GFlowNet Training by Policy Gradients

Puhua Niu [1] **Shili Wu** [1] **Mingzhou Fan** [1] **Xiaoning Qian** [1 2 3]

Abstract

Generative Flow Networks (GFlowNets) have been shown effective to generate combinatorial objects with desired properties. We here propose a new GFlowNet training framework, with policy-dependent rewards, that bridges keeping flow balance of GFlowNets to optimizing the expected accumulated reward in traditional Reinforcement-Learning (RL). This enables the derivation of new policy-based GFlowNet training methods, in contrast to existing ones resembling value-based RL. It is known that the design of backward policies in GFlowNet training affects efficiency. We further develop a coupled training strategy that jointly solves GFlowNet forward policy training and backward policy design. Performance analysis is provided with a theoretical guarantee of our policy-based GFlowNet training. Experiments on both simulated and real-world datasets verify that our policy-based strategies provide advanced RL perspectives for robust gradient estimation to improve GFlowNet performance. Our code is available at: github.com/niupuhua1234/GFN-PG.

1. Introduction

Generative Flow Networks (GFlowNets) are a family of generative models on the space of combinatorial objects $\mathcal{X}$, e.g. graphs composed by organizing nodes and edges in a particular manner, or strings composed of characters in a particular ordering. GFlowNets aim to solve a challenging task, sampling $x \in \mathcal{X}$ with a probability proportional to some non-negative reward function $R(x)$ that defines an unnormalized distribution, where $|\mathcal{X}|$ can be enormous and the distribution modes are highly isolated by its combinatorial nature. GFlowNets (Bengio et al., 2021; 2023) decompose the process of generating or sampling $x \in \mathcal{X}$ by generating incremental trajectories that start from a null state, pass through intermediate states, and end at x as the desired terminating state. These trajectory instances are interpreted as the paths along a Directed Acyclic Graph (DAG). Probability measures of trajectories are viewed as the amount of 'water' flows along the DAG, with $R(x)$ being the total flow of trajectories that end at x, so that following the forward generating policy defined by the measure, sampled trajectories will end at x with the probability proportional to $R(x)$.

GFlowNets bear a similar form of reinforcement learning (RL) in that they both operate over Markovian Decision Processes (MDP) with a reward function $R(x)$, where nodes, edges, and node transition distributions defined by Markovian flows are considered as states, actions, and stochastic policies in MDPs. They, however, differ in the following aspects: the goal of RL problems is to learn optimal policies that maximize the expected cumulative trajectory reward by R. For **value-based** RL methods, the key to achieve this is by reducing the Temporal Difference (TD) error of Bellman equations for the estimated state value function V and state-action value function Q (Sutton & Barto, 2018; Mnih et al., 2013). GFlowNets amortize the sampling problem into finding some Markovian flow that assigns the proper probability flow to edges (actions) so that the total flow of trajectories ending at x is $R(x)$. When studying these in the lens of RL, the existing GFlowNet training strategies are also value-based in that they achieve the goal by keeping the balance flow equation over states of the DAG, whose difference can be measured in trajectory-wise and edge-wise ways (Bengio et al., 2021; 2023; Malkin et al., 2022a; Madan et al., 2023).

Due to the similarity of GFlowNet training and RL, investigating the relationships between them can not only deepen understanding of GFlowNets but also help derive better training methods from RL. In this work, we propose policy-dependent rewards for GFlowNet training. This bridges GFlowNets to RL in that keeping the flow balance over DAGs can be reformulated as optimizing the expected accumulated rewards in RL problems. We then derive **policy-based** training strategies, which optimize the accumulated reward by its gradients with respect to (w.r.t.) the forward policy directly (Sutton et al., 1999; Sutton & Barto, 2018).

[1]Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA [2]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA [3]Computational Science Intiative, Brookhaven National Laboratory, Upton, NY, USA. Correspondence to: Xiaoning Qian <xqian@tamu.edu>.

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

In terms of RL, we acknowledge that the existing GFlowNet training methods can be considered value-based and have the advantage of allowing off-policy training over policy-based methods (Malkin et al., 2022b). Value-based methods, however, face the difficulty in designing a powerful sampler that can balance the exploration and exploitation trade-off, especially when the combinatorial space is enormous with well-isolated modes. Besides, employing typical annealing or random-mixing solutions may lead to the learned policy trapped in local optima. Finally, designing strategies for powerful samplers vary according to the structure and setting of modeling environments. Policy-based methods, especially the on-policy ones, transform the design of a powerful sampler into robust estimation of policy gradients, which can be achieved by variance reduction techniques (Schulman et al., 2016) and improvement of gradient descent directions, such as natural policy gradients (Kakade, 2001) and mirror policy descent (Zhan et al., 2023). Conservation policy updates such as Trust-Region Policy Optimization (TRPO) (Schulman et al., 2015; Achiam et al., 2017) and its first-order approximation, Proximal Policy Optimization (PPO) (Schulman et al., 2017b), have also been developed, for example as the backbone of ChatGPT (Ouyang et al., 2022). Moreover, policy-based methods can be made off-policy, for example, by importance sampling (Degris et al., 2012). Our work provides alternative ways to improve GFlowNet performance via policy-based training. Our contributions can be summarized as follows:

- We reformulate the GFlowNet training as RL over a special MDP where the reward is policy-dependent, and the underlying Markovian chain is absorbing. We further derive policy gradients for this special MDP and propose policy-based training strategies for GFlowNets, inspired by policy gradient and TRPO methods for stationary reward.
- We further formulate the design of backward policies in GFlowNets as an RL problem and propose a coupled training strategy. While finding a desired forward policy is the goal of GFlowNet training, well-designed backward policies, as the components of training objectives, are expected to improve training efficiency (Shen et al., 2023).
- We provide performance analyses for theoretical guarantees of our method for GFlowNet training. Our theoretical results are also accompanied by performing experiments in three application domains, hyper-grid modeling, biological and molecular sequence design, and Bayesian Network (BN) structure learning. The obtained experimental results serve as empirical evidence for the validity of our work and also help empirically understand the relationship between GFlowNet training and RL.

2. Preliminaries

For notation compactness, we restrict DAGs of GFlowNets to be *graded*[1]. In a DAG, $\mathcal{G} := (\mathcal{S}, \mathcal{A})$, modeling a MDP of GFlowNets: $s \in \mathcal{S}$ denotes a state, $a \in \mathcal{A}$ denotes a directed edge/action $(s \to s')$, and $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$. Assuming that there is a topological ordering $\mathcal{S}_0, \ldots, \mathcal{S}_T$ for $T+1$ disjoint subsets of $\mathcal{S}$, then $\mathcal{S} = \bigcup_{t=0}^{T} \mathcal{S}_t$ and an element of $\mathcal{S}_t$ is denoted as s_t. We use $\{\prec, \succ, \preceq, \succeq\}$ to define the partial orders between states; for example, $\forall t < t' : s_t \prec s_{t'}$. Furthermore, being *acyclic* means $\forall (s \to s') \in \mathcal{A}$: $s \prec s'$. Being *graded* means $\mathcal{A}$ can be decomposed into $\bigcup_{t=0}^{T-1} \mathcal{A}_t$ where $\mathcal{A}_t \bigcap \mathcal{A}_{t' \neq t} = \emptyset$ and $a_t \in \mathcal{A}_t$ represents an edge $(s_t \to s_{t+1})$ connecting $\mathcal{S}_t$ and $\mathcal{S}_{t+1}$. For any $s \in \mathcal{S}$, we denote its parent set by $Pa_{\mathcal{G}}(s) = \{s' | (s' \to s) \in \mathcal{A}\}$ and its child set $Ch_{\mathcal{G}}(s) = \{s' | (s \to s') \in \mathcal{A}\}$. Correspondingly, We denote the edge sets that start and end at s as $\mathcal{A}(s) = \{(s \to s') | s' \in Ch_{\mathcal{G}}(s)\}$ and $\dot{\mathcal{A}}(s) = \{(s' \to s) | s' \in Pa_{\mathcal{G}}(s)\}$ respectively. The complete trajectory set is defined as $\mathcal{T} = \{\tau = (s_0 \to \cdots \to s_T) | \forall (s \to s') \in \tau : (s \to s') \in \mathcal{A}\}$. We use $\tau_{\succeq s}$ to denote the sub-trajectory that starts at s, and $\tau_{\geq t}$ the sub-trajectory that starts at s_t. For the DAG $\mathcal{G}$ in GFlowNets, we have two special states: the initial state s^0 with $Pa(s^0) = \emptyset$ and $S_0 = \{s^0\}$, and the final state s^f with $Ch(s^f) = \emptyset$ and $S_T = \{s^f\}$. Furthermore, the terminal state set, S_{T-1} covers the object set $\mathcal{X}$ with a reward function $R : \mathcal{X} \to \mathbb{R}^+$.

2.1. GFlowNets

GFlowNets aim at efficient sampling from $P^*(x) := \frac{R(x)}{Z^*}$, where $Z^* = \sum_{x \in \mathcal{X}} R(x)$ and directly computing Z^* is often challenging with typically large $|\mathcal{X}|$. To achieve this, GFlowNets define a measure $F(\tau) : \mathcal{T} \to \mathbb{R}^+$, termed as 'flow' (Bengio et al., 2023), so that for any event E, $F(E) = \sum_{\tau \in E} F(\tau)$ and the total flow $Z = F(s^0) = F(s^f)$. For any event E and E', $P(E) := F(E)/Z$ and $P(E|E') := \frac{F(E \cap E')}{F(E')}$. Furthermore, F is restricted to be Markovian, which means $\forall \tau \in \mathcal{T}$:

$$P(\tau) = \prod_{t=1}^{T} P(s_{t-1} \to s_t | s_{t-1}) = \prod_{t=1}^{T} \frac{F(s_{t-1} \to s_t)}{F(s_{t-1})}, \quad (1)$$

where $F(s \to s') = \sum_{\tau \in \{\tau | (s \to s') \in \tau\}} F(\tau)$, $F(s) = \sum_{\tau \in \{\tau | s \in \tau\}} F(\tau)$ and $P_F(s_t | s_{t-1}) := P(s_{t-1} \to s_t | s_{t-1})$. Similarly, $P_B(s_{t-1} | s_t) := P(s_{t-1} \to s_t | s_t) = \frac{F(s_{t-1} \to s_t)}{F(s_t)}$. A desired generative flow F is set to have the terminal transition probability $P^{\mathsf{T}}(x) := P(x \to s_f)$ equal to $P^*(x)$. As shown in Bengio et al. (2023), the necessary

[1]Any DAG can be equivalently converted to be graded by adding dummy non-terminating states. Please refer to Appendix A of Malkin et al. (2022b) for more details.

and sufficient condition is that $\forall s' \in \mathcal{S} \setminus \{s^0, s^f\}$:

$$\sum_{s \in Pa(s')} F(s \to s') = \sum_{s'' \in Ch(s)} F(s' \to s''). \tag{2}$$

where $F(x \to s_f) := R(x)$ for any $x \in \mathcal{X}$.

