Learning to Scale Logits for Temperature-Conditional GFlowNets

Minsu Kim [*1,2] **Joohwan Ko** [*2] **Taeyoung Yun** [*2] **Dinghuai Zhang** [3,4] **Ling Pan** [5] **Woo Chang Kim** [2]
Jinkyoo Park [2] **Emmanuel Bengio** [6] **Yoshua Bengio** [3,4,7]

Abstract

GFlowNets are probabilistic models that sequentially generate compositional structures through a stochastic policy. Among GFlowNets, temperature-conditional GFlowNets can introduce temperature-based controllability for exploration and exploitation. We propose *Logit-scaling GFlowNets* (Logit-GFN), a novel architectural design that greatly accelerates the training of temperature-conditional GFlowNets. It is based on the idea that previously proposed approaches introduced numerical challenges in the deep network training, since different temperatures may give rise to very different gradient profiles as well as magnitudes of the policy's logits. We find that the challenge is greatly reduced if a learned function of the temperature is used to scale the policy's logits directly. Also, using Logit-GFN, GFlowNets can be improved by having better generalization capabilities in offline learning and mode discovery capabilities in online learning, which is empirically verified in various biological and chemical tasks. Our code is available at https://github.com/dbsxodud-11/logit-gfn

1. Introduction

Generative Flow Networks (GFlowNets) (Bengio et al., 2021) offer a training framework for learning generative policies that sequentially construct compositional objects to be sampled according to a given unnormalized probability mass or reward function. Whereas other generative models are trained to imitate a distribution implicitly specified by a training set, GFlowNets' target distribution is specified by a reward function seen as an unnormalized probability mass function. The primary advantage inherent to GFlowNets is their capacity to uncover a multitude of highly rewarded samples from a diverse set of modes (Bengio et al., 2021) of the given target distribution. This holds great significance in the context of scientific discovery, exemplified by domains such as drug discovery (Bengio et al., 2021; Jain et al., 2022; 2023b).

Temperature-conditional GFlowNets are training frameworks for learning conditional generative models that generate samples proportional to a tempered reward function: $p(x|\beta) \propto R(x)^\beta$. In contrast to typical GFlowNets trained to match a single target distribution with fixed (inverse) temperature β, temperature-conditional GFlowNets learn a family of generative policies corresponding to reward functions raised to different powers. The major benefit of temperature-conditional GFlowNets is that it can adjust the generative policy based on the β, thereby allowing for the management of the exploration-exploitation trade-off. Furthermore, temperature-conditional GFlowNets effectively handle the training difficulties encountered with typical GFlowNets at lower temperatures (i.e., high β). These lower temperatures yield distributions that are extremely selective, posing a challenge for effective training using unconditional GFlowNets. Our expectation is that temperature-conditional GFlowNets, when trained at higher temperatures (low β values), should be able to infer the distributions at lower temperatures (high β values) more effectively.

Temperature-conditional GFlowNets have already been introduced (Zhang et al., 2022c; Zhou et al., 2023) and have shown promising results for specific neural architectures, such as in the case of *Topoformer* (Gagrani et al., 2022) for solving scheduling problems (Zhang et al., 2022c). Nonetheless, the empirical research on generic architectures remains limited. This limitation in research underscores the need for meticulous customization of temperature-conditional architectures to suit each specific task. Moreover, incorporating varying temperature parameters in the training process of neural networks has been observed to create challenges. These arise from the altered gradient profiles corresponding to each target temperature distribution, leading to training instability. For example, a higher temperature results in a

[*]Equal contribution [1]Work performed while the author was at the Mila – Québec AI Institute [2]Korea Advanced Institute of Science and Technology [3]Mila – Québec AI Institute [4]Université de Montréal [5]Hong Kong University of Science and Technology [6]Recursion [7]CIFAR. Correspondence to: Minsu Kim <minsu@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

peakier reward landscape, which poses challenges for stably training GFlowNets (Malkin et al., 2022a). Thus, there is an urgent need for thorough empirical research into the behavior of temperature-conditional GFlowNets coupled with developing stable architectures to overcome these training challenges.

In this paper, we first suggest a new generic architecture design of temperature-conditional GFlowNets, called *Logit-scaling GFlowNets* (Logit-GFN), to obtain a simple yet stable training framework. Our key idea is to integrate a direct pathway into the architecture. This pathway adjusts the policy's logits according to the parameter β, providing an effective inductive bias that adapts the target distribution's temperature. We hypothesize and verify experimentally that the Logit-GFN, which directly adjusts the logits in the sampling policy through temperature input, enhances the generalization and speeds up the training of temperature-conditional GFlowNets.

Additionally, we introduce an online discovery algorithm that utilizes Logit-GFN (i.e., $p(x|\beta)$) along with a dynamic control policy, $P_{\text{exp}}(\beta)$. The algorithm samples β from the policy during online learning exploration. Then, we marginalize the Logit-GFN to train with decisions across multiple GFNs derived from different target distributions. This approach facilitates the discovery of new compositional structures within various tempered distributions, eliminating the need to train individuals GFN at different temperatures in every iteration. In our empirical study, we explore stationary distributions of $P_{\text{exp}}(\beta)$, including uniform, log uniform, normal, and dynamic distributions with simulated annealing, offering insights into the exploration-exploitation trade-off.

In our experimental results, the Logit-GFN architecture significantly enhances training stability, characterized by a smooth and rapid loss convergence. Moreover, it demonstrates improved offline generalization capability, showcasing its adeptness at generating novel and highly rewarded samples from fixed datasets. Our online learning with the Logit-GFN stands out with superior performance compared to GFN and alternative benchmarks, including well-established techniques in Reinforcement Learning (RL) (Schulman et al., 2017; Haarnoja et al., 2017) and Markov Chain Monte Carlo (MCMC) methods (Xie et al., 2020).

2. Related Works

Most generative AI approaches require a dataset to represent a target distribution to sample from, while GFlowNets are instead provided with an unnormalized probability mass, which we can consider as a reward function from an RL perspective. Methods for training and applying GFlowNets are rapidly evolving, demonstrating considerable progress across diverse domains, including causal discovery (Deleu et al., 2022; 2023), combinatorial optimization (Zhang et al., 2022c; 2023), biochemical discovery (Jain et al., 2022; 2023a), reinforcement learning (Tiapkin et al., 2024) with world modeling (Pan et al., 2023c), large language model inference (Hu et al., 2024), and diffusion-structured generative models (Zhang et al., 2022a; Lahlou et al., 2023; Zhang et al., 2024a). The enhancements in GFlowNets are primarily attributed to advancements in training objectives, credit assignment techniques, and improved exploration strategies. The advent of temperature-conditional GFlowNets represents an exciting frontier in research, offering significant potential to elevate the performance and versatility of GFlowNets. In this context, we provide a comprehensive survey covering these directions.

Training objective of GFlowNets. GFlowNets, introduced by Bengio et al. (2021), open interesting new avenues. One core feature is the ability to have a greater diversity of modes of the target distribution compared with existing RL, variational methods, or MCMC. The other one is their off-policy training objectives: they can be trained on examples and trajectories from any distribution with full support, not necessarily from the distribution corresponding to their current parameters (because that would not allow sufficient exploration and diversity) or from the target distribution (for which one may not have samples). Recent notable advances include loss functions like trajectory balance (TB) (Malkin et al., 2022a), subtrajectory balance (SubTB) (Madan et al., 2023), and quantile matching (QM) (Zhang et al., 2024b).

Local credit assignment for GFlowNets. Several approaches have been developed to enhance GFlowNets' training efficiency. Forward-Looking GFlowNet (FL-GFN) (Pan et al., 2023a) calculates intermediate energy from states, contributing to more stable training over longer trajectories. Building on this, the transition-based GFlowNet (Zhang et al., 2023) specializes in combinatorial optimization. Additionally, Jang et al. (2024) focus on decomposing energy into partial energies, achieving more effective local credit assignment compared with FL-GFN. Finally, Falet et al. (2024) utilize the inductive bias of sparse graphical models to avoid having to ever evaluate full trajectories and terminal states, facilitating rapid learning in probabilistic graphical models.

Better exploration for GFlowNets. Another research focus is enhancing the exploration capabilities of GFlowNets. Pan et al. (2023b) propose a framework for introducing augmented flows into the flow network, which are represented by intrinsic curiosity-based incentives to encourage exploration in sparse environments. Rector-Brooks et al. (2023) apply Thompson sampling (Osband et al., 2016) into the training of GFlowNets, while Shen et al. (2023) have advocated for prioritized replay training, a method that directs GFlowNets updates to concentrate on regions with higher

rewards. Additionally, Kim et al. (2024) incorporates a back-and-forth refining process (Zhang et al., 2022b) into its training algorithm. This integration of local search strategies significantly improves the quality of local exploration in GFlowNets, leading to the acquisition of higher-rewarded samples during training.

Temperature conditional GFlowNets. Temperature conditioning, applied in combinatorial scheduling problems (Zhang et al., 2022c), uses a variable temperature factor to modulate the scheduling objective's smoothness. In the Topoformer architecture, Gagrani et al. (2022) implement this through matrix multiplication with the temperature parameter in the initial linear layer. Similarly, Zhou et al. (2023) adopt temperature-conditional GFlowNets for phylogenetic inference, presenting a new approach to Bayesian variational inference with GFlowNets. Although these applications show promising results, there is a lack of concrete empirical evidence confirming the specific contribution of and issues with temperature conditioning. Recent studies in contrastive learning (Qiu et al., 2023) and the development of a temperature prediction network for large foundation models (Qiu et al., 2024) have also investigated the issue of learning context-dependent temperature values. Our work, focusing on the impact of temperature conditioning through extensive empirical research, introduces a novel generic architecture aimed at stabilizing the training of temperature-conditional GFlowNets.

3. Preliminaries

This section introduces GFlowNets and temperature conditional GFlowNets more formally; see Figure 1 for conceptual understanding and see Bengio et al. (2023) for a full introduction. Generative flow networks (GFlowNets) constitute a class of deep generative models and a reinforcement learning (RL) methodology designed to sample compositional objects $x \in \mathcal{X}$, given a target distribution specified by an unnormalized probability mass or positive reward function. A generated object corresponds to the *terminal state* in a Markov decision process starting from a unique initial state s_0. The GFlowNets policy sequentially adds an *action* into a partial object (called the *state*); the *complete trajectory* $\tau = (s_0 \to \cdots \to s_n = x)$, which leads to compositional object x, is sequentially generated by the policy through a corresponding sequence of actions, each conditioned on the current state.

The space of all possible state sequences (trajectories) from the initial state to a terminal state is specified by a directed acyclic graph (DAG) that can incorporate domain-specific constraints (about which action is allowed in any state). It is noteworthy that, in opposition to a tree structure, the more general DAG structure offers a pivotal advantage by enabling the modeling of numerous potential action sequences that converge to identical states. This is different from soft Q-learning and entropy-regularized RL (Haarnoja et al., 2017; 2018; Buesing et al., 2020), which are closely related to GFlowNets but may misbehave in the DAG setting where there are multiple ways of landing in the same terminal state (Bengio et al., 2021).

Definition for flows. The *trajectory flow*, denoted by $F(\tau)$, is a non-negative function that maps complete trajectories to unnormalized probabilities, representing the flow of probability from the initial state to a terminal state along that trajectory, with the idea that the total flow into a terminal state x should match the reward function value at x, $R(x)$.

