \subseteq \{1, \dots, d\}$, let $\sigma_\mathcal{J}$ be a permutation for which the first $|\mathcal{J}|$ elements are $\mathcal{J}$. For $i \in \{1, \dots, d\} \setminus \mathcal{J}$, let $\sigma^i_\mathcal{J}$ be a permutation for which the first $|\mathcal{J}|$ elements are $\mathcal{J}$, and the $(|\mathcal{J}|+1)$-th element is i.

In this context, let $F^{\sigma_\mathcal{J}}(x_\mathcal{J}) = p_\theta(x_\mathcal{J})$. Also, let $\mathcal{I} = \mathcal{J} \cup \{i\}$ and $p_F^{\sigma^i_\mathcal{J}}(x_\mathcal{J}, x_\mathcal{I}) = p_\phi(x_\mathcal{I}|x_\mathcal{J})$, with $x_\mathcal{I}$ and $x_\mathcal{J}$ as in Equation (11).

By Corollaries 3.1 and 3.2, the above construction does not depend on the choice of permutation σ. Consequently, (p_F^σ, F^σ) is equivalent to the MaM (p_θ, p_ϕ).

($\impliedby$) Conversely, let (p_F^σ, F^σ) be a PC-GFlowNet abiding by the DB condition. Let $\sigma^i_\mathcal{J}$, $\sigma^i_\mathcal{J}$, and $\mathcal{I}$ be defined as above, and let

$$p_\theta(x_\mathcal{J}) = F^{\sigma_\mathcal{J}(x_\mathcal{J}) \text{ and } p_\phi(x_\mathcal{I}|\mathcal{J})} = p_F(x_\mathcal{J}, x_\mathcal{I}).$$

By Corollaries 3.1 and 3.2, again, the definition above does not depend on permutation $\sigma_\mathcal{J}$ and $\sigma^i_\mathcal{J}$, as long as they satisfy the property that the first $|\mathcal{J}|$ elements are $\mathcal{J}$, and the $(|\mathcal{J}|+1)$-th is i.

Consequently, (p_θ, p_ϕ) implements the PC-GFlowNet (p_F^σ, F^σ).

This shows the equivalence.

B.5 PROOF OF PROPOSITION 3.3

To see this, let x be τ's terminal state. First, notice that p_E^σ simply denotes a(exploratory) policy such that $p_E(s, \cdot)$ and $p_F^\sigma(s, \cdot)$ have the same support for each $s \in \mathcal{S}$. By the definition of PC GFlowNets, each τ has size d and $s_i = x_{\sigma([i])}$ for $i \in \{0, \dots, d\}$. For conciseness, define

$$\Delta(x, \sigma, i) \coloneqq \left(\log \frac{F(x_{\sigma([i-1])}) p_F^\sigma(x_{\sigma([i])}|x_{\sigma([i-1])})}{F(x_{\sigma([i])})} \right)^2.$$

Then, the LHS of Proposition 3.3 can be written as

$$\mathop{\mathbb{E}}_{\tau \sim p_E^\sigma(s_o, \cdot)} \left[\sum_{1 \le i \le d} w_i \Delta(x, \sigma, i) \right] = \mathop{\mathbb{E}}_{x \sim q, i \sim \mathrm{Cat}(\mathbf{w})} [\Delta(x, \sigma, i)],$$

as the marginal of p_E^σ matches q. This is exactly the inner expectation of Proposition 3.3's RHS.

B.6 PROOF OF PROPOSITION 4.1

To ensure that $\{x_t\}$ is ergodic, we show that (i) it is irreducible (i.e., every state is reachable from every other state), (ii) aperiodic (i.e., the chain does not return to x_t at regular intervals), and (iii) stationary with respect to π [Meyn and Tweedie, 2009].

By definition of both $\mathcal{X}$ and $\mathcal{S}$, and since neither p_F or p_B are degenerate (i.e., they do not assign zero probability to valid transition), the chain $\{x_t\}$ is irreducible. Additionally, for any t, we can return to x_t with positive probability after any number of steps. Hence, the chain is aperiodic.