2.2. GFlowNet training

Directly estimating the transition flow $F(s \to s')$ via the flow matching objective (Bengio et al., 2021) can suffer from the explosion of F values, of which the numerical issues may lead to the failure of model training. In practice, the Trajectory Balance (TB) objective has been shown to achieve the state-of-the-art training performance (Malkin et al., 2022a). With the TB objective, the desired flow is estimated by the total flow Z and a pair of forward/backward policies, $P_F(s'|s)$ and $P_B(s|s')$. The TB objective $\mathcal{L}_{TB}(P_\mathcal{D})$ of a trajectory data sampler $P_\mathcal{D}$ is defined as:

$$\begin{aligned} \mathcal{L}_{TB}(P_\mathcal{D}) &:= \mathbb{E}_{P_\mathcal{D}(\tau)}[L_{TB}(\tau)], \\ L_{TB}(\tau) &:= \left(\log \frac{P_F(\tau|s_0)Z}{P_B(\tau|x)R(x)}\right)^2. \end{aligned} \tag{3}$$

In the equation above, $P_F(\tau|s_0) = \prod_{t=1}^{T} P_F(s_t|s_{t-1})$ with $P_F(\tau) = P_F(\tau|s_0)$, $P_F^\top(x) := P_F(x \to s^f)$, and $P_F(\tau|x) = P_F(\tau|x \to s^f) = P_F(\tau)/P_F^\top(x)$. Correspondingly, $P_B(\tau|x) = P_B(\tau|x \to s^f) = \prod_{t=1}^{T-1} P_B(s_{t-1}|s_t)$, $P_B^\top(x) := P_B(x \to s^f)$ equal to $P^*(x)$, $P_B(\tau) = P_B^\top(x)P_B(\tau|x)$, and $P_B(\tau|s_0) = P_B(\tau)$. Furthermore, we define $\mu(s_0 = s^0) := Z/\widehat{Z}$ as the 1-categorical distribution over $\mathcal{S}_0$ so that $P_{F,\mu}(\tau) := P_F(\tau|s_0)\mu(s_0) = P_F(\tau)$, where $\widehat{Z}$ is the normalizing constant whose value is clamped[2] to Z. We define $P_{B,\rho}(\tau) := P_B(\tau|x)\rho(x)$ with an arbitrary distribution ρ over $\mathcal{X}$.

3. Policy gradients for GFlowNet training

Following Malkin et al. (2022b), we first extend the relationship between the GFlowNet training methods based on the TB objective and KL divergence. With the extended equivalence, we then introduce our policy-based and coupled training strategies for GFlowNets. Finally, we present theoretical analyses on our proposed strategies.

3.1. Gradient equivalence

When choosing trajectory sampler $P_\mathcal{D}(\tau)$ equal to $P_F(\tau)$, the gradient equivalence between using the KL divergence and TB objective has been proven (Malkin et al., 2022b). However, this forward gradient equivalence does not take the total flow estimator Z into account. Moreover, the backward gradient equivalence requires computing the expectation over $P^*(x)$, which is not feasible. In this work, we extend the proof of the gradient equivalence to take all gradients into account and remove the dependency on $P^*(x)$, while keeping feasible computation.

Proposition 3.1. *Given a forward policy $P_F(\cdot|\cdot;\theta)$, a backward policy $P_B(\cdot|\cdot;\phi)$, and a total flow estimator $Z(\theta)$, the gradient of the TB objective*[3] *can be written as:*

$$\begin{aligned} \frac{\nabla_\theta \mathcal{L}_{TB}(P_{F,\mu};\theta)}{2} &= \nabla_\theta D_{KL}^{\mu(\cdot;\theta)}(P_F(\tau|s_0;\theta), P_B(\tau|s_0)) \\ &\quad + \frac{1}{2}\nabla_\theta \left(\log Z(\theta) - \log Z^*\right)^2 \\ &= \nabla_\theta D_{KL}^{\mu(\cdot;\theta)}(P_F(\tau|s_0;\theta), \widetilde{P}_B(\tau|s_0;\theta)); \\ \frac{\nabla_\phi \mathcal{L}_{TB}(P_{B,\rho};\phi)}{2} &= \nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), P_F(\tau|x)) \\ &= \nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), \widetilde{P}_F(\tau|x)). \end{aligned} \tag{4}$$

In the equations above, $\widetilde{P}_F(\tau|x) := P_F(\tau_{\preceq x}) = \prod_{t=1}^{T-1} P_F(s_t|s_{t-1})$ and $\widetilde{P}_B(\tau|s_0) := P_B(\tau|x)R(x)/Z$, denoting two unnormalized distributions of $P_F(\tau|x)$ and $P_B(\tau|s_0)$. For arbitrary distributions p, q, and u, $D_{KL}^u(p(\cdot|s), q(\cdot|s)) := \mathbb{E}_{u(s)}[D_{KL}(p(\cdot|s), q(\cdot|s))]$.

The proof is provided in Appendix A.1. As the TB objective is a special case of the Sub-Trajectory Balance (Sub-TB) objective (Madan et al., 2023), we also provide the proof of the gradient equivalence with respect to the Sub-TB objective in Appendix A.3, where the initial distribution μ becomes more flexible.

3.2. RL formulation of GFlowNet training

Inspired by the equivalence relationship in Proposition 3.1, we propose new reward functions that allow us to formulate GFlowNet training as RL problems with corresponding policy-based training strategies.

Definition 3.2 (Policy-dependent Rewards)**.** For any action $a = (s \to s') \in \mathcal{A}(s)$ $(a \in \dot{\mathcal{A}}(s'))$, we define two reward functions as:

$$\begin{aligned} R_F(s, a; \theta) &:= \log \frac{\pi_F(s, a; \theta)}{\pi_B(s', a; \theta)}, \\ R_B(s', a; \phi) &:= \log \frac{\pi_B(s', a; \phi)}{\pi_F(s, a)}, \end{aligned} \tag{5}$$

where $\pi_F(s, a; \theta) := P_F(s'|s; \theta), \pi_B(s', a; \phi) := P_B(s|s'; \phi)$, $\pi_B(s^f, a)$ is equal to $R(x)/Z$ for $a = (x \to s_f)$. For any $a \notin \mathcal{A}(s)$, $R_F(s, a) := 0$. For any $a \notin \dot{\mathcal{A}}(s')$, $R_B(s', a) := 0$.

[2] For any parametrized function $f(\cdot;\theta)$ and $\hat{f}(\cdot)$ clamped to f, $\hat{f}$ is equal to f, but regarded as constant w.r.t. θ during gradient computation.

[3] Training via TB was intrinsically done in an off-policy setting, so $\nabla \mathcal{L}_{TB}(P_\mathcal{D}) = \mathbb{E}_{P_\mathcal{D}(\tau)}[\nabla L_{TB}(\tau)]$ for any choice of $P_\mathcal{D}$.

Tuples $(\mathcal{S}, \mathcal{A}, \mathcal{G}, R_F)$ and $(\mathcal{S}, \dot{\mathcal{A}}, \mathcal{G}, R_B)$ specify two MDPs with policy-dependent rewards. In the MDPs, $\mathcal{G}$ specifies a deterministic transition environment such that $P(s'|s,a) = \mathbb{I}[(s \to s') = a]$ with the indicator function $\mathbb{I}$. $(\mathcal{G}, \pi_F)$ and $(\mathcal{G}, \pi_B)$ correspond to two absorbing Markovian chains. Accordingly, the nature of DAGs requires that each state has only one order index, allowing us to define time-invariant expected value functions of states and state-action pairs, which are defined as $V_F(s) := \mathbb{E}_{P_F(\tau_{>t}|s_t)}[\sum_{l=t}^{T-1} R_F(s_l, a_l)|s_t = s]$ and $Q_F(s,a) := \mathbb{E}_{P_F(\tau_{>t+1}|s_t,a_t)}[\sum_{l=t}^{T-1} R_F(s_l, a_l)|s_t = s, a_t = a]$. Then we define $J_F := \mathbb{E}_{\mu(s_0)}[V_F(s_0)]$, $A_F(s,a) := Q_F(s,a) - V_F(s)$, and $d_{F,\mu}(s) := \frac{1}{T}\sum_{t=0}^{T-1} P_F(s_t = s)$. We likewise denote the functions for the backward policy as $\{V_B, Q_B, J_B, A_B, d_{B,\rho}\}$. More details are provided in Appendix B.1. By definition, $V_F(s_0) = \mathbb{E}_{P(\tau|s_0)}[\sum_{t=0}^{T-1} R_F(s_t, a_t)] = D_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0))$, so $J_F = D^{\mu}_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0))$. Likewise, we can obtain $J_B = D^{\rho}_{KL}(P_B(\tau|x), \widetilde{P}_F(\tau|x))$. Thus, we can conclude that GFlowNet training can be converted into minimizing the expected value function J_F and J_B by Proposition 3.1. With the derived ∇J_F and ∇J_B provided in Appendix B.3, we update π_F, π_B, and μ to minimize J_F and J_B based on the correspondingly computed gradients of the following two objectives:

$$\mathbb{E}_{\mu(s_0;\theta)}[V_F(s_0)] + T\,\mathbb{E}_{d_{F,\mu}(s),\pi_F(s,a;\theta)}\left[A_F(s,a)\right], \\ (T-1)\,\mathbb{E}_{d_{B,\rho}(s),\pi_B(s,a;\phi)}\left[A_B(s,a)\right]. \quad (6)$$

Our policy-based method generalizes the TB-based training with $\pi_{\mathcal{D}}$ equal to π_F as follows: TB-based training corresponds to approximating $A(s,a)$ for $(s_t = s, a_t = a)$ empirically by $\widehat{Q}_F(s_t, a_t) - C$, where $\widehat{Q}_F(s_t, a_t) = \sum_{l=t}^{T-1} R_F(s_l, a_l)$, and C is a constant baseline for variance reduction. For comparison, our policy-based method can be considered approximating $A_F(s,a)$ functionally by $\widehat{A}^{\lambda}_F(s,a) = \sum_{l=t}^{T-1} \lambda^{l-t}(\widehat{Q}_F(s_l, a_l) - \widetilde{V}_F(s_l))$, where $\lambda \in [0,1]$ controls the **bias-variance trade-off** for gradient estimation (Schulman et al., 2016), $\widehat{Q}_F(s_l, a_l) = R_F(s_l, a_l) + \widetilde{V}_F(s_{l+1})$, and $\widetilde{V}_F(s_l)$ is a functional approximation of exact $V_F(s_l)$, serving as a functional baseline. Specifically, our policy-based method with $\lambda = 1$ can provide unbiased gradient estimation of ∇J_F as the TB-based method. This supports the stability of our policy-based method with the theoretical convergence guarantee by Theorem 3.7 in Section 3.4. A formal discussion of their connection is provided in Appendix B.4 and B.6. Additionally, we discuss the relationship between our method and traditional Maximum Entropy (MaxEnt) RL in Appendix B.5.

To further exemplify that the proposed rewards bridge policy-based RL techniques to GFlowNet training, we specifically focus on the TRPO method, whose performance

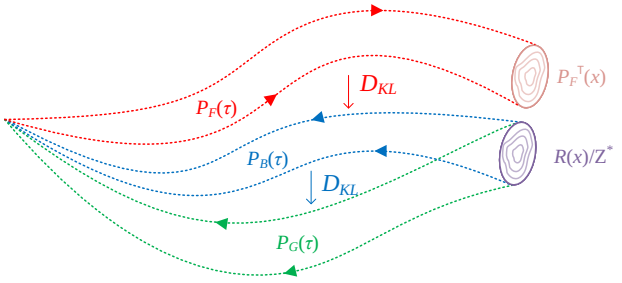

Figure 1. Dotted lines illustrate the spanning range of trajectories. P_B and P_G share the ground-truth terminating distribution $R(x)/Z^*$. When pushing P_F to match P_B trajectory-wise, $P_F^{\top}(x)$ will also be pushed to match $R(x)/Z^*$.

is usually more stable than vanilla policy-based methods due to conservative model updating rules (Schulman et al., 2015; Achiam et al., 2017). Likewise, we propose a TRPO-based objective for updating π_F:

$$\min_{\theta'} T\,\mathbb{E}_{d_{F,\mu}(s;\theta),\pi_F(s,a;\theta')}\left[A_F(s,a;\theta)\right] \\ \text{s.t. } D_{KL}^{d_{F,\mu}(\cdot;\theta)}\left(\pi_F(s,a;\theta), \pi_F(s,a;\theta')\right) \le \zeta_F. \quad (7)$$

The objective for π_B can be defined similarly and is omitted here. This objective is motivated as the approximation of the upper bound in Theorem 3.6, which generalizes the original results for static rewards and absorbing MDPs. We defer the discussion of their relationship in Section 3.4.