This flow is further dissected into *state flows* and *edge flows*. The state flow, represented by $F(s)$, is the sum of $F(\tau)$ over all trajectories τ containing state s. Similarly, $F(s \to s')$ is the sum of flows of trajectories with a transition from state s to state s'.

Relationship between flow and policy. The *forward policy*, $P_F(s_{t+1}|s_t)$, quantifies the probability of transitioning from a state to any of its child states, while the *backward policy*, $P_B(s_t|s_{t+1})$, captures the probability of transitioning from a state to one of its parent states. When F is a *Markovian flow*, the forward policy and backward policy can derived as follows: $P_F(s'|s) = F(s \to s')/F(s)$ and $P_B(s|s') = F(s \to s')/F(s')$.

Marginal distribution of GFlowNets. Eventually, we define the key concept, a marginal distribution, denoted as $P_F^\top(x) := \sum_{\tau \to x} P_F(\tau)$, which aggregates probabilities of trajectories terminating at a specific state x. The learning problem of GFlowNet is approximately achieving the following conditions:

$$P_F^\top(x) = \sum_{\tau \to x} P_F(\tau) = \frac{R(x)}{\sum_{x \in \mathcal{X}} R(x)}, \tag{1}$$

where $R(x) > 0$ refers to a reward for a terminal state x.

Learning objective of GFlowNets. Equation (1) can be satisfied by minimizing a GFlowNet training objective. The most commonly used objective is trajectory balance (TB) (Malkin et al., 2022b). Given the GFlowNet with parameters θ, the core principle of TB mandates that the flow of a trajectory computed forward (from the initial state to a terminal state) must match the flow computed backward (from a terminal state to the initial state):

$$Z_\theta \prod_{t=1}^{n} P_F(s_{t+1}|s_t;\theta) = R(x) \prod_{t=1}^{n} P_B(s_t|s_{t+1};\theta). \tag{2}$$

Here, Z_θ estimates the flow through the initial state s_0 and it also estimates the partition function, the sum of all trajectory flows: $Z_\theta = \sum_{\tau \in \mathcal{T}} F(\tau) = \sum_{x \in \mathcal{X}} R(x)$ when the constraint is satisfied, i.e., when the corresponding mismatch loss is minimized.

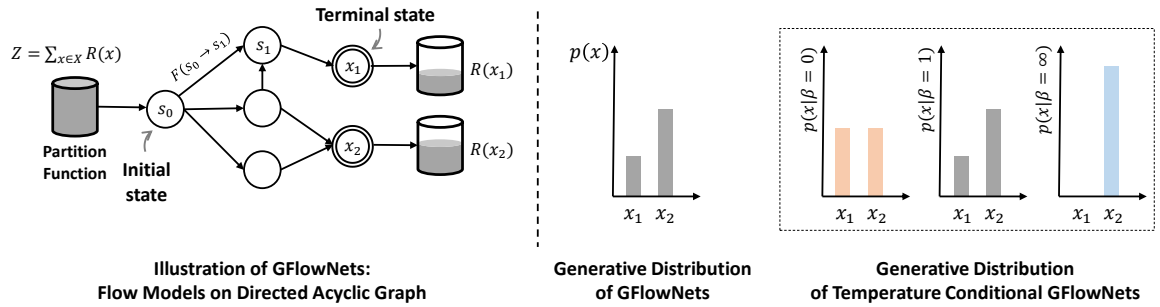

Figure 1. Illustration of GFlowNets and temperature conditional GFlowNets.

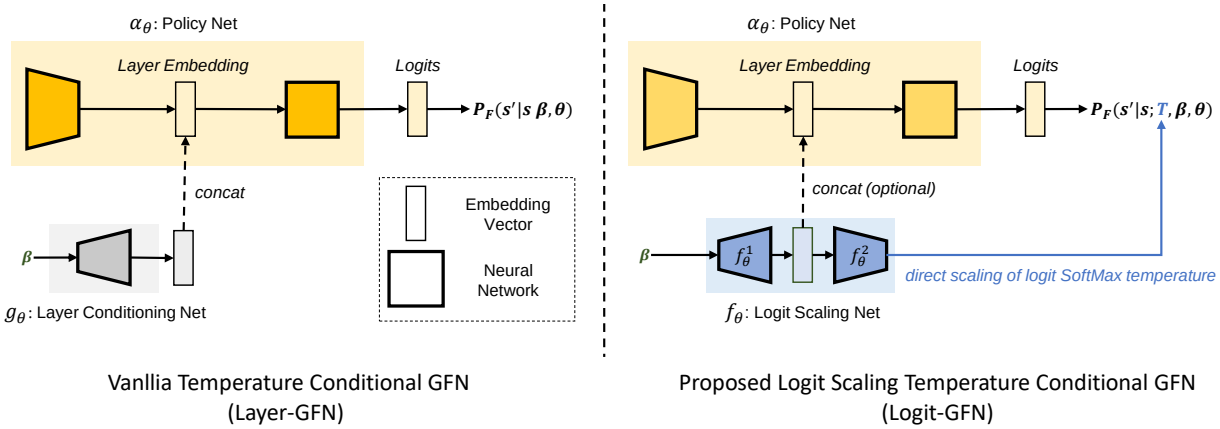

Figure 2. Architecture design of vanilla temperature conditional GFN and our Logit-GFN. The vanilla implementation of the temperature-conditional GFN integrates the embedding vector from β by concatenating it with the layer embedding of the policy network. In contrast, the proposed method directly modulates the logit Softmax temperature.

Other loss functions have been proposed to satisfy Equation (1), such as sub-trajectory balance (Madan et al., 2023), guided-trajectory balance (Shen et al., 2023) and detailed balance (Bengio et al., 2023).

3.1. Temperature conditional GFlowNets

The goal of the temperature-conditional GFlowNets is to train conditional generative models proportional to a tempered reward function, $p(x|\beta) \propto R(x)^\beta$; see Figure 1 for illustration. The models have to take the (inverse) temperature β as an additional input to represent a temperature-conditional distribution. A conventional approach for constructing a conditional model involves concatenating the conditioning values directly into model layers (Song et al., 2021; Ho et al., 2020; Zhang et al., 2022c). We denote this approach as *layer-conditioned* GFlowNet (Layer-GFN) that integrates temperature embeddings directly into the model parameterized by θ. The temperature embedding is a fixed or learned function $g_\theta : \mathbb{R} \to \mathbb{R}^d$, where d denotes the dimension of the temperature embedding.

Layer-GFN concatenates the output of the temperature embedding function, $g_\theta(\cdot)$ with the output of the state-input embedding module in order to produce logits for the forward and backward policies (and flows if desired). For example, the forward policy is parameterized as follows:

$$P_F^{\text{Layer}}(s'|s;\beta,\theta) := \frac{\exp(\alpha_\theta(s, s', g_\theta(\beta)))}{\sum_{s'' \in Ch(s)} \exp(\alpha_\theta(s, s'', g_\theta(\beta)))} \tag{3}$$

Limitations While the Layer-GFN is a useful method for constructing temperature-conditional GFlowNets, our experiments also suggest that training temperature-conditional GFlowNets is numerically more difficult than training unconditional GFlowNets for each temperature. This motivated the variants of logit scaling proposed below.

4. Methodology

We propose Logit-GFN, a novel temperature-conditional GFlowNet that addresses the numerical challenges identified in previous versions of temperature-conditional GFlowNets. Specifically we introduce *logit-scaling* trick for Logit-GFN that directly adjusts the output softmax temperature (T) based on the specified input inverse temperature β. This provides a direct pathway for controlling the softmax temperatures of the forward policy, creating an effective inductive bias for adapting to changes in the target distribution's

temperature. This, in turn, ensures more stable training. We verify the effectiveness of the proposed architecture in training stability and offline generalization capability. Also, we provide a novel online learning algorithm by leveraging the Logit-GFNs, which can effectively discover novel combinatorial structures (i.e., modes) in scientific discovery tasks. See the right side of Figure 2 for the overall architecture of Logit-GFN.

4.1. Logit scaling

The objective of the method aligns with that of layer-conditioning: to facilitate the training of temperature-conditional GFlowNets $p(x|\beta) \propto R(x)^{\beta}$ over varying inverse temperatures β. Logit-scaling trick use a simple skip connection to adjust the softmax temperature T of logits of P_F as a direct function of β. More specifically, logit-scaling into policy net α_θ can be defined as follows:

$$P_F(s'|s;\beta,\theta) := \frac{\exp\left(\alpha_\theta\left(s,s'\right)/f_\theta\left(\beta\right)\right)}{\sum_{s''\in Ch(s)}\exp(\alpha_\theta(s,s'')/f_\theta\left(\beta\right))}, \tag{4}$$

where $\alpha_\theta : S \times S \to \mathbb{R}$ now becomes neural net that is independent of β and $f_\theta : \mathbb{R} \to \mathbb{R}$ is the *logit scaling* net, which transforms the inverse temperature β into a softmax temperature, $T = f_\theta\left(\beta\right)$. Note that the logit-scaling method is agnostic to the policy network; the policy network can take any form, including a layer-conditioning network.

For a detailed parameterization of logit-scaling, the logit-scaling network, f_θ, consists of an encoder and a decoder. The encoder, $f_\theta^1 : \mathbb{R} \to \mathbb{R}^D$, maps a scalar input to a D-dimensional embedding vector. Conversely, the decoder, $f_\theta^2 : \mathbb{R}^D \to \mathbb{R}$, converts this embedding vector back into a scalar. The overall transformation is represented by the composition $f = f_\theta^1 \circ f_\theta^2$, seamlessly integrating the encoder and decoder functionalities. Given the logit scaling net, the softmax temperature $T = f_\theta\left(\beta\right)$ determines the confidence level of P_F as a lower T makes a sharper decision, and a higher T gives a smoother decision. By adjusting the scalar value T as a function of β, we can adjust the output generative distribution easily without heavy parameterization. The training objective makes the logit scaling net, f_θ, adjust the target temperature-conditional forward policy towards matching the tempered reward function.

Layer-conditioning with logit scaling. Utilizing the logit scaling network, f_θ, the Logit-GFN provides the flexibility to incorporate layer-conditioning using a simple yet intuitive technique. We start by taking the output of the encoder, $f_\theta^1\left(\beta\right)$, which serves as a latent temperature embedding vector. This vector can then be seamlessly concatenated with the layer embedding of the policy net, α_θ, as depicted in Figure 2. Since the expressive power of *logit-scaling* is limited when α_θ is used unconditionally with respect to β, integrating it with the *layer-conditioning* method can be a viable option to achieve full expressive power for temperature conditioning. We explore it further in the Appendix C.

4.2. Training objective

The training procedure for our experiments is based on the trajectory balance (TB) loss, so we aim to minimize the TB loss given a training replay buffer or dataset $\mathcal{D}$ similar to prior work (Shen et al., 2023). The difference is that we train the GFlowNets with multiple values of $\beta \sim P_{\text{train}}(\beta)$:

$$\mathcal{L}(\theta;\mathcal{D}) = \mathbb{E}_{P_{\text{train}}(\beta)}\mathbb{E}_{P_{\mathcal{D}}(\tau)}\left[\left(\log\frac{Z_\theta(\beta)\prod_{t=1}^{n}P_F(\cdot)}{R(x)^{\beta}\prod_{t=1}^{n}P_B(\cdot)}\right)^2\right]. \tag{5}$$

$$\text{where}\quad P_F(\cdot) = P_F(s_t|s_{t-1};\beta,\theta)$$
$$P_B(\cdot) = P_B(s_{t-1}|s_t;\beta,\theta)$$

When considering the parameterization of deep neural networks (DNNs) using θ, a key implementation feature lies in the conditioning of the partition function Z on the inverse temperature β, i.e., we write $Z_\theta(\beta)$ as a learned function rather than a learned constant. Additionally, the conditional dependencies of P_F and P_B on β are established by the DNN f_θ. This architecture necessitates the incorporation of two auxiliary DNNs, namely Z_θ and f_θ; however, it's worth noting that these mappings operate from a scalar to a scalar, mitigating the need for an excessive number of parameters. This training objective and the mathematical theory for conditional GFlowNets was originally proposed by (Bengio et al., 2023).