To see that the chain is stationary with respect to π, we first recall that the detailed balance implies

$$F(s) p_F(s, x) = R(x) p_B(x, s)$$

for any $x \in \mathcal{X}$ and $s \in \mathcal{S}$. Denote by $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ the transition kernel of $\{x_t\}$. Our objective is to show that

$$R(x)\kappa(x, x') = R(x')\kappa(x', x),$$

which implies $\{x_t\}$ is stationary with respect to $\pi(x) \propto R(x)$. For this, we notice that

$$\kappa(x, x') = \sum_{s \in \mathcal{S}} p_B(x, s) p_F(s, x').$$

Hence,

$$\begin{aligned} R(x)\kappa(x, x') &= \sum_{s \in \mathcal{S}} R(x) p_B(x, s) p_F(s, x') \\ &= \sum_{s \in \mathcal{S}} F(s) p_F(s, x) p_F(s, x') \\ &= \sum_{s \in \mathcal{S}} F(s) p_F(s, x) p_F(s, x') \\ &= \sum_{s \in \mathcal{S}} R(x') p_B(x', s) p_F(s, x) \\ &= R(x')\kappa(x', x); \end{aligned}$$

we highlight in teal and in blue the terms for which we apply the DB condition. This shows $\{x_t\}$ is stationary with respect to π. Taken together, our results ensure $\{x_t\}$ is ergodic with respect to π.

C RELATED WORKS

GFlowNets [Bengio et al., 2021, Lahlou et al., 2023, Bengio et al., 2023] are a topic of major interest in the probabilistic modelling literature, providing a clear framework for reasoning about complex hierarchical variational approximations of discrete stochastic models [Malkin et al., 2023, Choi et al., 2025]. The central challenge hampering GFlowNets' broader applicability, in our opinion, is that they are often notoriously difficult to train. A prominent research direction for mitigating this problem, in fact, has been the development of effective learning objectives [Madan et al., 2022, Malkin et al., 2022, Hu et al., 2024, Pan et al., 2024, 2023] that accelerate training convergence. As learning efficiency is characterized by not only the objective function, but also by which samples are observed throughout training, another significant recent line of research has explored meta-heuristic approaches for enhanced state space exploration [Lau et al., 2023, Kim et al., 2025, Madan et al., 2025, Malek et al., 2025, Dall'Antonia et al., 2026]. In particular, our method operationally resembles Kim et al. [2024], Hu et al. [2023b], both of which introduce a sampling technique based on repeated applications of forward and backward policies; however, only P-GFlowNets asymptotically reduces the number of forward passes per gradient step as a function of the underlying MDP's horizon. Nonetheless, despite significantly enhancing GFlowNet's sample efficiency, these approaches often incur in a significant computational cost due to expensive exploration strategies requiring numerous policy evaluations per sample. In fact, our evaluation of SA-GFlowNets, Adaptive Teachers GFlowNets, and ACE Madan et al. [2025], Kim et al. [2025], Dall'Antonia et al. [2026], alongside Local Search GFlowNets Kim et al. [2023], suggested an increase of up to $4\times$ in the per-sample processing time when compared against Madan et al. [2022], Malkin et al. [2022]'s traditional algorithms, which remain standard in the GFlowNet literature, e.g., [Hu et al., 2023a, Venkatraman et al., 2024, Zhou et al., 2023]. As in MaMs [Liu et al., 2024], our work addresses a fundamentally different problem: contrarily to prior approaches, we assume reward querying is cheap, and exploration cost is dominated by policy evaluation. This is the dominant setting for Bayesian inference over complex models in large state spaces, e.g., Deleu et al. [2022], wherein the policy p_F is frequently parameterized with inference-intensive models such as a transformer Vaswani et al. [2023]. With this in mind, as discussed in Section 6, we believe the ideal algorithm would adaptively provision the appropriate amount of computation for each batch of samples. How such an approach would have to be implemented, however, remains open.