Moreover, the model parameter updating rule based on ∇J_F can be written as $\theta' \leftarrow \theta - \alpha\nabla_\theta J_F(\theta)$ or equivalently $\theta' = \text{argmin}_{\theta'}(\nabla J_F)^T\theta' + \frac{1}{2\alpha}\|\theta' - \theta\|_2^2$. Here, $\|\theta - \theta'\|_2$ can be generalized to KL divergence or Bregman divergence corresponding to natural or mirror policy gradients, which we leave for future work.

Details of the model parameter updating rules for proposed methods are provided in Appendix B.6.

3.3. RL formulation of guided backward policy design

During GFlowNet training, (P_B, R) specifies the amount of desired flow that (P_F, Z) is optimized to match. While $P_B(\cdot|\cdot)$ can be chosen freely in principle (Bengio et al., 2023), a well-designed P_B that assigns high probabilities over sub-trajectories preceding the terminating state x with a high reward value $R(x)$, will improve training efficiency. Following Shen et al. (2023), we formulate the design problem as minimizing the following objective:

$$\mathcal{L}^G_{TB}(P^{\rho}_B) := \mathbb{E}_{P^{\rho}_B(\tau)}[L^G_{TB}(\tau)], \\ L^G_{TB}(\tau;\phi) := \left(\log \frac{P_B(\tau|x;\phi)}{P_G(\tau|x)}\right)^2, \quad (8)$$

where $P_G(\tau|x) = \prod_{t=1}^{T-1} P_G(s_{t-1}|\tau_{\ge t})$ is called the conditional guided trajectory distribution, which is usually non-

Markovian[4], and $P_G(\tau) = P_G(\tau|x)P^*(x)$. As required by the training w.r.t. P_F, the objective $\mathcal{L}_{TB}^G$ aims at finding the backward policy whose Markovian flow best matches the non-Markovian flow induced by P_G.

Proposition 3.3. *Given a conditional guided trajectory distribution $P_G(\tau|x)$ and a backward policy $P_B(\cdot|\cdot;\phi)$, the gradients of $\mathcal{L}_{TB}^G$ can be written as:*

$$\frac{\nabla_\phi \mathcal{L}_{TB}^G(P_B^\rho;\phi)}{2} = \nabla_\phi D_{KL}^\rho(P_B(\tau|x;\phi), P_G(\tau|x)). \quad (9)$$

The proof can be found in Appendix A.2. Based on the proposition, we propose a new reward that allows us to formulate the backward policy design problem as an RL problem.

Definition 3.4. Given $P_G(\tau|x)$, we define a reward function for any action $a := (s \to s') \in \dot{\mathcal{A}}(s')$ as:

$$R_B^G(s', a; \phi) := \log \frac{\pi_B(s', a; \phi)}{\pi_G(s', a)}, \quad (10)$$

where $\pi_G(s', a) := P_G(s|\tau_{\succeq s'})$. For any $a \notin \dot{\mathcal{A}}(s')$, $R_B^G(s', a) := 0$.

Accordingly, we denote the associated function set as $\{V_B^G, Q_B^G, J_B^G, A_B^G, d_{B,\rho}^G\}$, which are defined in a similar way as R_B but replacing P_F by P_G. By the definition of J_B^G and Proposition 3.3, we can conclude that $\nabla_\phi J_B^G(\phi) = \frac{1}{2}\nabla_\phi \mathcal{L}_{TB}^G(P_B^\rho;\phi)$ and the design of backward policy can be solved by minimizing J_B^G. The form of P_G is detailed in Appendix E for the corresponding experimental tasks.

In principle, following the pipeline by Shen et al. (2023), we need to solve the optimization of $\mathcal{L}_{TB}^G$ to find the desired P_B at first. Then, freezing P_B, we can optimize $\mathcal{L}_{TB}$ to find the desired P_F. This gives rise to training inconvenience in practice. To avoid doing two-phase training, the authors mixed P_B and P_G by $\alpha P_B + (1-\alpha)P_G$ within the training objective w.r.t. P_F. This operation, however, lacks theoretical guarantees as the mixed distribution is still non-Markovian. By comparison, the RL formulation allows us to optimize J_F and J_B^G jointly with a theoretical performance guarantee, which we defer to the next section.

The workflow of our coupled training strategy is summarized in Algorithm 1 and depicted by Fig. 1.

3.4. Performance Analysis

In the previous sections, we formulate two RL problems with respect to R_F and R_B^G. Now, we show below that the two problems can be solved jointly.

[4]By non-Markovian assumption, $P_G(\tau|x)$ can factorize in arbitrary ways conditioning on x. Here it is assumed to factorize in the backward direction for notation compactness.

Algorithm 1 GFlowNet Training Workflow

Require: $P_F(\cdot|\cdot;\theta)$, $Z(\theta)$, $P_B(\cdot|\cdot;\phi)$, $P_G(\cdot|\cdot)$
  for $n = \{1, \dots, N\}$ **do**
    $\mathcal{D} \leftarrow \{\widehat{\tau}|\widehat{\tau} \sim P_F(\tau;\theta)\}$
    Update θ w.r.t. R_F and $\mathcal{D}$
    if $\phi \neq \emptyset$ **then**
      $\dot{\mathcal{D}} \leftarrow \{\widehat{\tau}|\forall x \in \mathcal{D} : \widehat{\tau}|x \sim P_B(\tau|x)\}$
      if $P_G(\widehat{\tau}|x) \neq P_B(\widehat{\tau}|x)$ **then**
        Update ϕ w.r.t. R_B^G and $\dot{\mathcal{D}}$
      else
        Update ϕ w.r.t. R_B and $\dot{\mathcal{D}}$
      end if
    end if
  end for

Theorem 3.5. *Denoting J_F^G as the corresponding function of R_F^G obtained by replacing π_B within R_F with π_G and choosing $\rho(x) = P_F^\top(x)$, then J_F^G, J_F and J_B^G satisfy the following inequality:*

$$J_F^G \leq J_F + J_B^G + (T-1)R_B^{G,\max}\sqrt{\frac{(J_F + \log Z^* - \log Z)}{2}}, \quad (11)$$

where $R_B^{G,\max} = \max_{s,a}\left|R_B^G(s,a)\right|$.

The proof is given in Appendix C.1. As shown in Proposition 3.1, minimization of J_F will incur the decrease of $D_{KL}^\mu(P_F(\tau|s_0), P_B(\tau|s_0)) = J_F + \log Z^* - \log Z$. Thus, by minimizing J_F and J_B^G jointly, the upper bound of J_F^G decreases.

Moreover, the TRPO-based objective introduced in the previous section is motivated by the following upper bounds.

Theorem 3.6. *For two forward policies (π_F, π_F') with $D_{KL}^{d_{F,\mu}'}(\pi_F'(s,\cdot), \pi_F(s,\cdot)) < \zeta_F$, and two backward policies (π_B, π_B') with $D_{KL}^{d_{B,\rho}'}(\pi_B'(s,\cdot), \pi_B(s,\cdot)) < \zeta_B$, we have:*

$$\frac{J_F' - J_F}{T} \leq \mathbb{E}_{d_{F,\mu}(s)\pi_F'(s,a)}[A_F(s,a)] + \zeta_F + \epsilon_F\sqrt{2\zeta_F},$$
$$\frac{J_B' - J_B}{T-1} \leq \mathbb{E}_{d_{B,\rho}(s)\pi_B'(s,a)}[A_B(s,a)] + \zeta_B + \epsilon_B\sqrt{2\zeta_B}, \quad (12)$$

where $\epsilon_F = \max_s\left|\mathbb{E}_{\pi_F'(s,a)}[A_F(s,a)]\right|$ and $\epsilon_B = \max_s\left|\mathbb{E}_{\pi_B'(s,a)}[A_B(s,a)]\right|$. Similar results also apply to J_B^G and A_B^G for the backward policy π_B.

The proof is given in Appendix C.2. The TRPO-based objective can be derived following a similar logic in Schulman et al. (2015) and Achiam et al. (2017). Let's denote $M(\pi) = \mathbb{E}_{d_{F,\mu}(s),\pi(s,a)}[A_F(s,a)] + \zeta_F + \epsilon_F\sqrt{2\zeta_F}$ and set $\pi_F' = \text{argmax}_\pi M(\pi)$. In the worst case, we choose $\pi_F' = \pi_F$ and $M(\pi_F') = 0$; then it can be expected that there is a conservative solution. That is, $\pi_F' \neq \pi_F$ and ζ_F

is negligibly small, so that $M(\pi'_F) < 0$, thereby resulting in $J'_F - J_F < 0$. This implies the monotonic performance gain. TRPO method is an approximation to this update and usually provides more stable performance gain than the vanilla policy-based method.

Lastly, we provide a theoretical guarantee that policy-based methods with policy-dependent rewards can asymptotically converge to stationary points, which draws inspiration from the results for static rewards by Agarwal et al. (2019).

Theorem 3.7. *Suppose that:* $J_F(\theta)$ *is* $\beta-$*smooth;* $\mathbb{E}_{P(\cdot|\theta)}[\widehat{\nabla}_\theta J_F(\theta)] = \nabla_\theta J_F(\theta)$*; the estimation variance,* $\mathbb{E}_{P(\cdot|\theta)}\left[\left\|\widehat{\nabla}_\theta J_F(\theta) - \nabla_\theta J_F(\theta)\right\|_2^2\right] \leq \sigma_F$*;* $|\log Z(\theta) - \log Z^*| \leq \sigma_Z$*; we update* θ *for* N $(> \beta)$ *iterations by* $\theta_{n+1} \leftarrow \theta_n - \alpha\widehat{\nabla} J_F(\theta_n)$ *with* $n \in \{0, \ldots, N-1\}$*,* $\alpha = \sqrt{1/(\beta N)}$ *and initial parameter* θ_0*. Then we have:*

$$\min_{n\in\{0,\ldots,N-1\}} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \frac{\sigma_F + \sigma_Z + \mathbb{E}_{P(\theta_0)}[J_F(\theta_0)]}{(\sqrt{(2N)/\beta} - 1)}. \quad (13)$$

Similar results also apply to J_B *and* J_B^G*.*

The proof is provided in Appendix C.3. The assumption $\mathbb{E}_{P(\cdot|\theta)}[\widehat{\nabla}_\theta J_F(\theta)] = \nabla_\theta J_F(\theta)$ means gradient estimation is unbiased as explained in Appendix B.6.