4.3. Online discovery algorithm with Logit-GFN

In scientific discovery (e.g., molecule optimization), we aim to discover a set of diverse candidate objects $x \in \mathcal{X}$ with high reward $R(x)$, e.g., molecules with high binding affinity to some protein. What we care about is not only the top rewards among these candidates but also their diversity and the number of modes (a local peak in which the reward is above a certain threshold) (Jain et al., 2022), due to the uncertainty and imperfections of the reward functions.

Typically, training of GFlowNets involves an exploratory policy that generates trajectories that can be put in a prioritized replay buffer and are then used to perform gradient updates on the GFlowNets parameters. We can significantly enhance the exploratory phase of GFlowNets by querying multiple values of β with temperature-conditional GFlowNets when forming samples for the replay buffer. This enables the model to generate a diverse set of candi-

dates sampled from various generative distributions:

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_1, \ldots, \tau_M\} \tag{6}$$

$$\tau_1, \ldots, \tau_M \sim \int_{\beta} P_F(\tau|\beta) dP_{\text{exp}}(\beta). \tag{7}$$

The dynamic control policy $P_{\text{exp}}(\beta)$ controls the range of β for exploration, and it may be chosen differently from $P_{\text{train}}(\beta)$ in Eq. 5. We provide several empirical observations for different P_{exp} in Section 5.4 and Appendix D.2. See Algorithm 1 for full pseudocode.

5. Experiments

We present experimental results on 4 biochemical tasks: QM9, sEH, TFBind8, and RNA-binding. Following recent work (Shen et al., 2023), we formulate the problem as a sequence prepend/append MDP, where the actions add a token either to the leftmost or rightmost end of a partial sequence or molecule. This setting results in multiple trajectories τ for each sample x, where the DAG structure is crucial for exploring the trajectory space. We describe the tasks below.

QM9 This task requires generating a small molecule graph. We build a graph using 12 building blocks with 2 stems, and each molecule consists of 5 blocks. Our objective is to maximize the HOMO-LUMO gap (Zhang et al., 2020).

sEH This task requires generating a small molecule graph. We build a graph using 18 building blocks with 2 stems, and each molecule consists of 6 blocks. Our objective is to maximize binding affinity (Bengio et al., 2021).

TFBind8 This task requires generating a string of 8 nucleotides. The objective is to maximize the DNA binding affinity to a human transcription factor, SIX6, from (Trabucco et al., 2022).

RNA-Binding This task requires generating a string of 14 nucleobases. The objective is to maximize the binding affinity to the target transcription factor. We use L14-RNA1 as the target factor, introduced by Sinai et al. (2020).

5.1. Evaluation of training stability

We first assess the training stability of temperature-conditional GFlowNets.

To evaluate stability, we compute the TB loss using samples generated by the forward policy under extremely high β conditions ($\beta = 5,000$). During off-policy training (with exploration), we sample multiple β values from a range of relatively low temperatures, ($\beta \sim U^{[10,50]}$). For unconditional GFlowNets, we implement an unconditional policy that is independent of the temperature and put the original reward to the power of 5,000. For additional details on the experimental setup, please see Appendix A.3.1.

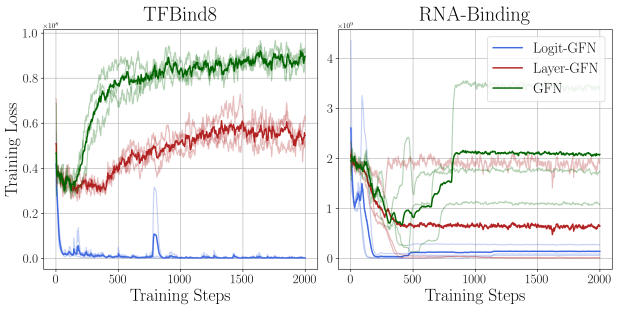

Figure 3. Loss of Temperature-conditional GFlowNets and unconditional GFlowNets as a function of a number of training steps on the TFBind8 and RNA-Binding tasks. Logit-GFN yields more stable training curves and converges faster. We draw curves with three different random seeds and highlight the mean over seeds.

Figure 3 illustrates the loss curves for both temperature-conditional and unconditional GFlowNets in TFbind8 and RNA-binding tasks. It is evident that training unconditional GFlowNets with high β values leads to extreme instability. Although Layer-GFN alleviates this issue to some extent, it still exhibits considerable instability. Our novel approach, Logit-GFN, stands out by maintaining stable loss levels even when operating with high, previously unseen inverse temperatures. This underscores the training stability of our proposed logit scaling in the development of temperature-conditional GFlowNets.

5.2. Evaluation of offline generalization

In this section, we examine the controllability of temperature-conditional GFlowNets, specifically the ability to induce a relationship: $p(x|\beta) \propto R(x)^{\beta}$. To verify this, we prepare an offline dataset $\mathcal{D} = \{(x_i, R(x_i))\}_{i=1}^{N}$, similar to offline model-based optimization (Trabucco et al., 2022).

We train temperature-conditional GFlowNets with relatively low values of β and generate samples by querying high β to verify that we can find high-scoring samples that surpass the offline dataset via out-of-distribution generalization. We also train unconditional GFlowNets trained with a high, fixed β independently for comparison. To evaluate performance, we use 25^{th} percentile and median reward of samples generated from trained models, which can be used as an indicator of how trained policies adaptively respond to varying temperatures. Note that we train a single temperature-conditional model and investigate its generalization performance by querying with different β values. For details on the experiment setting, please refer to Appendix A.3.2.

Figure 4 shows the offline generalization performance of GFlowNets in three biochemical tasks: QM9, TFBind8, and RNA-binding. The shaded region indicates the training range for temperature-conditional GFlowNets. We refer to it as in-distribution and consider querying outside of

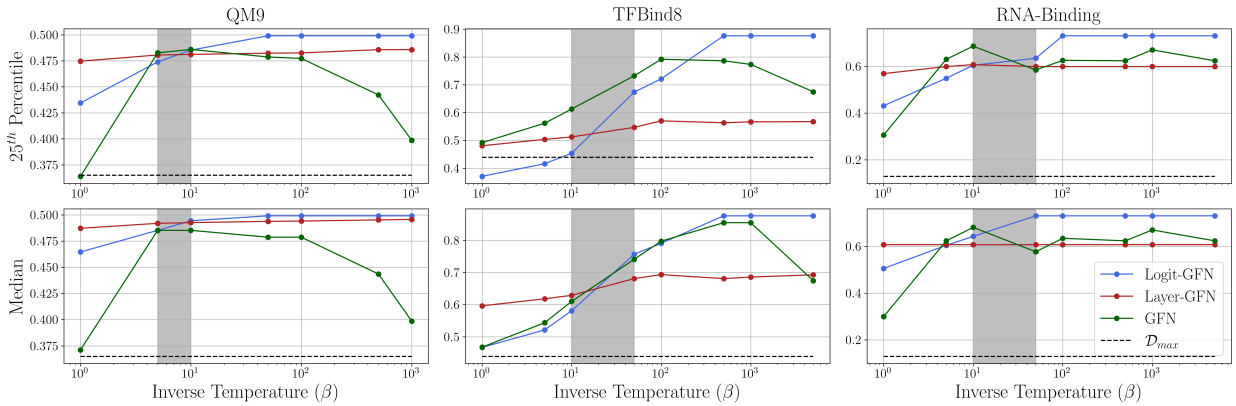

Figure 4. Performance of Temperature-conditional GFlowNets and unconditional GFlowNet in offline generalization. Shaded regions denote the temperature range used in training. Logit-GFN generates high-rewarding samples that surpass the offline datasets when conditioned on high β values.

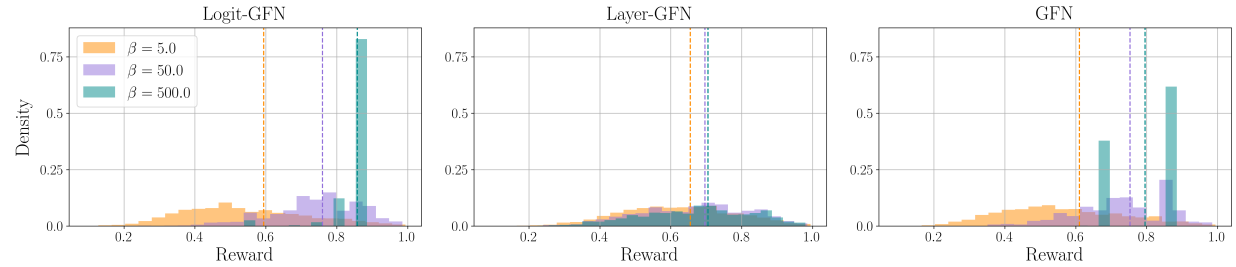

Figure 5. Reward distribution of samples from Temperature-conditional GFlowNets and unconditional GFlowNet in offline generalization. Logit-GFN dynamically shifts its reward distribution towards a high-reward region when conditioned on high β values.

the shaded region as an out-of-distribution generalization. The figure demonstrates that Logit-GFN exhibits powerful generalization performance in high β and is able to generate high-rewarding samples that surpass the offline dataset, corresponding to the queried values. However, training unconditional GFlowNets with high β results in degraded performance despite their specialization to the fixed high β. While Layer-GFN often shows promising results on high β, it seems that Layer-GFN does not adaptively respond to different temperatures and tends to return relatively high-reward samples.

Figure 5 demonstrates Logit-GFN's ability in accurately capturing the reward distribution of samples across various temperatures. While Layer-GFN is unable to effectively shift its reward distribution according to β, the figure distinctly highlights that Logit-GFN dynamically adapts to different β values. Unconditional GFlowNets individually trained with their respective β values converge to generate sub-optimal samples, necessitating temperature-conditional GFlowNets, which can generalize across temperatures.

Mapping from β to T. Figure 6 depicts the learned mapping from β to T in QM9 and TFBind8 tasks. We note that for both tasks, T tends towards zero with the increase in β. It is a reasonable behavior since lower T makes a narrow decision-making and accentuates the minor differences between probabilities.

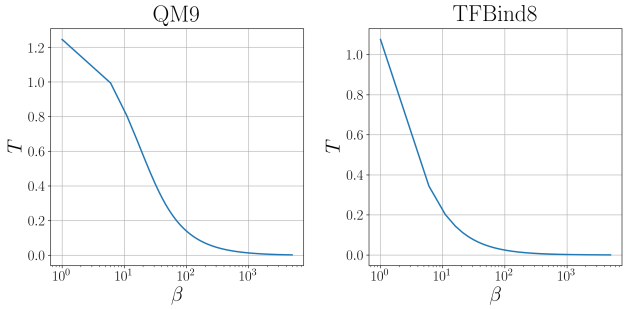

Figure 6. Learned relationship between β and T captured by Logit-GFN. Logit-GFN suggests low softmax temperature when the policy is conditioned on high temperature (low β).