D EXPERIMENTAL DETAILS

Our models were implemented in JAX [Bradbury et al., 2018]. All our experiments were run in a Apple MacBook Pro, Apple M4 (10-core: 4P + 6E), 16 GB unified memory, macOS 26.2 (Tahoe). In Section 5, the GFlowNet was trained by minimizing the TB loss Malkin et al. [2022]. We used Muon optimizer Jordan et al. [2024] for minimizing the learning objectives for both GFlowNets and P-GFlowNets. As in Madan et al. [2022], we used a learning rate of 10^{-3} for p_F and of 10^{-2} for F and Z; p_B was fixed as an uniform policy. For each experiment, we implemented a 2-layer MLP with 256 hidden units for parameterizing the policy network, and fixed $B = 64$ for the batch size. We trained each model with a fixed time budget shown in Figures 4 to 6. In particular, we also set $\nu = \eta = 10^{-1}$ for the task of Bayesian variable selection and,

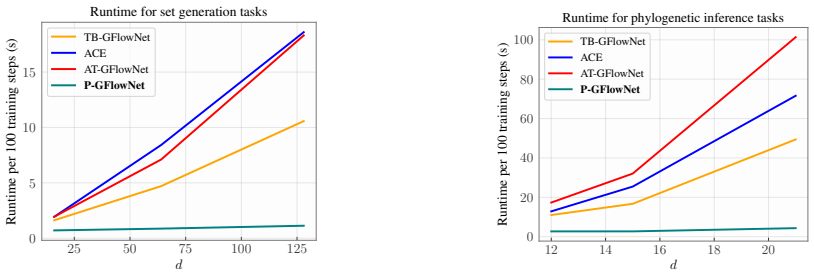
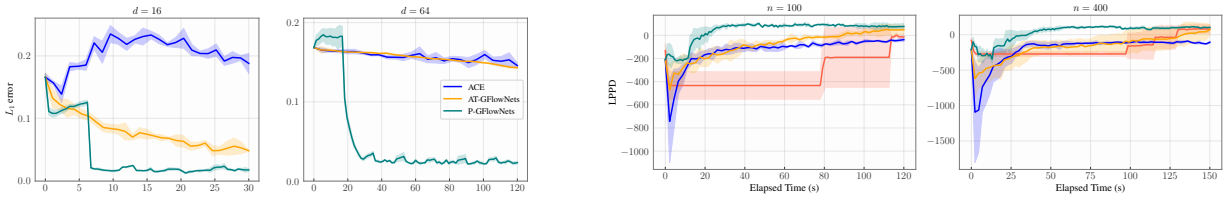


Figure 8: **Per-training step runtime for ACE, AT, TB, and P-GFlowNets (ours)**. By circumventing complete trajectory sampling during training, P-GFlowNets reduce per-step computation cost by several orders of magnitude.



(a) L_1 error for set generation.

(b) LPPD for variable selection.

Figure 9: **P-GFlowNets converge faster than AT, ACE, and TB GFlowNets** in the set generation and variable selection tasks, achieving more accurate marginals (a) and larger log-predictive density (b), respectively, within a shorter time span.

for the bit sequence generation, followed Malkin et al. [2022]'s approach for constructing the set of modes $\mathcal{M}$. All plots show the average across 3 independent runs; error bars represent one standard deviation from the average. We periodically evaluated the $\hat{R}$ asynchronously based on the 512 most recently observed states, following the implementation in Stan [Carpenter et al., 2017], rejuvenating the chain when the computed $\hat{R}$ exceeded 1.1.

E ADDITIONAL EXPERIMENTS

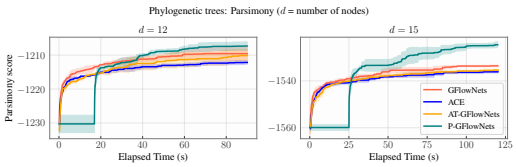


Figure 7: **P-GFlowNets finds more parsimonious** phylogenetic trees than ACE, AT GFlowNets and TB GFlowNets.

To further evaluate P-GFlowNets, we compare it against the recently proposed Adaptive Teachers (AT) Kim et al. [2025] and the Adaptive Complementary Exploration (ACE) Dall'Antonia et al. [2026] training algorithms. We also confirm P-GFlowNets' effectiveness in the phylogenetic inference task Zhou et al. [2024].