3.5. Related Work

GFlowNet training GFlowNets were first proposed by Bengio et al. (2021) and trained by a Flow Matching (FM) objective, which aims at minimizing the mismatch of equation (2) w.r.t. a parameterized edge flow estimator $F(s \to s')$ directly. Bengio et al. (2023) reformulated equation (2) and proposed a Detailed Balance (DB) objective, where edge flows $F(s \to s')$ are represented by $F(s)P_F(s'|s)$ or $F(s')P_B(s|s')$. Malkin et al. (2022a) claimed that the FM and DB objectives are prone to inefficient credit propagation across long trajectories and showed that the TB objective is the more efficient alternative. Madan et al. (2023) proposed a Sub-TB objective that unified the TB and DB objectives as special cases. They can be considered as Sub-TB objectives with sub-trajectories, which are complete or of length 1 respectively. Zimmermann et al. (2022) proposed KL-based training objectives and Malkin et al. (2022b) first established the equivalence between the KL and TB objectives. Shen et al. (2023) analyzed how the TB objective helps to learn the desired flow under the sequence prepend/append MDP setting, and proposed a guided TB objective. Forward-looking GFlowNets (Pan et al., 2023) improved the formulation of the DB objective by a better local credit assignment scheme, which was further generalized by learning energy decomposition GFlowNets (Jang et al., 2023). Finally, back-and-forth local search (Kim et al., 2023b), Thompson Sampling (TS) (Rector-Brooks et al., 2023), and temperature conditioning (Kim et al., 2023a) were proposed for the explicit design of $P_\mathcal{D}$.

Hierarchical Variational inference Hierarchical Variational Inference (HVI) (Vahdat & Kautz, 2020; Zimmermann et al., 2021) generalizes amortized VI (Zhang et al., 2018) to better explore specific statistical dependency structures between observed variables and latent variables by introducing the hierarchy of latent variables. Training HVI models typically involves minimizing the selected divergence measures between the target distribution and the variational distribution parametrized by neural networks (Kingma & Welling, 2014; Burda et al., 2015). GFlowNets can be considered as a special HVI model, where non-terminating states are latent variables, the hierarchy corresponds to a DAG, and the task of minimizing divergences is achieved by keeping flow balance (Malkin et al., 2022b). Our work provides another view of divergence minimization by interpreting the divergence as the expected accumulated reward.

Imitation learning Imitation learning in RL is to learn a policy that mimics the expert demonstrations with limited expert data, by minimizing the empirical gap between the learned policy and expert policy. (Rajaraman et al., 2020; Ho & Ermon, 2016). For GFlowNet training in this work, we reduce the gap between the forward policy and the expert forward policy at the trajectory level, as the expert trajectory distribution is equal to $P_B(\tau)$, implicitly encouraging the learned policy to match the desired expert policy.

Policy-based RL Policy-based RL optimizes the expected value function J directly based on policy gradients (Sutton et al., 1999). The most relevant policy-based methods are the Actor-Critic method (Sutton & Barto, 2018) and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) along with its extension – Constrained Policy Optimization (CPO) (Achiam et al., 2017). Compared to our methods, they work under the assumption that the reward functions must be fixed w.r.t. policies. The underlying Markov chains are further assumed to be ergodic by CPO and TRPO. We note that Weber et al. (2015) proposed a VI method based on policy gradient despite lacking experimental support. Here, the objective can be interpreted as the KL divergence between two forward trajectory distributions. Without the help of A, the policy gradient is estimated in a vanilla manner, corresponding to $\widehat{A}^1$ and $\widehat{A}^0$. Besides, Rengarajan et al. (2022) proposed a TRPO method for imitation learning, where the objective is the expected KL divergence between two forward policies, and the underlying MDP is assumed to be ergodic as the original method.

MaxEnt RL Bengio et al. (2021) have shown that directly applying MaxEnt RL with fixed $R(s,a)$ defined based on the terminating state is problematic as it corresponds to modeling $p(x) \propto n(x)R(x)$, where $n(x)$ is the number of trajectories that can pass through x. As discussed in Appendix B.5, our policy-based methods, when fixing $\log \pi_B(s',a)$ and $\log Z$, can be connected to Soft-Q-learning, a typical MaxEnt RL method.

Bi-level optimization Our proposed training strategy can also be seen as a Stochastic Bi-level Optimization method for GFlowNet training (Ji et al., 2021; Hong et al., 2023; Ghadimi & Wang, 2018). The inner problem is the RL problem w.r.t. R_B or R_B^G for designing backward policies. The outer problem is the RL problem w.r.t. R_F for forward policies. For gradient-based solutions to Bi-level optimization in general, the learning rate of inner problems is carefully selected to guarantee the overall convergence, which is not required in our methods designed for GFlowNet training.

Additional discussion about policy-based and valued-based methods in the context of RL is provided in Appendix D.

4. Experiments

To compare our policy-based training strategies for GFlowNets with the existing value-based methods, we have conducted three simulated experiments for hyper-grid modeling, four real-world experiments for biological and molecular sequence design, one on Bayesian Network structure learning, and ablation study of λ. We compare the performance of GFlowNets by the following training strategies: (1) DB-U, (2) DB-B, (3) TB-U, (4) TB-B, (5) TB-Sub, (6) TB-TS, (7) RL-U, (8) RL-B; (9) RL-T and (10) RL-G, where notion '-U' means that π_B is a fixed uniform policy; '-B' means that π_B is a parameterized policy; 'RL' represent our policy-based method; '-T' represent our TRPO-based method with a uniform π_B and '-G' represent our joint training strategy with guided policy;'-Sub' represent the weighted Sub-TB objective with a parameterized π_B in Madan et al. (2023); '-TS' represent the TS objective with a parameterized π_B in Rector-Brooks et al. (2023). By default, $P_{\mathcal{D}}$ is γ-decayed-noisy for valued-based methods. Total variation D_{TV}, Jensen–Shannon divergence D_{JSD}, and mode accuracy Acc are used to measure the gap between $P_F^{\top}(x)$ and $P^*(x)$. Detailed descriptions of experimental settings, including metric definitions, guided policy design, hyper-parameters, etc., can be found in Appendix E. Our implementation is built upon the *torchgfn* package (Lahlou et al., 2023).

4.1. Hyper-grid Modeling

In this set of experiments, we use the hyper-grid environment following Malkin et al. (2022b). In terms of GFlowNets, states are the coordinate tuples of an D-dimensional hyper-cubic grid with heights equal to N. The initial state s^0 is $(0,\ldots,0)$. Starting from s^0, actions correspond to increasing one of D coordinates by 1 for the current state or stopping the process at the current state and outputting it as the terminating state x. A manually designed reward function $R(\cdot)$ assigns high reward values to some grid points while assigning low values to others. We conduct experiments on 256×256, 128×128, $64 \times 64 \times 64$, and $32 \times 32 \times 32 \times 32$ grids. For performance evaluation, $P_F^{\top}(x)$ is computed exactly by dynamic programming (Malkin et al., 2022b).

The training curves by D_{TV} across five runs for 256×256 and 128×128 grids are plotted in Fig. 2, and Table 1 in Appendix E.5 reports the mean and standard deviation of metric values at the last iteration. The graphical illustrations of $P_F^{\top}(x)$ are shown in Figs. 12 and 13 in Appendix E.6. **In the first setting**, it can be observed that our policy-based methods, in terms of convergence rate or converged D_{TV}, perform much better than all the considered value-based training methods. This shows that our policy-based training strategies give a more robust gradient estimation. Besides, RL-G achieves the smallest D_{TV} and converges much faster than all the other competing methods. In RL-G, the guided distribution assigns small values to the probability of terminating at coordinates with low rewards. This prevents the forward policy from falling into the reward 'desert' between the isolated modes. Finally, RL-T outperforms RL-U and behaves more stably than RL-U during training. This confirms that with the help of trust regions, the gradient estimator becomes less sensitive to estimation noises. Here we use a fixed constant ζ_F for trust region control. It is expected that using a proper scheduler of ζ_F during training may further improve the performance of RL-T. **In the second setting**, the converged D_{TV} of policy-based and TB-based methods are similar and significantly better than those of DB-based methods. As expected, policy-based methods converge much faster than all the value-based methods. Thus, the results further support the effectiveness of our policy-based methods. Moreover, RL-G and RL-T achieve the second-best and the best convergence, and RL-T shows better stability than RL-U. This again shows the superiority of coupled and TRPO-based strategies, confirming our theoretical analysis conclusions.

More results and discussions for $64 \times 64 \times 64$ and $32 \times 32 \times 32 \times 32$ grids can be found in Appendix E.2.

4.2. Biological and Molecular Sequence Design

In this set of experiments, we use GFlowNets to generate nucleotide strings of length D and molecular graphs composed of D blocks according to given rewards. The initial state $s^0 := (-1,\ldots,-1)$ denotes an empty sequence. The

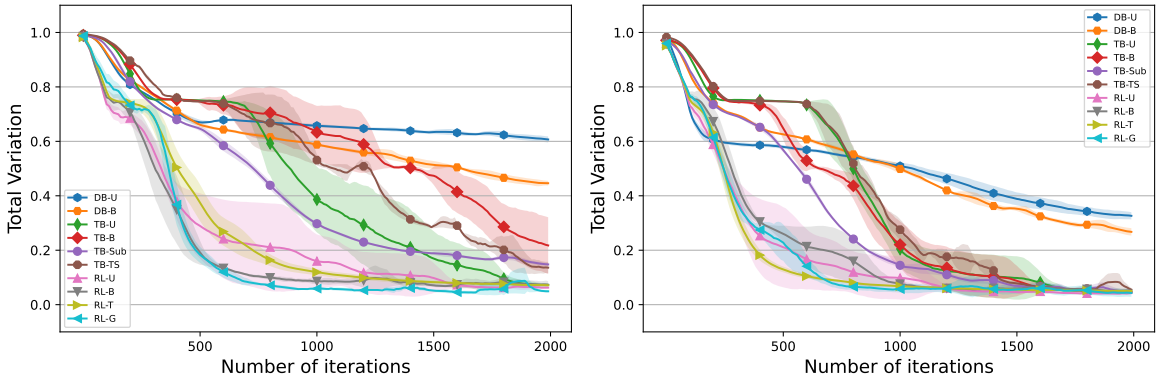

Figure 2. Training curves by D_{TV} between $P_F^{\top}$ and P^* for 256×256 (left) and 128×128 hyper-grids (right). The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

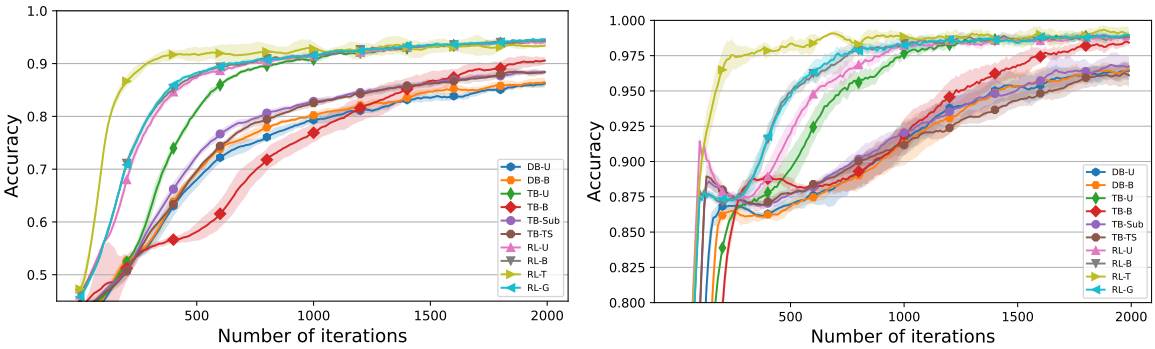

Figure 3. Training curves by Acc of $P_F^{\top}$ w.r.t. P^* for SIX6 (left) and QM9 (right) datasets. The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

generative process runs as follows: starting from s^0, an action is taken to pick one of the empty slots and fill it with one element until the sequence is completed. Then the sequence is returned as the terminating state x. We use nucleotide string datasets, SIX6 and PH04, and molecular graph datasets, QM9 and sEH, from Shen et al. (2023). For metric D_{TV} and D_{JSD}, $P_F^{\top}$ is computed exactly by dynamic programming.