5.3. Evaluation of online mode seeking capability

The preceding sections have demonstrated that Logit-GFN provides enhanced training stability and better controllability compared to unconditional GFlowNets and Layer-GFN. In this section, we aim to validate the usefulness of Logit-GFN in solving online mode-seeking problems.

The online mode-seeking problem is an important problem

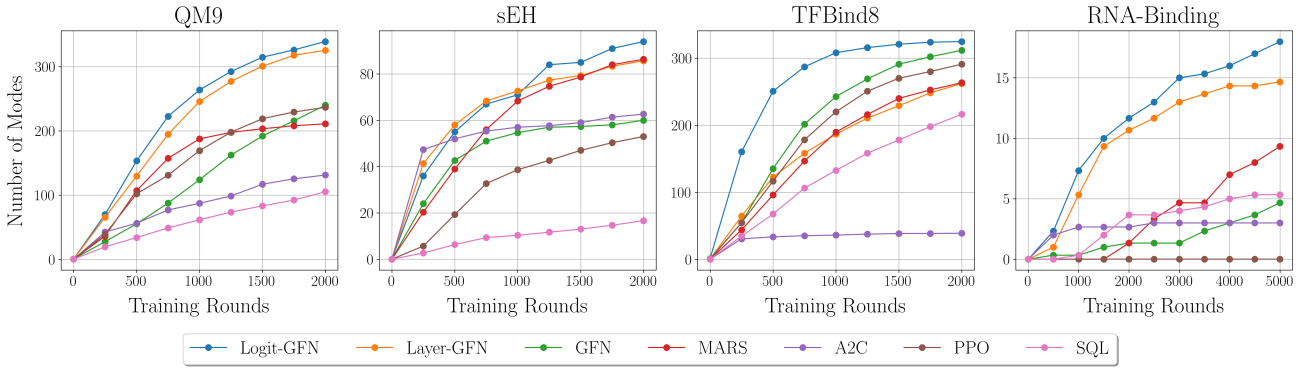

Figure 7. Number of modes discovered over training. Logit-GFN outperforms established baselines including Layer-GFN for all tasks.

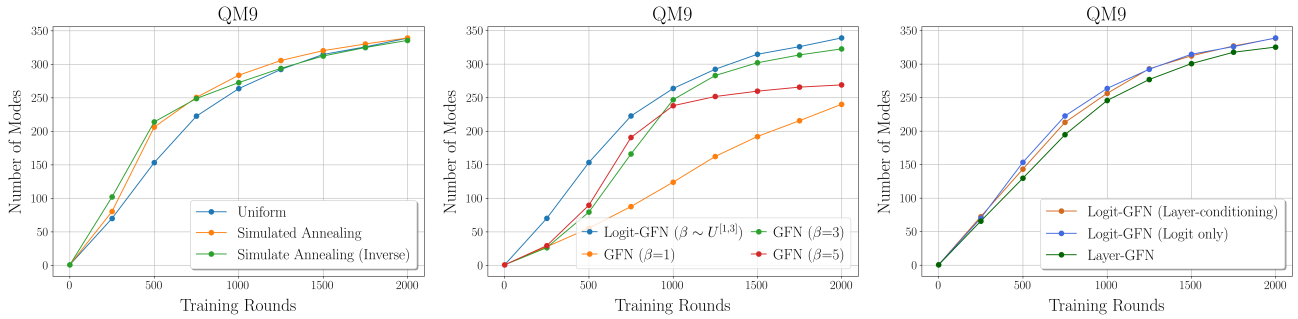

Figure 8. **Left**: Different distributions for sampling temperatures. The simulated annealing strategy is more effective than uniform sampling in online mode-seeking problems. **Middle**: Ablation on different β for unconditional GFlowNets. Logit-GFN outperforms all GFlowNets trained on specific β. **Right**: Ablation on Layer-conditioning for Logit Scaling. Both outperforms Layer-GFN.

in biochemical tasks as we require generating both diverse and high-reward samples due to the lack of robustness and misspecification of the estimated reward function (Bengio et al., 2021). The use of temperature-conditional GFlowNet is particularly beneficial in this setting as we can more easily balance exploration and exploitation by querying different β values, which is very useful for discovering a greater diversity of modes of the reward function.

Baselines. To evaluate the proposed methods, we first compare them with the established unconditional GFlowNet. We also consider MARS (Xie et al., 2020), which is an MCMC-based method known to work well in this molecule sampling domain, and RL-based methods which include A2C with Entropy Regularization (Mnih et al., 2016), Soft Q-Learning (Haarnoja et al., 2018), and PPO (Schulman et al., 2017) as baselines. All experiments are conducted with three different random seeds to evaluate reliability.

Distribution for sampling temperature. Online mode seeking problems involve two stages of sampling temperatures, $P_{\text{train}}(\beta)$ and $P_{\text{exp}}(\beta)$. $P_{\text{train}}(\beta)$ is used for training temperature-conditional policy via off-policy manner, and $P_{\text{exp}}(\beta)$ is for sampling objects using trained policy in the online round. For both stages, we employ a uniform distribution as a default setting. We discuss various types of distribution, such as simulated annealing, in the next section.

Figure 7 showcases the number of modes discovered during the training of our proposed approach compared to established baselines in four distinct biochemical tasks. Our method not only outperforms both the unconditional GFN and other reward maximization techniques in the number of modes uncovered but also exhibits remarkable efficiency in detecting these modes with a limited number of online evaluation rounds. This underscores the effective capability of temperature-conditional GFlowNets to strike a balance between exploitation and exploration strategies.

5.4. Different distribution for sampling temperatures

We explore diverse distributions for sampling temperatures, especially during the exploration phase, namely $P_{\text{exp}}(\beta)$. One of the most widely used heuristics in combinatorial optimization is *simulated annealing*, which gradually shifts the distribution of β towards a high region (Kirkpatrick et al., 1983). Accordingly, we implement both simulated annealing and its inverse (shift distribution from high to low β region). The effects of these exploration strategies in the QM9 task are illustrated in Figure 8 (Left). The simulated annealing strategy demonstrates high efficiency in discovering modes compared to the uniform strategy. While the inverse simulated annealing strategy exhibits fast convergence at the early stage of training rounds, but encounters

a significant decrease in efficiency as the rounds advance. It indicates that carefully adjusting the balance between exploration and exploitation can effectively enhance the performance of temperature-conditional GFlowNets for online mode-seeking problems. We conduct more extensive experiments on $P_{\text{exp}}(\beta)$ in Appendix D.2.

5.5. Ablation studies

We can adjust the β of unconditional GFlowNets and train them independently to find the most effective configuration for a given task. Figure 8 (Middle) shows the number of modes discovered by GFlowNets trained with various β values. We observe that all methods trained with fixed β are outperformed by Logit-GFN, which trains conditional GFlowNets and balances exploitation and exploration with different β values.

We validate the effectiveness of layer-conditioning for Logit-GFN in online mode-seeking problems. As depicted in Figure 8 (Right), it seems layer-conditioning has a marginal effect on the performance of Logit-GFN, while still outperforming Layer-GFN. We discuss layer-conditioning in Appendix C in more detail.

We perform additional experiments on several design choices of temperature-conditional GFlowNets, such as thermometer embedding for Layer-GFN, the number of gradient steps per batch (K), and different GFlowNet training methods such as DB and SubTB in Appendices B.1, B.3 and B.4.

6. Conclusion

We introduced the Logit-GFN, a novel architecture design that improves the training process of temperature-conditional GFlowNets. This approach addresses the numerical challenges of prior temperature-conditional GFlowNets by adopting the logit scaling net to scale the logits of policy directly. Empirical evaluations confirm that Logit-GFN can stabilize the training of temperature-conditional GFlowNets and exhibit strong generalization performance and effectiveness in diverse biochemical tasks. Additionally, while it demonstrates strong offline generalization and training stability in the tested biochemical tasks, further validation across a broader range of application domains is necessary to confirm its robustness and adaptability. A potential limitation of the Logit-GFN is that its effectiveness in very high-dimensional or highly complex generative tasks has not been fully explored.

Impact Statement

Our research is a part of the ongoing developments in the field of generative modeling and encompasses various societal implications typical of such advancements. However, we believe there are no specific consequences that require particular emphasis in this work.

Acknowledgement

The authors acknowledge the financial support provided by CIFAR, NSERC, Samsung, FACS Acuité, and NRC AI4Discovery. This research was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (2022-0-01032, Development of Collective Collaboration Intelligence Framework for Internet of Autonomous Things). This research was supported in part by the computational resources supplied by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), NVIDIA. Additionally, J. Ko and W. Kim received support from the National Research Foundation of Korea (NRF) grants funded by the Ministry of Science and ICT (NRF-2022M3J6A1063021 and RS-2023-00208980).

References

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

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023. URL http://jmlr.org/papers/v24/22-0364.html.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW.

Buesing, L., Heess, N., and Weber, T. Approximate inference in discrete distributions with monte carlo tree search and value functions. In *International Conference on Artificial Intelligence and Statistics*, pp. 624–634. PMLR, 2020.

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

Deleu, T., Nishikawa-Toomey, M., Subramanian, J., Malkin, N., Charlin, L., and Bengio, Y. Joint bayesian inference of graphical structure and parameters with a single generative flow network. *Advances in Neural Information Processing Systems*, 36, 2023.

Falet, J.-P. R., Lee, H. B., Malkin, N., Sun, C., Secrieru, D., Zhang, D., Lajoie, G., and Bengio, Y. Delta-AI: Local objectives for amortized inference in sparse graphical models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LemSSn8htt.

Gagrani, M., Rainone, C., Yang, Y., Teague, H., Jeon, W., Bondesan, R., van Hoof, H., Lott, C., Zeng, W., and Zappi, P. Neural topological ordering for computation graphs. *Advances in Neural Information Processing Systems*, 35:17327–17339, 2022.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Hu, E. J., Jain, M., Elmoznino, E., Kaddar, Y., Lajoie, G., Bengio, Y., and Malkin, N. Amortizing intractable inference in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ouj6p4ca60.

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

Jain, M., Deleu, T., Hartford, J., Liu, C.-H., Hernandez-Garcia, A., and Bengio, Y. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023a.

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

Jang, H., Kim, M., and Ahn, S. Learning energy decompositions for partial inference in GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=P15CHILQlg.

Kim, M., Berto, F., Ahn, S., and Park, J. Bootstrapped training of score-conditioned generator for offline design of biological sequences. *arXiv preprint arXiv:2306.03111*, 2023.

Kim, M., Yun, T., Bengio, E., Zhang, D., Bengio, Y., Ahn, S., and Park, J. Local search GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6cFcw1Rxww.

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

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pp. 18269–18300. PMLR, 2023.

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

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

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E. J., Everett, K. E., Zhang, D., and Bengio, Y. Gflownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2022b.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937. PMLR, 2016.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems*, 29, 2016.

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

Pan, L., Zhang, D., Courville, A., Huang, L., and Bengio, Y. Generative augmented flow networks. In *The Eleventh International Conference on Learning Representations*,

2023b. URL https://openreview.net/forum?id=urF_CBK5XC0.

Pan, L., Zhang, D., Jain, M., Huang, L., and Bengio, Y. Stochastic generative flow networks. In *Conference on Uncertainty in Artificial Intelligence*, 2023c.

Qiu, Z.-H., Hu, Q., Yuan, Z., Zhou, D., Zhang, L., and Yang, T. Not all semantics are created equal: Contrastive self-supervised learning with automatic temperature individualization. *arXiv preprint arXiv:2305.11965*, 2023.