Comparison against AT and ACE. In contrast to P-GFlowNets, whose focus lies on reducing the per-step computational cost for both training and inference, AT and ACE aim at improving a GFlowNet's sample efficiency by training an exploratory model to search for highly informative (e.g., unvisited, high-probability under the target) regions during learning. In doing so, however, training cost often increases multifold due to the evaluation of both the exploratory and target GFlowNets on both forward and backward trajectories; see Figure 8. From this perspective, we found that P-GFlowNets converge significantly faster than AT and ACE with respect to wall-clock time. We show illustrate this in Figure 9 for both the set generation and variable selection tasks. In both cases, we followed the implementations of Dall'Antonia et al. [2026] for both AT and ACE GFlowNets.

The main setting for which these artificial curiosity-inspired strategies would be appropriate, in our opinion, is when the reward function is extremely expensive to evaluate relatively to the policy network *and* this expensiveness cannot be reduced by exploiting the shared structure of adjacent states through caching, as explained next.

Phylogenetic inference. To further evaluate P-GFlowNets, we also consider the problem of phylogenetic inference using the parsimony score as the log-reward function Fitch [1971]. We adopt the algorithm suggested by Zhou et al. [2024]. Given

a phylogenetic tree T with leaves L annotated with $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$, let $\mathbf{e}_n \in \{1, 0\}^4$ be the one-hot encoding of node n. For n in L, $\mathbf{e}_n^{(1)} = 1$ if n is annotated with A; $\mathbf{e}_n^{(2)} = 1$, if T, and etc. Otherwise, $\mathbf{e}_n^{(j)} = 0$. Then, by letting $\mathrm{LC}(n)$ and $\mathrm{RC}(n)$ denote the left and right child of n, the parsimony score is recursively defined as

$$\mathbf{e}_n = \begin{cases} \mathbf{e}_{\mathrm{LC}(n)} \wedge \mathbf{e}_{\mathrm{RC}(n)} \text{ if } \sum_{i=1}^{4} (\mathbf{e}_{\mathrm{LC}(n)} \wedge \mathbf{e}_{\mathrm{RC}(n)})^{(i)} \geq 1 \\ \mathbf{e}_{\mathrm{LC}(n)} \vee \mathbf{e}_{\mathrm{RC}(n)}, \text{ otherwise.} \end{cases}$$

$$\mathrm{ParScore}(n, \mathrm{T}) = \mathrm{ParScore}(\mathrm{LC}(n), \mathrm{T}) + \mathrm{ParScore}(\mathrm{RC}(n), \mathrm{T}) + [\mathbf{e}_n \neq \mathbf{e}_{\mathrm{LC}(n)} \wedge \mathbf{e}_{\mathrm{RC}(n)}],$$

with the boundary condition $\mathrm{ParScore}(n, \mathrm{T}) = 0$ for $n \in \mathrm{L}$ and $\mathbf{u}, \mathbf{v} \mapsto [\mathbf{u} \neq \mathbf{v}]$ being 1 if $\mathbf{u}$ and $\mathbf{v}$ match element-wise and 0 otherwise. The parsimony score of a tree whose leaves are annotated with a sequence of $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$ is the sum of the parsimony score of a tree whose leaves are solely annotated with each element in this sequence, corresponding to the usual bag-of-words assumption in phylogenetic statistical models. The intuition is that a tree is as parsimonious (having low parsimony score) as the number of disagreements between a parent and its children. We define $R(\mathrm{T}) = \exp\{-\mathrm{ParScore}(\mathrm{r}(\mathrm{T}), \mathrm{T})\}$ as our reward function, with $\mathrm{r}(\mathrm{T})$ as the root of T.

Importantly, the modular nature of $\mathrm{ParScore}$ ensures that, in transitioning from T to T' using P-GFlowNets' backward-forward kernels, most of the computation required for $\mathrm{ParScore}(\mathrm{T}')$ can be reused from $\mathrm{ParScore}(\mathrm{T})$, reducing the cost of reward querying. Based on this, we compare P-GFlowNets against AT and ACE GFlowNets on their state space exploration capabilities during training in Figure 7. The initial exploration phase, highlighted as a horizontal line for P-GFlowNets, corresponds to the period before which samples are collected for computing $\hat{R}$, which is used to decide whether the chain should be rejuvenated. As in the set generation and variable selection tasks, P-GFlowNets improve upon both AT and ACE given a similar wall-clock time budget. That said, extending P-GFlowNets to mixed, discrete and continuous, spaces—as often required in phylogenetic inference based on stochastic evolution models—remains an interesting venue for future research.