Following Shen et al. (2023), the training curves by the mode accuracy Acc and the number of modes for SIX6 and QM9 datasets are shown in Fig. 3, and Fig. 7 in Appendix E.3. For evaluation consistency, we also provide the curves by D_{TV} in Fig. 6 in Appendix E.3, as well as the metric values at the last iteration summarized in Tables 3 and 4 in Appendix E.5. The graphical illustrations of $P_F^{\top}(x)$ are shown in Figs. 16 and 17 in Appendix E.6. In both experiments, TB-based and policy-based methods achieve better performance than DB-based methods. While the converged Acc values of TB-based methods and our policy-based methods are similar, the latter converge much faster than TB-based methods with only TB-U achieving a comparable convergence rate. Besides, RL-T has the fastest convergence rates in both experiments. The performances of RL-G are similar to those of RL-B, which has a parameterized π_B, but slightly better than RL-U with a uniform π_B. In summary, experimental results for QM9 and SIX6 datasets align with those of hyper-grid tasks, confirming again the advantage offered by our policy-based methods for robust gradient estimation.

More results and discussions for PHO4 and sEH datasets can be found in Appendix E.3.

4.3. Ablation Study of λ

To investigate how the setting of λ, which controls the bias-variance-trade-off, may help robust estimation of gradients, we conduct experiments in the 256×256 grid environment. We compare the performance of RL-U methods with different λ values and TB-U methods with different γ values The obtained training curves by D_{TV} across five runs are shown in Fig. 5 in Appendix E.2. Among the choices of γ for TB-U, the values 0.99 and 0.95 yield the best and

the worst performances. In contrast, RL-U under all setups except $\lambda = 1$, demonstrates significantly faster convergence than TB-U. It should be pointed out that when $\lambda = 1$, Q_F is approximated empirically as TB-based methods, but V_F is approximated functionally. Additionally, the converged D_{TV} in all setups of RL-U are better than those in all setups of TB-U. These results verify that by controlling λ, our policy-based methods can provide more robust gradient estimation than TB-based methods.

We have also conducted performance comparisons between policy-based and value-based methods for Bayesian network structure learning. The results and discussions can be found in Appendix E.4.

5. Conclusion, Limitations and Future Work

This work bridges the flow-balance-based GFlowNet training to RL problems. We have developed policy-based training strategies, which provide alternative ways to improve training performance compared to the existing value-based strategies. The experimental results support our claims. Our policy-based methods are not limited to the cases where $\mathcal{G}$ must be a DAG as it intrinsically corresponds to minimizing the KL divergence between two distributions, which does not necessitate $\mathcal{G}$ to be a DAG. Future work will focus on extending the proposed methods to general $\mathcal{G}$ with the existence of cycles for more flexible modeling of generative processes of object $x \in \mathcal{X}$. While our policy-based training strategies do not require an explicit design of a data sampler and are shown to achieve better GFlowNet training performance, they may still get trapped into local optima due to the variance of gradient estimation when the state space is very large. Thus, future research will also focus on further improving policy-based methods by more robust gradient estimation techniques, under the gradient equivalence relationship.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation (NSF) grants SHF-2215573, and by the U.S. Department of Engergy (DOE) Office of Science, Advanced Scientific Computing Research (ASCR) under Awards B&R# KJ0403010/FWP#CC132 and FWP#CC138. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

Impact Statement

The presented research aims to improve GFlowNet training methods to address the training performance challenge. The applications of our work encompass various societal realms, ranging from medicine to materials design.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Beck, A. *First-order methods in optimization*. SIAM, 2017.

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.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Degris, T., White, M., and Sutton, R. S. Off-policy actor-critic. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 179–186, 2012.

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.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Golpar Raboky, E. and Eftekhari, T. On nilpotent interval matrices. *Journal of Mathematical Modeling*, 7(2):251–261, 2019.

Grinstead, C. and Snell, L. J. *Introduction to probability*. 2006.

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.

Hestenes, M. R., Stiefel, E., et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.

Jang, H., Kim, M., and Ahn, S. Learning energy decompositions for partial inference of gflownets. In *The Twelfth International Conference on Learning Representations*, 2023.

Ji, K., Yang, J., and Liang, Y. Bilevel optimization: Convergence analysis and enhanced design. In *International conference on machine learning*, pp. 4882–4892. PMLR, 2021.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Kim, M., Ko, J., Zhang, D., Pan, L., Yun, T., Kim, W. C., Park, J., and Bengio, Y. Learning to scale logits for temperature-conditional gflownets. In *NeurIPS 2023 AI for Science Workshop*, 2023a.

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*, 2023b.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *ICLR*, 2014. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2014.html#KingmaW13.

Kuipers, J., Moffa, G., and Heckerman, D. Addendum on the scoring of gaussian directed acyclic graphical models. 2014.

Lahlou, S., Viviano, J. D., and Schmidt, V. torchgfn: A pytorch gflownet library. *arXiv preprint arXiv:2305.14594*, 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., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

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, 2023.

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33: 2914–2924, 2020.

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. In *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2023.

Rengarajan, D., Vaidya, G., Sarvesh, A., Kalathil, D., and Shakkottai, S. Reinforcement learning with sparse rewards using guidance from offline demonstration. *arXiv preprint arXiv:2202.04628*, 2022.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

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

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.

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

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Tsitsiklis, J. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

Weber, T., Heess, N., Eslami, A., Schulman, J., Wingate, D., and Silver, D. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2015.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhan, W., Cen, S., Huang, B., Chen, Y., Lee, J. D., and Chi, Y. Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence. *SIAM Journal on Optimization*, 33(2):1061–1091, 2023.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Zimmermann, H., Wu, H., Esmaeili, B., and van de Meent, J.-W. Nested variational inference. *Advances in Neural Information Processing Systems*, 34:20423–20435, 2021.

Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. A variational perspective on generative flow networks. *Transactions on Machine Learning Research*, 2022.

A. Gradient Equivalence

Lemma A.1. *(REINFORCE trick (Williams, 1992)) Given a random variable u following a distribution $p(\cdot;\psi)$ parameterized by ψ and a arbitrary function f, we have $\nabla_\psi \mathbb{E}_{p(u;\psi)}[f(u)] = \mathbb{E}_{p(u)}[f(u)\nabla_\psi \log p(u;\psi)] = \mathbb{E}_{p(u)}[f(u)\nabla_\psi \log \tilde{p}(u;\psi)]$, where $p(u;\phi) = \tilde{p}(u;\phi)/\hat{Z}_p$, and $\hat{Z}_p$ is the normalizing constant and clamped to $\sum_u \tilde{p}(u)$.*

A.1. Proof of Proposition 3.1

Proof. First of all, we split the parameters of the total flow estimator and forward transition probability and denote them as $Z(\theta_Z)$ and $P_F(\cdot|\cdot;\theta_F)$ respectively. We further define $c(\tau) = \left(\log \frac{P_F(\tau|s_0)}{R(x)P_B(\tau|x)}\right)$.

For the gradients w.r.t. θ_F:

$$\begin{aligned}
&\frac{1}{2}\nabla_{\theta_F}\mathbb{E}_{P_F(\tau|s_0)}[L_{TB}(\tau;\theta_F)] = \frac{1}{2}\mathbb{E}_{P_{F,\mu}(\tau)}\left[\nabla_{\theta_F}\left(c(\tau;\theta_F)+\log Z\right)^2\right] \\
&= \mathbb{E}_{P_{F,\mu}(\tau)}\left[\left(c(\tau)+\log Z\right)\nabla_{\theta_F}\log P_F(\tau|s_0;\theta_F)\right] \\
&= \mathbb{E}_{P_{F,\mu}(\tau)}\left[\left(c(\tau)+\log Z\right)\nabla_{\theta_F}\log P_F(\tau|s_0;\theta_F)\right] + \underbrace{\mathbb{E}_{P_{F,\mu}(\tau)}\left[\nabla_{\theta_F}\left(c(\tau;\theta_F)+\log Z\right)\right]}_{(a)} \\
&= \mathbb{E}_{\mu(s_0)}\left[\nabla_{\theta_F}\mathbb{E}_{P_F(\tau|s_0;\theta_F)}\left[c(\tau;\theta_F)+\log Z\right]\right] \\
&= \mathbb{E}_{\mu(s_0)}[\nabla_{\theta_F} D_{KL}(P_F(\tau|s_0;\theta_F), \widetilde{P}_B(\tau|s_0))] \\
&= \nabla_{\theta_F} D^{\mu}_{KL}(P_F(\tau|s_0;\theta_F), \widetilde{P}_B(\tau|s_0)),
\end{aligned} \tag{14}$$

where (a) is equal to zero as $\mathbb{E}_{P_F(\tau|s_0)}[\nabla_{\theta_F} c(\tau;\theta_F)] = \mathbb{E}_{P_F(\tau|s_0)}[1\cdot\nabla_{\theta_F}\log P_F(\tau|s_0;\theta_F)] = \nabla_{\theta_F}\mathbb{E}_{P_F(\tau|s_0;\theta_F)}[1] = 0$ by Lemma A.1. We also have:

$$\begin{aligned}
\nabla_{\theta_F} D^{\mu}_{KL}(P_F(\tau|s_0;\theta_F), \widetilde{P}_B(\tau|s_0)) =& \nabla_{\theta_F} D^{\mu}_{KL}(P_F(\tau|s_0;\theta_F), \widetilde{P}_B(\tau|s_0)) + \underbrace{\nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\tau;\theta_F)}\left[\log Z^* - \log Z\right]}_{=0} \\
=& \nabla_{\theta_F} D^{\mu}_{KL}(P_F(\tau|s_0;\theta_F), P_B(\tau|s_0)).
\end{aligned} \tag{15}$$

It should be emphasized that $P_B(\tau)$ is the ground-truth distribution with $P_B(x) := R(x)/Z^*$, while $\widetilde{P}_B(\tau)$ is the approximated one with $\widetilde{P}_B(x) := R(x)/Z$. The gradients w.r.t. θ_Z can be written as:

$$\begin{aligned}
\frac{1}{2}\nabla_{\theta_Z}\mathbb{E}_{P_F(\tau|s_0)}[L_{TB}(\tau;\theta_Z)] &= \frac{1}{2}\mathbb{E}_{P_F(\tau|s_0)}\left[\nabla_{\theta_Z}\left(c(\tau)+\log Z(\theta_Z)\right)^2\right] \\
&= \mathbb{E}_{P_F(\tau|s_0)}\left[\left(c(\tau)+\log Z\right)\nabla_{\theta_Z}\log Z(\theta_Z)\right] \\
&= \left[D_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0))\right]\left[\nabla_{\theta_Z}\log Z(\theta_Z)\right] \\
&= \nabla_{\theta_Z}\frac{Z(\theta_Z)}{\widehat{Z}} D_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0)) \\
&= \nabla_{\theta_Z} D^{\mu(\cdot;\theta_Z)}_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0)).
\end{aligned} \tag{16}$$