Qiu, Z.-H., Guo, S., Xu, M., Zhao, T., Zhang, L., and Yang, T. To cool or not to cool? temperature network meets large foundation models via dro. *arXiv preprint arXiv:2404.04575*, 2024.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. Towards understanding and improving gflownet training. In *International Conference on Machine Learning*, pp. 30956–30975. PMLR, 2023.

Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., and Kelsic, E. D. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Tiapkin, D., Morozov, N., Naumov, A., and Vetrov, D. P. Generative flow networks as entropy-regularized rl. In *International Conference on Artificial Intelligence and Statistics*, pp. 4213–4221. PMLR, 2024.

Trabucco, B., Geng, X., Kumar, A., and Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pp. 21658–21676. PMLR, 2022.

Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., and Li, L. Mars: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2020.

Zhang, D., Chen, R. T. Q., Malkin, N., and Bengio, Y. Unifying generative models with gflownets. *ArXiv*, abs/2209.02606, 2022a.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pp. 26412–26428. PMLR, 2022b.

Zhang, D., Dai, H., Malkin, N., Courville, A., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial optimization problems with gflownets. *Advances in Neural Information Processing Systems*, 36, 2023.

Zhang, D., Chen, R. T. Q., Liu, C.-H., Courville, A., and Bengio, Y. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=OIsahq1UYC.

Zhang, D., Pan, L., Chen, R. T. Q., Courville, A., and Bengio, Y. Distributional GFlownets with quantile flows. *Transactions on Machine Learning Research*, 2024b. ISSN 2835-8856. URL https://openreview.net/forum?id=vFSsRYGpjW. Expert Certification.

Zhang, D. W., Rainone, C., Peschl, M., and Bondesan, R. Robust scheduling with gflownets. In *The Eleventh International Conference on Learning Representations*, 2022c.

Zhang, S., Liu, Y., and Xie, L. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures. *arXiv preprint arXiv:2011.07457*, 2020.

Zhou, M., Yan, Z., Layne, E., Malkin, N., Zhang, D., Jain, M., Blanchette, M., and Bengio, Y. Phylogfn: Phylogenetic inference with generative flow networks. *arXiv preprint arXiv:2310.08774*, 2023.

A. Detailed experimental setting

A.1. Implementation

In our GFlowNet implementations, we adhere closely to the methodologies outlined in (Shen et al., 2023). We take the approach of re-implementing only those methods that do not already exist in the literature.

All of our GFlowNet models incorporate the parametrization mapping of relative edge flow policy (SSR), as originally proposed by (Shen et al., 2023).

When dealing with pairs of states (s, s'), we encode each state into a one-hot encoding vector and concatenate them as input for the forward and backward policy networks. For QM9 and sEH tasks, we employ a two-layer architecture with 1024 hidden units, while for the other tasks, we choose to use a two-layer architecture with 128 hidden units. Both forward and backward policies use the same architecture but with different parameters. We initialize $\log Z_\theta$ to 5.0 for the baseline implementation.

For temperature-conditional GFlowNets, we introduce a two-layer MLP with a 32-dimensional hidden layer and a Leaky ReLU activation function for embedding inverse temperature β. For Layer-GFN, we use the output of the MLP as an embedding and concatenate with state embedding before passing forward policy networks. For Logit-GFN, we introduce an additional two-layer MLP, which has 32 hidden units and returns scale value, which represents T. To ensure that T should be greater than zero, we use the Softplus activation function for the last layer. For parameterizing $Z_\theta(\beta)$, we use a two-layer MLP with a 32-dimensional hidden layer and Leaky ReLU.

A.2. Hyperparameters

Regarding the hyperparameters for GFlowNets, we also follow the initial settings proposed by (Shen et al., 2023) without alteration. Across all tasks, we employ the Adam optimizer (Kingma & Ba, 2014) with the following learning rates: 1×10^{-2} for Z_θ and 1×10^{-4} for both the forward and backward policy. Furthermore, we apply distinct reward exponents (e) and reward normalization constants for each task, following the guidelines suggested by (Shen et al., 2023). As a result, we power the reward $e \times \beta$ times for temperature-conditional GFlowNets conditioned on β. Table 1 summarizes the reward exponent and normalization constants for different task settings.

In the implementation of RL baselines, we utilize the same MLP architecture as employed in the GFlowNet baselines. The optimization of hyperparameters is achieved through a grid search approach on the QM9 task, with a focus on determining the optimal number of modes. For the A2C algorithm with entropy regularization, we segregate parameters for the actor and critic networks. The selected learning rate of 1×10^{-4} is chosen from a range of options, including $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-3}, 1 \times 10^{-3}\}$, and we incorporate an entropy regularization coefficient of 1×10^{-2} selected from $\{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$. In the case of Soft Q-Learning, we opt for a learning rate of 1×10^{-4} selected from the same set of values: $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-3}, 1 \times 10^{-3}\}$. For the PPO algorithm, we introduce an entropy regularization term of 1×10^{-2} and employ a learning rate of 1×10^{-4}, similarly chosen from $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-3}, 1 \times 10^{-3}\}$. The entropy regularization coefficient is selected from $\{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$.

Table 1. GFlowNet hyperparameters for various tasks

Tasks	Reward Exponent	Reward Normalization Constant
QM9	5	100.0
sEH	6	10.0
TFBind8	3	10.0
RNA-binding	8	10.0

A.3. Experiment details

A.3.1. Training stability

To evaluate the training stability, we compute the loss on samples generated from the forward policy conditioned on $\beta = 5,000$. For each active round, we generate 32 samples for evaluating loss. After sampling, we train GFlowNets in an off-policy manner. We utilize the reward prioritized replay buffer (PRT) suggested in (Shen et al., 2023) to obtain samples for computing the loss of off-policy training for all models. We perform 1 gradient step per active round and use 32 samples from PRT to compute loss. During off-policy training, both Logit-GFN and Layer-GFN samples β from the uniform distribution, i.e., $\beta \sim U^{[10,50]}$.

A.3.2. Offline generalization

For offline generalization, we train temperature-conditional GFlowNets in a relatively small β during training and generate samples by querying with various β from low to high values to see their generalization performance. For the QM9 task, we train temperature-conditional GFlowNets with $\beta \sim U^{[5,10]}$ and query with $\beta = \{1, 5, 10, 50, 100, 500, 1000\}$. For TFBind8 and RNA-binding tasks, we train temperature-conditional GFlowNets with $\beta \sim U^{[10,50]}$ and query with $\beta = \{1, 5, 10, 50, 100, 500, 1000, 5000\}$. As querying $\beta = 5,000$ for the QM9 task leads to numerical instability due to the reward scaled to 100, we exclude the process from this experiment. Unconditional GFlowNets are trained with a fixed β independently for each β. All methods are run for 1,000 training steps and we generate 2,048 samples from the trained policy to measure the performance.

A.3.3. Online mode seeking

For online mode-seeking problems, we run experiments with $T = 2,000$ training rounds for QM9, sEH, and TFBind8 and $T = 5,000$ training rounds for RNA-Binding tasks. As sEH and RNA-binding tasks have a large combinatorial space to explore, we apply PRT for all GFlowNets methods to enhance sample efficiency. For each training round, we collect 32 online samples. During the off-policy training phase, we sample 32 samples from the replay buffer and update the policy via gradient descent K times, where K holds a significant role in the training of GFlowNets. A low number of gradient updates before sampling new trajectories can result in underfitting, while an excessive number of gradient updates may lead to overfitting. In contrast to the unconditional GFlowNets, we observed that temperature-conditional GFlowNets require more training iterations at each round to adapt to various temperature conditions effectively. We present ablations on K in Appendix B.3 with more details.

For online mode-seeking problems, the distribution of β for training and exploration is crucial. As a default setting, we choose a uniform distribution for both the training and exploration phases. We summarize the distribution we used for different biochemical tasks in Table 2.

Table 2. Temperature Distributions of Temperature-conditioned GFlowNets for various tasks

Tasks	Training Phase, $P_{\text{train}(\beta)}$	Exploration Phase, $P_{\text{exp}(\beta)}$	K
QM9	$U^{[1,3]}$	$U^{[1,3]}$	4
sEH	$U^{[1,5]}$	$U^{[1,5]}$	5
TFBind8	$U^{[1,3]}$	$U^{[1,3]}$	4
RNA-Binding	$U^{[2,3]}$	$U^{[2,3]}$	1

A.4. Offline dataset details

We verify the controllability of Logit-GFN with offline generalization. We prepare a sub-optimal offline dataset similar to offline model-based optimization for training GFlowNets. We describe the details of the dataset for each task.

- **QM9**: In QM9 task, we build an offline dataset $\mathcal{D}$ using under 50th percentile data, which consists of 29,382 samples.
- **TFBind8**: In TFBind8 task, we follow the method suggested in Design-bench (Trabucco et al., 2022). We build an offline dataset $\mathcal{D}$ using under 50th percentile data, which consists of 32,898 samples.
- **RNA-Binding**: In the RNA-binding task, we follow the method suggested in BootGen (Kim et al., 2023). We prepare an offline dataset consisting of 5,000 randomly generated RNA sequences.

A.5. Mode metrics details

We define the measure of the number of modes in a sampled dataset as the count of data points with a reward above a specified threshold, where each data point is dissimilar from the others. The threshold is determined by the reward of the top 0.5% of samples Additionally, we use the Tanimoto diversity metric for molecule optimization tasks to measure dissimilarity. We only accept samples as modes that are far away from previously accepted samples in terms of Tanimoto diversity. We set the threshold as 0.5. For RNA-binding tasks, we define mode as a local optimum among its 2-hamming ball neighborhoods. For the TFBind8 task, since there is already a well-pre-defined set of modes considering both optimality and diversity, we use that pre-defined set for evaluation.

A.6. Additional results in offline generalization

We also visualize the reward distribution obtained from QM9 and RNA-binding tasks in Figures 9 and 10, which show the reward distribution of samples when querying with different β values. For all tasks, it is observed that Logit-GFN focuses on high-scoring samples when querying with high β, and reacts more adaptively to changes in β compared to Layer-GFN. The performance of Unconditional GFN methods degrades when trained with excessively high β values, highlighting the need for temperature-conditional GFlowNets with enhanced generalization capabilities. Additionally, histograms are presented in Figure 11 to facilitate a direct comparison between different GFlowNets using the same β values.

Note that our objective is not to achieve state-of-the-art performance in offline MBO setting (Trabucco et al., 2022). Our aim is to verify that Logit-GFN can generate out-of-distribution high-scoring samples by querying with high β through generalization capability learned from training with various β values.

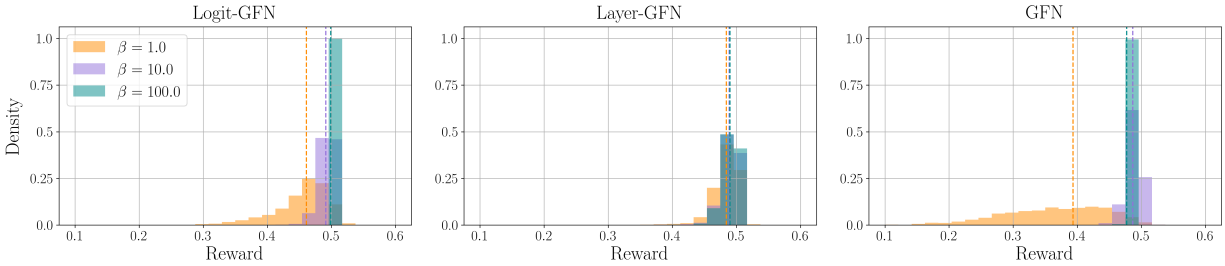

Figure 9. Reward distribution of samples generated from between Temperature-conditional GFlowNets and unconditional GFlowNet in offline generalization. Task is QM9.