Besides, we have:

$$\begin{aligned}
\nabla_{\theta_Z} D^{\mu(\cdot;\theta_Z)}_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0)) =& \left[\nabla_{\theta_Z}\log Z(\theta_Z)\right]\left[D_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0)) + \log\frac{Z^*}{Z^*}\right] \\
=& \left[\nabla_{\theta_Z}\log Z(\theta_Z)\right]\left[D_{KL}(P_F(\tau|s_0), P_B(\tau|s_0)) + \log\frac{Z}{Z^*}\right] \\
=& \nabla_{\theta_Z}\frac{Z(\theta_Z)}{\widehat{Z}}\left[D_{KL}(P_F(\tau|s_0), P_B(\tau|s_0))\right] + \left[\nabla_{\theta_Z}\log\frac{Z(\theta_Z)}{Z^*}\right]\left[\log\frac{Z}{Z^*}\right] \\
=& \nabla_{\theta_Z} D^{\mu(\cdot;\theta_Z)}_{KL}(P_F(\tau|s_0), P_B(\tau|s_0)) + \frac{1}{2}\nabla_{\theta_Z}\left(\log Z(\theta_Z) - \log Z^*\right)^2.
\end{aligned} \tag{17}$$

Combining equations (14) and (16), we obtain:

$$\frac{1}{2}\nabla_\theta \mathbb{E}_{P_{F,\mu}(\tau)}[L_{TB}(\tau;\theta)] = \nabla_\theta D^{\mu(\cdot;\theta)}_{KL}(P_F(\tau|s_0;\theta), \widetilde{P}_B(\tau|s_0)). \tag{18}$$

Combining equations (15) and (17), we obtain:

$$\frac{1}{2}\nabla_\theta \mathbb{E}_{P_{F,\mu}(\tau)}[L_{TB}(\tau;\theta)] = \nabla_\theta \left\{ D_{KL}^{\mu(\cdot;\theta)}(P_F(\tau|s_0;\theta), P_B(\tau|s_0)) + \frac{1}{2}\left(\log Z(\theta) - \log Z^*\right)^2 \right\}. \tag{19}$$

Now let's consider the backward gradients and denote $c(\tau) = \left(\log \frac{P_B(\tau|x)}{P_F(\tau_{\preceq x})}\right)$. Then,

$$\begin{aligned}
&\frac{1}{2}\nabla_\phi \mathbb{E}_{P_{B,\rho}(\tau)}[L_{TB}(\tau;\phi)] = \frac{1}{2}\mathbb{E}_{P_{B,\rho}(\tau)}\left[\nabla_\phi \left(c(\tau;\phi) + \log R(x) - \log Z - \log P_F(s^f|x)\right)^2\right] \\
&= \mathbb{E}_{P_{B,\rho}(\tau)}\left[\left(c(\tau) + \log R(x) - \log Z - \log P_F(s^f|x)\right)\nabla_\phi \log P_B(\tau|x;\phi)\right] \\
&= \mathbb{E}_{P_{B,\rho}(\tau)}\left[c(\tau)\nabla_\phi \log P_B(\tau|x;\phi)\right] + \mathbb{E}_{\rho(x)}\Big[\left(\log R(x) - \log Z - \log P_F(s^f|x)\right)\underbrace{\mathbb{E}_{P_B(\tau|x)}[\nabla_\phi \log P_B(\tau|x;\phi)]}_{=0 \text{ by Lemma A.1}}\Big] \\
&= \mathbb{E}_{P_{B,\rho}(\tau)}[c(\tau)\nabla_\phi \log P_B(\tau|x;\phi)] + \underbrace{\mathbb{E}_{P_{B,\rho}(\tau)}[\nabla_\phi c(\tau;\phi)]}_{=0 \text{ by Lemma A.1}} \\
&= \mathbb{E}_{\rho(x)}\left[\nabla_\phi D_{KL}(P_B(\tau|x;\phi), \widetilde{P}_F(\tau|x))\right] \\
&= \nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), \widetilde{P}_F(\tau|x)).
\end{aligned} \tag{20}$$

Besides, we have

$$\begin{aligned}
\nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), \widetilde{P}_F(\tau|x)) &= \nabla_\phi D_{KL}^{\rho}(P_{B,\rho}(\tau|x;\phi), \widetilde{P}_F(\tau|x)) + \mathbb{E}_{\rho(x)}\Big[\underbrace{\nabla_\phi \mathbb{E}_{P_B(\tau|x;\phi)}\left[\log(P_F^\top(x)/P_F(s^f|x))\right]}_{=\log(P_F^\top(x)/P_F(s^f|x))\nabla_\phi 1 = 0}\Big] \\
&= \nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), \widetilde{P}_F(\tau|x)) + \nabla_\phi \mathbb{E}_{P_{B,\rho}(\tau;\phi)}\left[\log(P_F^\top(x)/P_F(s^f|x))\right] \\
&= \nabla_\phi D_{KL}^{\rho}(P_B(\tau|x;\phi), P_F(\tau|x)).
\end{aligned} \tag{21}$$

Equations (20) and (21) are the expected results. □

A.2. Proof of Proposition 3.3

Proof. The proof can be done by a procedure similar to that of backward gradients in Proposition 3.1 by replacing $\widetilde{P}_F(\tau|x)$ with $P_G(\tau|x)$. □

A.3. Sub-trajectory equivalence

Proposition 2 in the paper by Malkin et al. (2022b) only considered the gradients of the Sub-TB objective (Madan et al., 2023) w.r.t. $P_F(\cdot|\cdot)$ and $P_B(\cdot|\cdot)$. We provide an extended proposition below that also takes the gradients w.r.t. state flow estimator $F(\cdot)$ into consideration. For any $m < n$ and $n, m \in \{1, T-1\}$, we denote the set of sub-trajectories that start at some state in $\mathcal{S}_m$ and end in some state in $\mathcal{S}_n$ as $\bar{\mathcal{T}} = \{\bar{\tau} = (s_m \to \ldots \to s_n) | \forall t \in \{m, \ldots, n-1\} : (s_t \to s_{t+1}) \in \mathcal{A}_t\}$. The sub-trajectory objective $\mathcal{L}_{Sub-TB}(P_\mathcal{D})$ is defined by:

$$\mathcal{L}_{Sub-TB}(P_\mathcal{D}) = \mathbb{E}_{P_\mathcal{D}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau})], \quad L_{Sub-TB}(\bar{\tau}) = \log\left(\frac{P_F(\bar{\tau}|s_m)F(s_m)}{P_B(\bar{\tau}|s_n)F(s_n)}\right)^2. \tag{22}$$

In the equations above, $P_F(\bar{\tau}|s_m) = \prod_{t=m}^{n-1} P_F(s_{t+1}|s_t)$, $P_B(\bar{\tau}|s_n) = \prod_{t=m}^{n-1} P_B(s_t|s_{t+1})$ and $F(s_n = x) := R(x)$. Besides, we define $\mu(s_m) := F(s_m)/\widehat{Z}_m$ and $\rho(s_n) := F(s_n)/\widehat{Z}_n$ where $\widehat{Z}_m$ and $\widehat{Z}_n$ are the two normalizing constants whose values are clamped to $\sum_{s_m} F(s_m)$ and $\sum_{s_n} F(s_n)$.

Furthermore, $P_{F,\mu}(\bar{\tau}) := \mu(s_m)P_F(\bar{\tau}|s_m)$ and $P_{B,\rho}(\bar{\tau}) := \rho(s_n)P_B(\bar{\tau}|s_n)$ so that $P_{F,\mu}(\bar{\tau}|s_n) = P_{F,\mu}(\bar{\tau})/\rho^*(s_n)$ and $P_{B,\rho}(\bar{\tau}|s_m) = P_{B,\rho}(\bar{\tau})/\mu^*(s_m)$. Here, $\rho^*(s_n) := F^*(s_n)/\widehat{Z}_n^*$, $\widehat{Z}_n^*$ is clamped to $\sum_{s_n} F^*(s_n)$, and $F^*(s_n) := \sum_{\bar{\tau}:s_n\in\bar{\tau}} F(s_m)P_F(\bar{\tau}|s_m)$ is the ground-truth state flow over $\mathcal{S}_n$ implied by P_F; $\mu^*(s_m) := F^*(s_m)/\widehat{Z}_m^*$, $\widehat{Z}_m^*$ is clamped to $\sum_{s_m} F^*(s_m)$, and $F^*(s_m) := \sum_{\bar{\tau}:s_m\in\bar{\tau}} F(s_n)P_B(\bar{\tau}|s_n)$ is the ground-truth state flow over $\mathcal{S}_m$ implied by P_B.

Proposition A.2. *For a forward policy $P_F(\cdot|\cdot;\theta)$, a backward policy $P_B(\cdot|\cdot;\phi)$, a state flow estimator $F(\cdot;\theta)$ for $\mathcal{S}_m$, and*

a state flow estimator $F(\cdot;\phi)$ for $\mathcal{S}_n$[5], the gradients of Sub-TB can be written as:

$$\begin{aligned}
\frac{1}{2}\nabla_\theta \mathcal{L}_{Sub-TB}(P_{F,\mu};\theta) &= \nabla_\theta D_{KL}^{\mu(\cdot;\theta)}(P_F(\bar{\tau}|s_m;\theta), P_{B,\rho}(\bar{\tau}|s_m)) + \nabla_\theta D_{KL}(\mu(s_m;\theta), \mu^*(s_m)) \\
&= \nabla_\theta D_{KL}^{\mu(\cdot;\theta)}(P_F(\bar{\tau}|s_m;\theta), \widetilde{P}_{B,\rho}(\bar{\tau}|s_m)), \\
\frac{1}{2}\nabla_\phi \mathcal{L}_{Sub-TB}(P_{B,\rho};\phi) &= \nabla_\phi D_{KL}^{\rho(\cdot;\phi)}(P_B(\bar{\tau}|s_n;\phi), P_{F,\mu}(\bar{\tau}|s_n)) + \nabla_\phi D_{KL}(\rho(s_m;\phi), \rho^*(s_m)) \\
&= \nabla_\phi D_{KL}^{\rho(\cdot;\phi)}(P_B(\bar{\tau}|s_n;\phi), \widetilde{P}_{F,\mu}(\bar{\tau}|s_n)),
\end{aligned} \tag{23}$$

where $\widetilde{P}_{F,\mu}(\bar{\tau}|s_n) := P_{F,\mu}(\bar{\tau})/\rho(s_n)$ and $\widetilde{P}_{B,\rho}(\bar{\tau}|s_m) := P_{B,\rho}(\bar{\tau})/\mu(s_m)$ are approximation to $P_{F,\mu}(\bar{\tau}|s_n)$ and $P_{B,\rho}(\bar{\tau}|s_m)$.

Proof. First of all, we split the parameters of the state flow estimator and forward transition probability and denote them as $F(\cdot;\theta_M)$ and $P_F(\cdot|\cdot;\theta_F)$ respectively. We further define $c(\bar{\tau}) = \left(\log \frac{P_F(\bar{\tau}|s_m)}{F(s_n)P_B(\bar{\tau}|s_n)}\right)$.

For the gradients w.r.t. θ_F:

$$\begin{aligned}
&\frac{1}{2}\nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta_F)] = \frac{1}{2}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_F}\left(c(\bar{\tau};\theta_F) + \log F(s_m)\right)^2\right] \\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[(c(\bar{\tau}) + \log F(s_m))\nabla_{\theta_F}\log P_F(\bar{\tau}|s_m;\theta_F)\right] + \underbrace{\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_F}(c(\bar{\tau};\theta_F) + \log F(s_m))\right]}_{=0 \text{ by Lemma A.1}} \\
&= \nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}\left[(c(\bar{\tau};\theta_F) + \log F(s_m))\right] + \underbrace{\nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}[\log \widehat{Z}_n - \log \widehat{Z}_m]}_{=0} \\
&= \nabla_{\theta_F} D_{KL}^{\mu}(P_F(\bar{\tau}|s_m;\theta_F), \widetilde{P}_{B,\rho}(\bar{\tau}|s_m)).
\end{aligned} \tag{24}$$

Besides,

$$\begin{aligned}
&\frac{1}{2}\nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta_F)] = \frac{1}{2}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_F}\left(c(\bar{\tau};\theta_F) + \log F(s_m)\right)^2\right] \\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[c(\bar{\tau})\nabla_{\theta_F}\log P_F(\bar{\tau}|s_m;\theta_F)\right] + \mathbb{E}_{\mu(s_m)}\Big[\log F(s_m)\underbrace{\mathbb{E}_{P_F(\bar{\tau}|s_m)}\left[\nabla_{\theta_F}\log P_F(\bar{\tau}|s_m;\theta_F)\right]}_{=0 \text{ by Lemma A.1}}\Big] \\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[c(\bar{\tau})\nabla_{\theta_F}\log P_F(\bar{\tau}|s_m;\theta_F)\right] + \underbrace{\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_F} c(\bar{\tau};\theta_F)\right]}_{=0} \\
&= \nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}\left[c(\bar{\tau};\theta_F)\right] + \mathbb{E}_{\mu(s_m)}\Big[\underbrace{\nabla_{\theta_F}\mathbb{E}_{P_F(\bar{\tau}|s_m;\theta_F)}[\log \mu^*(s_m)]}_{=\log\mu^*(s_m)\nabla_{\theta_F}1=0}\Big] + \underbrace{\nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}[\log \widehat{Z}_n]}_{=0} \\
&= \nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}\left[c(\bar{\tau};\theta_F)\right] + \nabla_{\theta_F}\mathbb{E}_{P_{F,\mu}(\bar{\tau};\theta_F)}[\log \mu^*(s_m) + \log \widehat{Z}_n] \\
&= \nabla_{\theta_F} D_{KL}^{\mu}(P_F(\bar{\tau}|s_m;\theta_F), P_{B,\rho}(\bar{\tau}|s_m))].
\end{aligned} \tag{25}$$

For the gradients w.r.t. θ_M, we have:

$$\begin{aligned}
&\frac{1}{2}\nabla_{\theta_M}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta_M)] = \frac{1}{2}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_M}\left(c(\bar{\tau}) + \log F(s_m;\theta_M)\right)^2\right] \\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[(c(\bar{\tau}) + \log F(s_m))\nabla_{\theta_M}\log F(s_m;\theta_M)\right] + \underbrace{\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[(\log \widehat{Z}_n - \log \widehat{Z}_m)\nabla_{\theta_M}\log F(s_m;\theta_M)]}_{=0 \text{ by Lemma A.1}} \\
&= \mathbb{E}_{\mu(s_m)}\left[D_{KL}(P_F(\bar{\tau}|s_m), \widetilde{P}_B(\bar{\tau}|s_m))\nabla_{\theta_M}\log\frac{F(s_m;\theta_M)}{\widehat{Z}_m}\right] \\
&= \nabla_{\theta_M} D_{KL}^{\mu(\cdot;\theta_M)}(P_F(\bar{\tau}|s_m), \widetilde{P}_B(\bar{\tau}|s_m)).
\end{aligned} \tag{26}$$

[5]Here, F_θ and F_ϕ actually share the same parameters and represent the same flow estimator F. Model parameters are duplicated just for the clarity of gradient equivalences. Therefore the true gradient of the state flow estimator F is $\nabla_\theta F_\theta + \nabla_\phi F_\phi$.

Besides,

$$\begin{aligned}
&\frac{1}{2}\nabla_{\theta_M}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta_M)] = \frac{1}{2}\mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[\nabla_{\theta_M}\left(c(\bar{\tau})+\log F(s_m;\theta_M)\right)^2\right]\\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[c(\bar{\tau})\nabla_{\theta_F}\log F(s_m;\theta_M)\right] + \mathbb{E}_{\mu(s_m)}\left[\log F(s_m)\nabla_{\theta_M}\log F(s_m;\theta_M)\right]\\
&= \mathbb{E}_{P_{F,\mu}(\bar{\tau})}\left[(c(\bar{\tau})+\log\mu^*(s_m))\nabla_{\theta_M}\log F(s_m;\theta_M)\right] + \mathbb{E}_{\mu(s_m)}\left[(\log F(s_m)-\log\mu^*(s_m))\nabla_{\theta_M}\log F(s_m;\theta_M)\right]\\
&\quad + \underbrace{\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[(\log\widehat{Z}_n-\log\widehat{Z}_m)\nabla_{\theta_M}\log F(s_m;\theta_M)]}_{=0 \text{ by Lemma A.1}}\\
&= \mathbb{E}_{\mu(s_m)}\left[D_{KL}(P_F(\bar{\tau}|s_m),P_{B,\rho}(\bar{\tau}|s_m))\nabla_{\theta_M}\log\frac{F(s_m;\theta_M)}{\widehat{Z}_m}\right]\\
&\quad + \mathbb{E}_{\mu(s_m)}\left[(\log\mu(s_m)-\log\mu^*(s_m))\nabla_{\theta_M}\log\frac{F(s_m;\theta_M)}{\widehat{Z}_m}\right] + \underbrace{\mathbb{E}_{\mu(s_m)}\left[\nabla_{\theta_M}\left(\log\mu(s_m;\theta_M)-\log\mu^*(s_m)\right)\right]}_{=0 \text{ By Lemma A.1}}\\
&= \nabla_{\theta_M}D_{KL}^{\mu(\cdot;\theta_M)}(P_F(\bar{\tau}|s_m),P_{B,\rho}(\bar{\tau}|s_m)) + \nabla_{\theta_M}D_{KL}(\mu(s_m;\theta_M),\mu^*(s_m)).
\end{aligned} \tag{27}$$

Combining equations (24) and (26), we obtain

$$\frac{1}{2}\nabla_\theta\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta)] = \nabla_\theta D_{KL}^{\mu(\cdot;\theta)}(P_F(\bar{\tau}|s_m;\theta),\widetilde{P}_B(\bar{\tau}|s_m)). \tag{28}$$

Combining equations (25) and (27), we obtain

$$\frac{1}{2}\nabla_\theta\mathbb{E}_{P_{F,\mu}(\bar{\tau})}[L_{Sub-TB}(\bar{\tau};\theta)] = \nabla_\theta\left\{D_{KL}^{\mu(\cdot;\theta)}(P_F(\bar{\tau}|s_m;\theta),P_B(\bar{\tau}|s_m)) + D_{KL}(\mu(s_m;\theta),\mu^*(s_m))\right\}. \tag{29}$$

Splitting ϕ into ϕ_B and ϕ_M and denoting $c(\bar{\tau}) = \log\frac{P_B(\bar{\tau}|s_n)}{F(s_m)P_F(\bar{\tau}|s_m)}$, the gradient derivation of ϕ follows the similar way as θ, and is omitted here. □

B. RL framework

B.1. Derivation of RL functions

Let's first consider the case of forward policies. For any $s\in\mathcal{S}_t$ and $a=(s\to s')\in\mathcal{A}(s)$ with $t\in\{0,\dots,T-1\}$, we define the $V_{F,t}$ and $Q_{F,t}$ as:

$$\begin{aligned}
V_{F,t}(s) &:= \mathbb{E}_{P_F(\tau_{>t}|s_t)}\left[\sum_{l=t}^{T-1}R_F(s_l,a_l)\middle|s_t=s\right]\\
&= R_F(s) + \mathbb{E}_{P_F(s_{t+1}|s_t)}\left[\mathbb{E}_{P_F(\tau_{>t+1}|s_{t+1})}\left[\sum_{l=t+1}^{T}R_F(s_l,a_l)\middle|s_{t+1}=s'\right]\middle|s_t=s\right]\\
&= R_F(s) + \mathbb{E}_{\pi_F(s,a)}[V_{F,t+1}(s')],\\
Q_{F,t}(s,a) &:= \mathbb{E}_{P_F(\tau_{>t+1}|s_t,a_t)}\left[\sum_{l=t}^{T-1}R_F(s_l,a_l)\middle|s_t=s,a_t=a\right]\\
&= R_F(s,a) + \mathbb{E}_{P_F(\tau_{>t+1}|s_{t+1})}\left[\sum_{l=t+1}^{T}R_F(s_l,a_l)\middle|s_{t+1}=s'\right]\\
&= R_F(s,a) + V_{F,t+1}(s'),
\end{aligned} \tag{30}$$

where $R_F(s) := \mathbb{E}_{\pi_F(s,a)}[R_F(s,a)]$, $V_{F,T}(\cdot):=0$, and $Q_{F,T}(\cdot,\cdot):=0$. Since $S_t\cap S_{t'}=\emptyset$ for any $t\neq t'$, we can read off the time indices (topological orders) from state values. Plus the fact that $R_F(s,a):=0$ for any $a\notin A(s)$, we are allowed to define two universal functions $V_F:\mathcal{S}\to\mathbb{R}$ and $Q_F:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ such that $V_F(s_t=s):=V_{F,t}(s)$ and $Q_F(s_t=s,a):=Q_{F,t}(s,a)$.

Remark B.1. While the transition environment $\mathcal{G}$ is exactly known, the state space $\mathcal{S}$ can be exponentially large, making the exact values of V and Q intractable. This fact, in spirit, corresponds to a regular RL problem where the exact values of V and Q are infeasible due to the unknown and uncertain transition environment $P(s'|s,a)$.

For backward policies, rewards are accumulated from $T-1$ to 1. Similarly, for $s' \in \mathcal{S}_t$ and $a = (s \to s') \in \dot{\mathcal{A}}(s')$,

$$V_{B,t}(s') := \mathbb{E}_{P_B(\tau_{<t}|s_t)}\left[\sum_{l=1}^{t} R_B(s_l,a_l)\middle| s_t = s'\right] = R_B(s') + \mathbb{E}_{\pi_B(s',a)}[V_{B,t-1}(s)],$$
$$Q_{B,t}(s',a) := \mathbb{E}_{P_B(\tau_{<t-1}|s_t,a_t)}\left[\sum_{l=1}^{t} R_B(s_l,a_l)\middle| s_t = s', a_t = a\right] = R_B(s',a) + V_{B,t-1}(s), \tag{31}$$

where $R_B(s') = \mathbb{E}_{\pi_B(s',a)}[R_B(s',a)]$, $V_{B,0}(\cdot) := 0$, and $Q_{B,0}(\cdot,\cdot) := 0$. For the same reason as forward policies, we can define universal functions $V_B : \mathcal{S} \to \mathbb{R}$ and $Q_B : \mathcal{S} \times \dot{\mathcal{A}} \to \mathbb{R}$ such that $V_B(s_t = s') := V_{B,t}(s')$ and $Q_B(s_t = s', a) := Q_{B,t}(s',a)$.