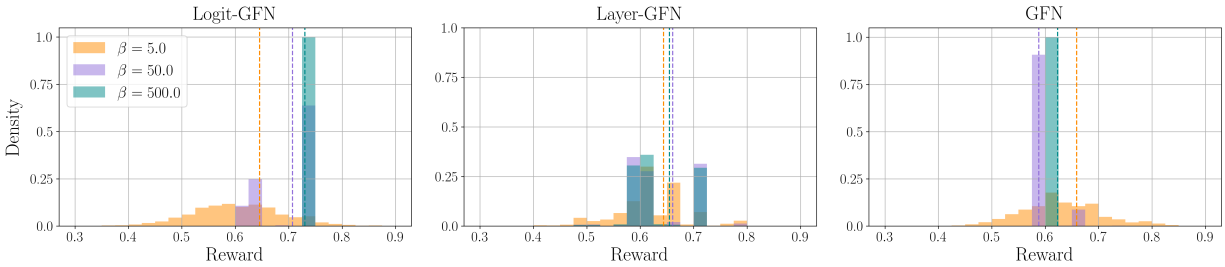

Figure 10. Reward distribution of samples generated from between Temperature-conditional GFlowNets and unconditional GFlowNet in offline generalization. Task is RNA-binding.

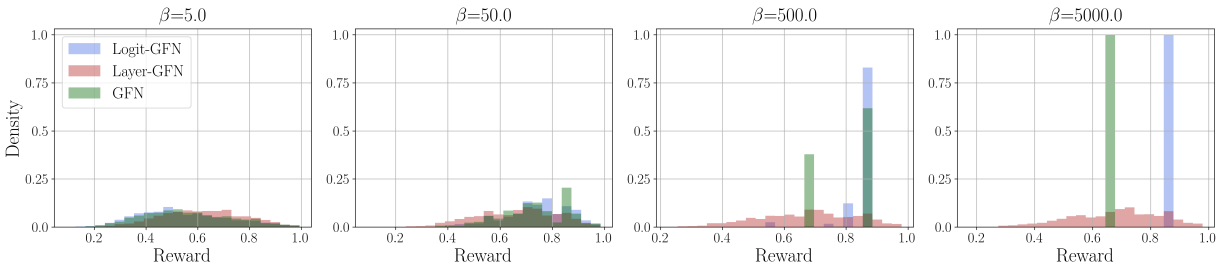

Figure 11. Reward distribution of samples generated from between temperature-conditional GFlowNets and unconditional GFlowNet in offline generalization. Task is TFBind8.

B. Additional Experiments

B.1. Study on thermometer encoding

It is often challenging to retrieve useful features from scalar input, which in our case is β. One way to overcome this is to utilize an additional encoding function in the very first of the embedding function. Among different encoding techniques, Buckman et al. (2018) introduced the thermometer encoding for such inputs to increase the robustness of neural networks, and Jain et al. (2023b) proved its effectiveness in preference-conditional GFlowNets.

Therefore, we ablate the effectiveness of thermometer encoding for Layer-GFN in online mode-seeking problems. Figure 12 shows the performance of Layer-GFN with and without thermometer encoding in various tasks. As shown in the figure, Layer-GFN with thermometer encoding outperforms naive Layer-GFN in TFBind8 but underperforms in QM9 and sEG. It seems applying thermometer encoding is not always useful, and its effectiveness varies across tasks. Therefore, we do not use thermometer encoding for Layer-GFN in the main experiments.

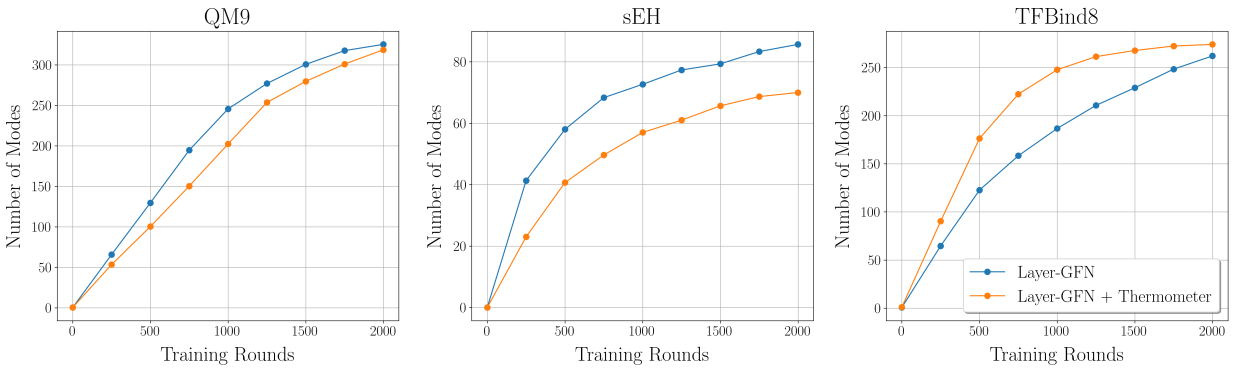

Figure 12. Ablation on Thermometer Encoding for Layer-GFN (Number of Modes)

We also analyze the loss curve across training Layer-GFN with and without thermometer encoding. As Figure 13 illustrates, thermometer encoding generally stabilizes the training of Layer-GFN while it is not always helpful in seeking modes.

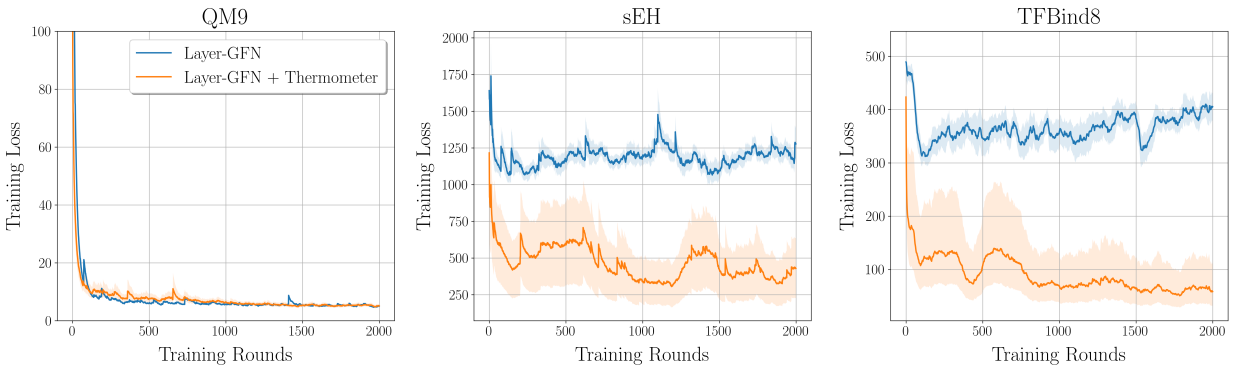

Figure 13. Ablation on Thermometer Encoding for Layer-GFN (Loss Curve)

B.2. Ablation on β of unconditional GFlowNets

In the main section, we ablate on β of unconditional GFlowNets in QM9. We also conduct similar experiments in other biochemical tasks. Figure 14 shows the performance of unconditional GFlowNets with different β in QM9 and TFBind8 tasks. As shown in the figure, Logit-GFN outperforms all the other unconditional GFlowNets tailored by a fixed β value, demonstrating the effectiveness of Logit-GFN in seeking online modes.

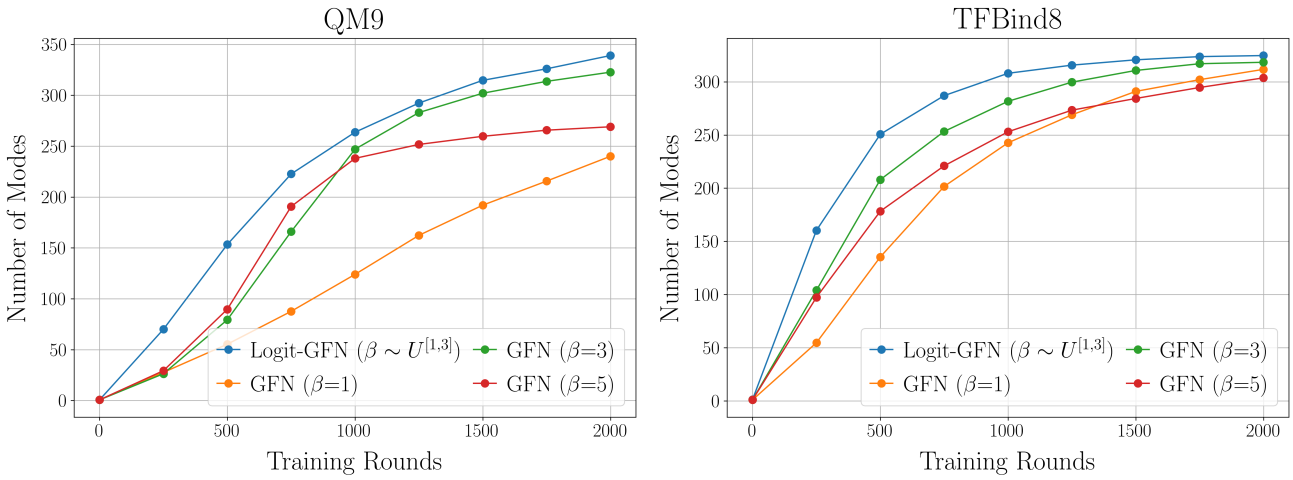

Figure 14. Ablation on different β for unconditional GFlowNets

B.3. Ablation on number of steps per batch (K)

The number of gradient steps per batch, K, holds a significant role in training GFlowNets. As a default setting, we choose $K = 1$ for unconditional GFlowNets as increasing K can lead to overfitting on current observations and converge to the local optimum. Figure 15 shows the performance of unconditional GFlowNets with increasing K. As illustrated in the figure, even if we increase K, Logit-GFN still outperforms unconditional GFlowNets by a large margin.

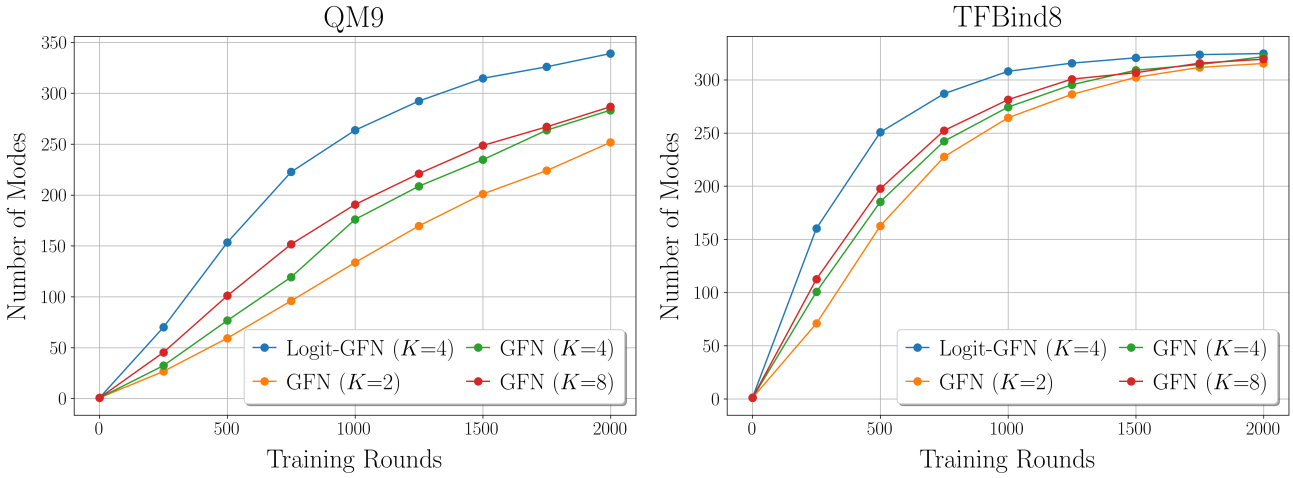

Figure 15. Ablation on different K for unconditional GFlowNets

B.4. Ablation on different GFlowNet training methods

Apart from TB, there are other loss functions for training GFlowNets that satisfy Equation (1), such as detailed balance (DB) and sub-trajectory balance (SubTB). In principle, our logit-scaling trick can accommodate any loss function. To verify this, we evaluate the effectiveness of our method with DB and SubTB training objectives as well. For SubTB, we use $\lambda = 0.9$. To implement DB and SubTB objectives, we need to predict state flow. We introduce a temperature-conditional state flow $F(s; \beta, \theta) : \mathcal{S} \times \mathbb{R} \to \mathbb{R}$ to parameterize state flow for Logit-GFN. We choose a two-layer MLP with a 32-dimensional hidden layer and a Leaky ReLU activation. The temperature-conditioned version of DB and SubTB can be written as:

$$\mathcal{L}_{\text{DB}}(\theta; \mathcal{D}) = \mathbb{E}_{P_{\text{train}}(\beta)} \mathbb{E}_{P_{\mathcal{D}}(\tau)} \left[\sum_{t=1}^{n-1} \left(\log \frac{F(s_{t-1}; \beta, \theta) P_F(s_t | s_{t-1}; \beta, \theta)}{F(s_t; \beta, \theta) P_B(s_{t-1} | s_t; \beta, \theta)} \right)^2 + \left(\log \frac{F(s_{n-1}; \beta, \theta) P_F(x | s_{n-1}; \beta, \theta)}{R(x)^\beta P_B(s_{n-1} | x; \beta, \theta)} \right)^2 \right]. \tag{8}$$

$$\mathcal{L}_{\text{SubTB}}(\theta; \mathcal{D}) = \mathbb{E}_{P_{\text{train}}(\beta)} \mathbb{E}_{P_{\mathcal{D}}(\tau)} \left[\frac{\sum_{0 \le i < j \le n} \lambda^{j-i} \mathcal{L}^{i,j}_{\text{SubTB}}(\theta; \tau)}{\sum_{0 \le i < j \le n} \lambda^{j-i}} \right]. \tag{9}$$

$$\mathcal{L}^{i,j}_{\text{SubTB}}(\theta; \tau) = \begin{cases} \left(\log \frac{F(s_i;\beta,\theta) \prod_{t=i+1}^{n} P_F(s_t | s_{t-1};\beta,\theta)}{R(x)^\beta \prod_{t=i+1}^{n} P_B(s_{t-1} | s_t;\beta,\theta)} \right)^2 & \text{if } j = n, \\ \left(\log \frac{F(s_i;\beta,\theta) \prod_{t=i+1}^{j} P_F(s_t | s_{t-1};\beta,\theta)}{F(s_j;\beta,\theta) \prod_{t=i+1}^{j} P_B(s_{t-1} | s_t;\beta,\theta)} \right)^2 & \text{otherwise.} \end{cases} \tag{10}$$

Figures 16 and 17 shows the number of discovered modes over training with different objectives in QM9 and TFBind8. Logit-GFN consistently outperforms unconditional GFN across various objectives, demonstrating its effectiveness.

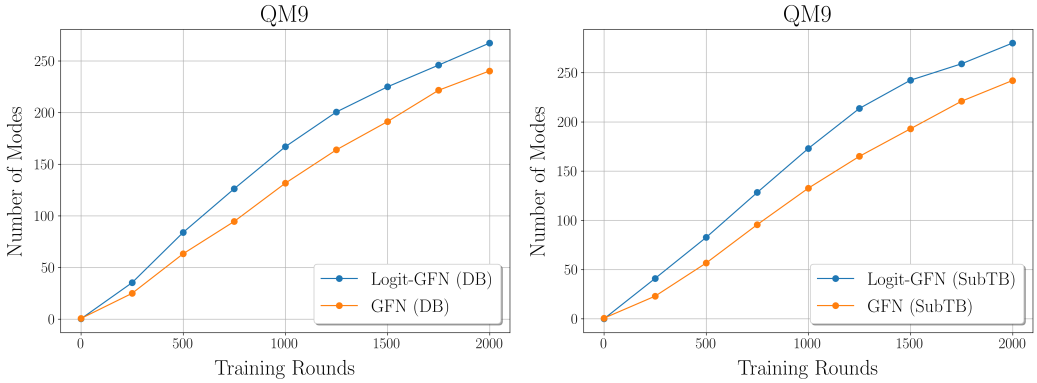

Figure 16. Ablation on different GFlowNet Training Methods in QM9 Task.

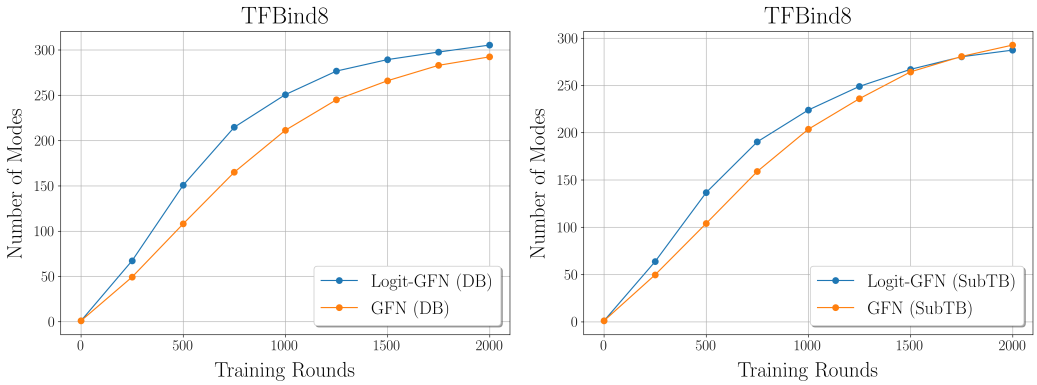

Figure 17. Ablation on different GFlowNet Training Methods in TFBind8 task

B.5. Experiments on non-biochemical tasks

As GFlowNets are not limited to biochemical domains, though they show significant promise in that area, our approach, Logit-GFN, is capable of performing well in other non-biochemical domains. One of the most critical areas for application is combinatorial optimization (CO). Specifically, we trained Logit-GFN in the graph combinatorial optimization domain, including tasks such as maximum independent sets (MIS) and maximum clique (MC) on top of detailed balance (DB) GFlowNet. We followed the implementation provided in Zhang et al. (2023) [1]. As shown in Table 3, Logit-GFN outperforms GFlowNet across various experiments within the combinatorial optimization domain, demonstrating its generalizability.

Table 3. Experiments in non-biochemical domains. We sample 20 solutions for each graph configuration and take the best result.

	MIS (Small)	MIS (Large)	MC (Small)	MC (Large)
Logit-GFN (DB)	**19.20 ± 0.07**	**35.65 ± 0.61**	**16.54 ± 0.20**	**34.08 ± 3.15**
GFN (DB)	17.93 ± 0.69	34.19 ± 0.37	15.56 ± 0.11	29.36 ± 0.01

[1] https://github.com/zdhNarsil/GFlowNet-CombOpt

C. Logit-GFN with Layer-conditioning

We investigate the impact of incorporating layer-conditioning into Logit-GFN. In our approach, the embedding produced by the encoder of the logit scaling network, denoted as f_θ^1, is concatenated with the state embedding. This combined embedding is then input into the forward policy networks. The process of concatenation is illustrated in Figure 2.

C.1. Experiments on Gridworld

We conduct experiments on Gridworld to quantify the discrepancy between target and sampled distributions. In these experiments, we measure the L1 distance between the true distributions and the generative distributions, following the methodology outlined in previous works (Bengio et al., 2021).

As shown in the tables below, Logit-GFN (both Layer-conditioning and Logit-only) achieves more accurate sampling compared to Layer-GFN and GFlowNet. Moreover, layer-conditioning yields the smallest L1 distance from the true underlying distribution. This suggests that integrating logit-scaling with the layer-conditioning method enhances the expressive power for temperature conditioning.

Table 4. Experiments on GridWorld (n=3, H=32). Train with $\beta \sim U[1,3]$ for Temperature-conditional GFlowNets, GFN are trained with specialized β.

L1($\times 10^{-5}$)	$\beta=1$	$\beta=2$	$\beta=3$	$\beta=4$	$\beta=5$
Logit-GFN (Layer-conditioning)	**3.607 ± 0.074**	**2.376 ± 0.231**	**1.490 ± 0.075**	**1.284 ± 0.199**	**1.957 ± 0.470**
Logit-GFN (Logit only)	3.926 ± 0.043	2.777 ± 0.119	2.130 ± 0.158	2.426 ± 0.114	2.579 ± 0.160
Layer-GFN	3.621 ± 0.025	2.470 ± 0.228	1.479 ± 0.193	2.926 ± 0.849	5.507 ± 0.039
GFN	3.689 ± 0.037	2.484 ± 0.071	5.363 ± 0.003	5.370 ± 0.001	5.363 ± 0.001

Table 5. Experiments on GridWorld (n=4, H=16). Train with $\beta \sim U[1,3]$ for Temperature-conditional GFlowNets, GFN are trained with specialized β.

L1($\times 10^{-5}$)	$\beta=1$	$\beta=2$	$\beta=3$	$\beta=4$	$\beta=5$
Logit-GFN (Layer-conditioning)	**2.392 ± 0.020**	**1.396 ± 0.033**	**0.733 ± 0.047**	**0.908 ± 0.011**	1.226 ± 0.050
Logit-GFN (Logit only)	2.397 ± 0.017	1.463 ± 0.035	1.099 ± 0.008	1.210 ± 0.046	**1.190 ± 0.010**
Layer-GFN	2.415 ± 0.020	1.416 ± 0.033	0.840 ± 0.062	1.644 ± 0.035	2.902 ± 0.018
GFN	2.426 ± 0.015	1.403 ± 0.011	0.812 ± 0.022	3.060 ± 0.001	3.075 ± 0.000

C.2. Experiments on offline generalization

Figure 18 visualizes the results of Logit-GFN with layer-conditioning in offline generalization tasks. As depicted in the figure, Logit-GFN with layer-conditioning performs similarly or occasionally better than Logit-GFN without layer-conditioning. In certain scenarios, incorporating a concatenated path can enhance the performance of Logit-GFN. This improvement can be attributed to the simplicity and ease of training of the logit-only method, which results in good performance despite its limited expressiveness compared to Logit-GFN with layer-conditioning.

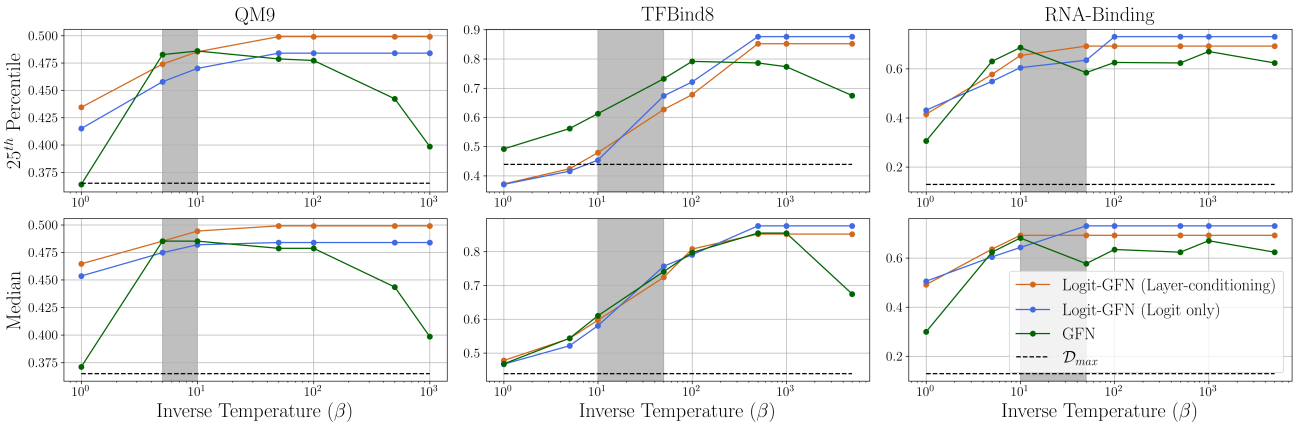

Figure 18. Ablation on Layer-conditioning with Logit Scaling in offline generalization.

C.3. Experiments on online mode-seeking

We also conducted experiments to evaluate the effect of layer conditioning on Logit-GFN in online mode-seeking problems. Figure 19 illustrates the number of modes discovered during training when applying layer-conditioning to Logit-GFN in the QM9 and TFBind8 tasks. As shown in the figure, while layer-conditioning has a marginal impact on the performance of Logit-GFN, it still outperforms Layer-GFN, demonstrating the superiority of logit scaling.

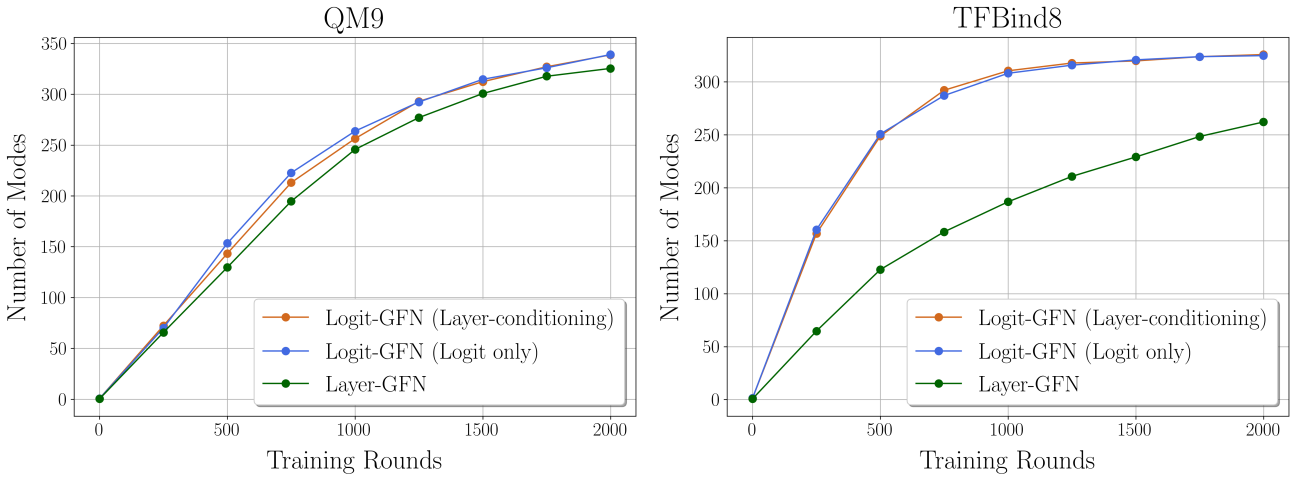

Figure 19. Ablation on Layer-conditioning with Logit Scaling in online mode seeking problems.

D. Online Discovery Algorithm

D.1. Algorithm PseudoCode

Algorithm 1 Scientific Discovery with Temperature-Conditional GFlowNets

Set $\mathcal{D} \leftarrow \emptyset$ ▷ *Initialize dataset.*
for $t = 1, \ldots, T$ **do** ▷ *Training T rounds*
  $\beta_1, \ldots, \beta_M \sim P_{\text{exp}}(\beta)$ ▷ *Sample temperatures from exploration query prior.*
  for $m = 1, \ldots, M$ **do**
    $\tau_m \sim P_F(\tau|\beta = \beta_m; \theta)$ ▷ *Sample trajectories from Logit-GFN.*
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_m\}$
  end for
  for $k = 1, \ldots K$ **do** ▷ *Training K epochs per each training rounds*
    Use ADAM for gradually minimizing $\mathcal{L}(\theta; \mathcal{D})$.
  end for
end for
Output: $\mathcal{D}$

We present the pseudocode of our online discovery algorithm in Algorithm 1. For each training round t, we sample M (inverse) temperatures from the query prior, $P_{\text{exp}}(\beta)$. Then, we generate trajectories using the policy $P_F(\cdot|\beta; \theta)$ conditioned on the sampled β values. Finally, we iterate the procedure by updating the policy parameters using the TB loss for K epochs per training round.

D.2. Ablation on different temperature sampling distribution

In this section, we present experiments conducted on different temperature sampling distributions, $P_{\text{exp}}(\beta)$, which is a crucial component for balancing exploration and exploitation in online mode seeking problems. While we use a uniform distribution as a default setting, there are various options available. We present extensive experiment results on various distributions we have tried in this research.

D.2.1. Stationary distributions

We first implement various stationary distributions. Among various options, we choose widely used distributions for evaluation as listed below:

- **Uniform**: This is a default setting of our algorithm. We sample β from the uniform distribution, i.e, $\beta \sim U^{[a,b]}$.
- **LogUniform**: We sample β from the loguniform distribution, which leads to sample high values of β with more probability, i.e, $\beta \sim \log(U^{[e^a, e^b]})$.
- **ExpUniform**: We sample β from the expuniform distribution, which leads to sample low values of β with more probability, i.e, $\beta \sim e^{U^{[\log a, \log b]}}$.
- **Normal**: We sample β from the normal distribution, i.e, $\beta \sim \mathcal{N}(\mu, \sigma^2)$.

Figure 20 (Left) shows the number of modes discovered during training with different query distributions. As shown in the figure, there is no big difference between uniform, lognormal and expuniform distribution. Normal distribution shows slightly better results than other distributions.

In the Figure 20 (Middle), we present the average reward computed by samples generated from the policy every 250 training rounds. We find that samples generated from LogUniform and Normal distribution generally give high rewards compared to a naive uniform distribution. Conversely, samples generated from ExpUniform distribution are placed in relatively low score regions. It tells us that our Logit-GFN can generate appropriate samples according to the conditioning variable, β. Figure 20 (Right) shows the histogram of generated β values for ease of understanding.

D.2.2. DYNAMIC DISTRIBUTIONS

We also apply dynamic temperature sampling distributions, which change the distribution across training. Among various options, we implement simulated annealing, one of the most widely used heuristics in combinatorial optimization. We prepare two variants of simulated annealing strategies as follows:

- **Simulated Annealing**: We first define uniform distribution with range 0.5, and shift the mean across the training rounds $t \in [1, T]$ from the lower bound to the upper bound, i.e, $\mu = (b-a) \times (t/T) + a,\ \beta \sim U^{[\max(a,\mu-0.5),\min(\mu+0.5,b)]}$.
- **Simulated Annealing (Inverse)**: In the inverse setting, we shift the mean across the training rounds from the upper bound to the lower bound, i.e, $\mu = (b-a) \times (1-t/T) + a,\ \beta \sim U^{[\max(a,\mu-0.5),\min(\mu+0.5,b)]}$.

As shown in the Figure 21, Simulated annealing demonstrates superior performance compared to a naive uniform distribution. While the inverse approach also exhibits fast convergence at the early stage, it encounters a significant decrease in efficiency as the rounds advance.

We also visualize the average reward of generated samples from the policy at every 250 training rounds in Figure 21. We observe that simulated annealing makes the average reward of samples drastically towards high reward regions. It seems exploration with low β at the early stage and focusing on exploitation with high β at the later stage can boost the performance in online mode seeking problems. Figure 21 (Right) shows the histogram of generated β values at training round $t = 800$ for ease of understanding.

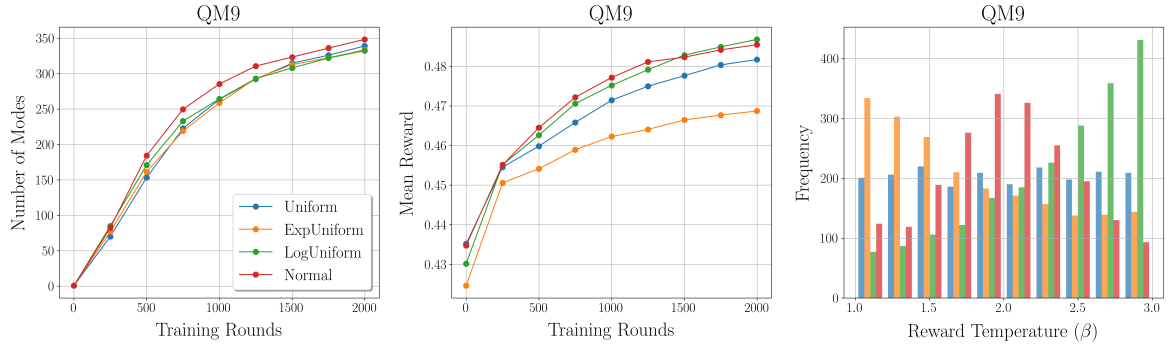

Figure 20. Ablation on different temperature sampling distribution. (LogUniform, ExpUniform, Normal)

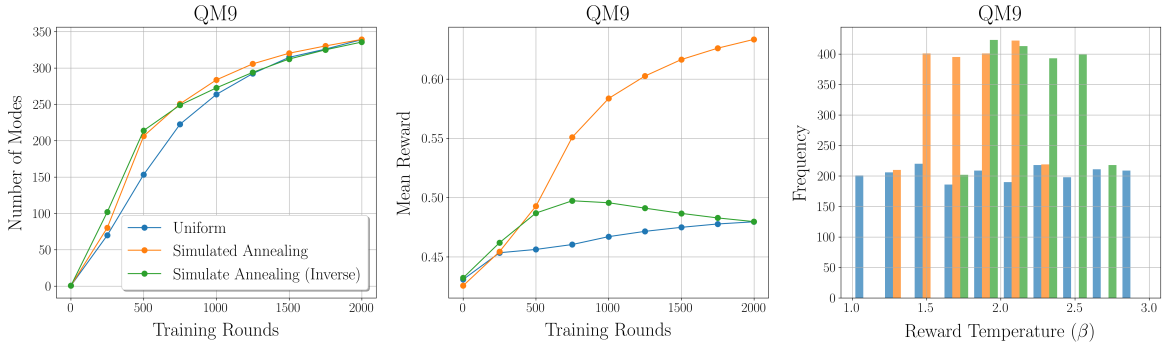

Figure 21. Ablation on different temperature sampling distribution. (Simulated Annealing)