Based on the definitions above, the expected value functions are defined as:

$$J_F := \mathbb{E}_{\mu(s_0)}[V_F(s_0)], \quad J_B := \mathbb{E}_{\rho(x)}[V_B(x)]. \tag{32}$$

By definitions, $V_F(s_0) = \mathbb{E}_{P_F(\tau|s_0)}[\sum_{t=0}^{T-1} R_F(s_l,a_l)|s_0] = D_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0))$, so $J_F = D^{\mu}_{KL}(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0))$. Likewise, we can obtain $J_B = D^{\rho}_{KL}(P_B(\tau|x), \widetilde{P}_F(\tau|x))$. The advantages functions are defined as:

$$A_F(s,a) := Q_F(s,a) - V_F(s), \quad A_B(s',a) := Q_B(s',a) - V_B(s'). \tag{33}$$

We define the forward accumulated state distribution as $d_{F,\mu}(s) := \frac{1}{T}\sum_{t=0}^{T-1} P_{F,\mu}(s_t = s)$ such that for arbitrary function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\begin{aligned}
&\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} f(s_t,a_t)\right] = \sum_{t=0}^{T-1} \mathbb{E}_{P_{F,\mu}(s_t \to s_{t+1})}\left[f(s_t,a_t)\right] = \sum_{t=0}^{T-1} \mathbb{E}_{P_{F,\mu}(s_t),\pi_F(s_t,a_t)}\left[f(s_t,a_t)\right] \\
&= \sum_{t=0}^{T-1}\sum_{s}^{\mathcal{S}} P_{F,\mu}(s_t = s)\sum_{a}^{\mathcal{A}} \pi_F(s,a) f(s,a) = \sum_{s}^{\mathcal{S}}\sum_{a}^{\mathcal{A}}\left(\sum_{t=0}^{T-1} P_{F,\mu}(s_t = s)\right)\pi_F(s,a) f(s,a) \\
&= T\sum_{s}^{\mathcal{S}}\sum_{a}^{\mathcal{A}} d_{F,\mu}(s)\pi_F(s,a) f(s,a) = T\,\mathbb{E}_{d_{F,\mu}(s),\pi_F(s,a)}[f(s,a)],
\end{aligned} \tag{34}$$

where equation (34) holds in that $\forall s \notin \mathcal{S}_t : P(s_t = s) = 0$ and $\forall a \notin \mathcal{A}(s) : \pi_F(s,a) = 0$. By the fact that $S_t \cap S_{t'} = \emptyset$ for any $t \neq t'$ and any trajectory $\tau \in \mathcal{T}$ must pass some $s_t \in \mathcal{S}_t$ for $t \in \{0,\dots,T-1\}$, $P_{F,\mu}(s_t)$ is a valid distribution over $\mathcal{S}_t$ and $\sum_{s_t} P_{F,\mu}(s_t) = 1$. Accordingly, $d_{F,\mu}(s)$ is a valid distribution over $\mathcal{S}$ and $T\,d_{F,\mu}(s_t) = P_{F,\mu}(s_t)$. Analogously, we can define $d_{B,\rho}(s') := \frac{1}{T-1}\sum_{t=1}^{T-1} P_{B,\rho}(s_t = s')$ such that for arbitrary function $f : \mathcal{S} \times \dot{\mathcal{A}} \to \mathbb{R}$,

$$\mathbb{E}_{P_{B,\rho}(\tau)}\left[\sum_{t=1}^{T-1} f(s_t,a_t)\right] = (T-1)\,\mathbb{E}_{d_{B,\rho}(s'),\pi_B(s',a)}[f(s',a)]. \tag{35}$$

B.2. DAGs as transition environments

Theorem B.2. *(Golpar Raboky & Eftekhari, 2019) Let $P \in \mathbb{R}^{N\times N}$ be a non-negative matrix. The following statements are equivalent:*

1. *P is nilpolent;*
2. *P^N =0;*
3. *The directed graph $\mathcal{G}(\mathcal{S},\mathcal{A})$ associated with P is a DAG;*

4. There exists a permutation matrix U such that $U^T P U$ is a strictly triangular matrix.

where $\mathcal{S} = \{s^0, \dots, s^{N-1}\}$ and $\mathcal{A} = \{(s^i \to s^j) | P_{i,j} \neq 0\}$ are node and edge sets.

Lemma B.3. *For any DAG graph $\mathcal{G}(\mathcal{S}, \mathcal{A})$ associated with $P \in R^{N \times N}$ with $T + 1 (\leq N)$ different topological node orders indexed by integers $[0, T]$,*

$$\forall t > T, \quad P^t = \mathbf{0}. \tag{36}$$

Proof. We prove the result by contradiction. Assuming $P^t (t > T)$ is not zero, then $\exists\, i \neq j$:

$$[P^t]_{i,j} = \sum_{k_{1:t-1}} P_{i,k_1} P_{k_1,k_2} \dots P_{k_{t-1},j} > 0. \tag{37}$$

By the nature of DAGs, $\forall (s' \to s) \in \mathcal{A} : s' \prec s$. Then the above expression is equal to:

$$\begin{aligned}
[P^t]_{i,j} &= \sum_{k_1 : s^i \prec s^{k_1}} P_{i,k_1} \left(\sum_{k_2 : s^{k_1} \prec s^{k_2}} P_{k_1,k_2} \dots \left(\sum_{k_{t-1} : s^{k_{t-2}} \prec s^{k_{t-1}}} P_{k_{t-2},k_{t-1}} P_{k_{t-1},j} \right) \right) \\
&= \sum_{k_{1:t-1} : (s^i \prec s^{k_1} \prec \dots \prec s^{k_{t-1}} \prec s^j)} P_{i,k_1} P_{k_1,k_2} \dots P_{k_{t-1},j} \\
&> 0.
\end{aligned} \tag{38}$$

This means that there at least exists a trajectory $(s^i \prec s^{k_1} \prec \dots \prec s^{k_{t-1}} \prec s^j)$ with non-zero probability. However, there are $t + 1$ distinct node orders in the path, which contradicts the assumption that there are $T + 1$ different node orders. □

Let's return to the graded DAG, $\mathcal{G}(\mathcal{S}, \mathcal{A})$ in GFlowNets. For the easiness of analysis, we restrict forward and backward policies and initial distribution to be tabular forms, $P_F \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, $\mu \in \mathbb{R}^{|\mathcal{S}|}$, $P_B \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, and $\rho \in \mathbb{R}^{|\mathcal{S}|}$ such that $P_F(s^j | s^i) = [P_F]_{j,i}$ and $P_B(s^i | s^j) = [P_B]_{i,j}$. Besides, we split initial distribution vectors by $\mu = (\bar{\mu}; 0) \in \mathbb{R}^{|\mathcal{S}|}$ and $\rho = (0; \bar{\rho}) \in \mathbb{R}^{|\mathcal{S}|}$, where $\bar{\mu}$ and $\bar{\rho}$ denote the probabilities of states except s^f and s^0 respectively. We denote the graph equipped with a self-loop over s^f as $\mathcal{G}_F(\mathcal{S}, \mathcal{A} \cup \{(s^f \to s^f)\})$, and the reverse graph equipped with a self-loop over s^0 as $\mathcal{G}_B(\mathcal{S}, \dot{\mathcal{A}} \cup \{(s^0 \to s^0)\})$. Accordingly, we enhance P_F and P_B by defining $P_F(s^f | s^f) := 1$ and $P_B(s^0 | s^0) := 1$. $(\mathcal{G}_F, P_F)$ specifies an absorbing Markov Chains: s^f is the only absorbing state as only the self-loop is allowed once entering s^f; the sub-graph over $S \setminus \{s^f\}$, denoted as $\bar{\mathcal{G}}_F$ is still a DAG, so any state $s \in S \setminus \{s^f\}$ is transient as it can be visited at most one time. Similarly, $(\mathcal{G}_B, P_B)$ specifies another absorbing Markov Chain with absorbing state s^0 and a DAG over $S \setminus \{s^0\}$, denoted as $\bar{\mathcal{G}}_B$. For graph $\mathcal{G}_F$ and $\mathcal{G}_B$, their transition matrices P_F and P_B can be decomposed into:

$$P_F = \begin{pmatrix} \bar{P}_F & \mathbf{0} \\ r_F & 1 \end{pmatrix}, P_B = \begin{pmatrix} 1 & r_B \\ \mathbf{0} & \bar{P}_B \end{pmatrix}. \tag{39}$$

In the equations above, $r_F \in \mathbb{R}^{1 \times (|\mathcal{S}|-1)}$ and $r_B \in \mathbb{R}^{1 \times (|\mathcal{S}|-1)}$ denote the probabilities of $(s \to s^f)$ for any $s \in \mathcal{S} \setminus \{s^f\}$ and $(s^0 \leftarrow s)$ for any $s \in \mathcal{S} \setminus \{s^0\}$ respectively; $\bar{P}_F \in \mathbb{R}^{(|\mathcal{S}|-1) \times (|\mathcal{S}|-1)}$ and $\bar{P}_B \in \mathbb{R}^{(|\mathcal{S}|-1) \times (|\mathcal{S}|-1)}$ denote probability of $(s \to s')$ for any $s, s' \in \mathcal{S} \setminus \{s^f\}$ and $(s \leftarrow s')$ for any $s, s' \in \mathcal{S} \setminus \{s^0\}$, that is, the transition matrices over $\bar{\mathcal{G}}_F$ and $\bar{\mathcal{G}}_B$ respectively.

Lemma B.4. *For $(\mathcal{G}, P_F, \mu)$ and $(\mathcal{G}_B, P_B, \rho)$, $d_{F,\mu} \in \mathbb{R}^{|S|}$ and $d_{B,\rho} \in \mathbb{R}^{|S|}$, can be written in the following forms:*

$$\begin{aligned}
d_{F,\mu} &= \begin{pmatrix} \bar{d}_{F,\mu} \\ 0 \end{pmatrix}, \quad \bar{d}_{F,\mu} = \frac{1}{T} (I - \bar{P}_F)^{-1} \bar{\mu}, \\
d_{B,\rho} &= \begin{pmatrix} 0 \\ \bar{d}_{B,\rho} \end{pmatrix}, \quad \bar{d}_{B,\rho} = \frac{1}{T-1} (I - \bar{P}_B)^{-1} \bar{\rho}.
\end{aligned} \tag{40}$$

Proof. We first prove the result for the forward case. By the nature of Markov Chains, $P_{F,\mu}(s_t = s^i) = [(P_F)^t \mu]_i$, and $d_{F,\mu} = \frac{1}{T} \sum_{t=0}^{T-1} (P_F)^t \mu$. Then, it can be easily verified (Grinstead & Snell, 2006) that:

$$(P_F)^t = \begin{pmatrix} (\bar{P}_F)^t & \mathbf{0} \\ * & 1 \end{pmatrix}, \tag{41}$$

where the explicit expression of the upper right corner is omitted. By Theorem B.2, $\bar{P}_F$ is a nilpotent matrix and by Lemma B.3, $\sum_{t=0}^{T-1}(\bar{P}_F)^t = \sum_{t=0}^{\infty}(\bar{P}_F)^t = (I-\bar{P}_F)^{-1}$, where the first equality follows from the fact that $\bar{\mathcal{G}}_F$ has T topological orders, and the second equality is by the fact that $(I-\bar{P}_F)\sum_{t=0}^{\infty}(\bar{P}_F)^t = \sum_{t=0}^{\infty}(\bar{P}_F)^t - \sum_{t=1}^{\infty}(\bar{P}_F)^t = I$. Therefore,

$$d_{F,\mu} = \frac{1}{T}\begin{pmatrix}\sum_{t=0}^{T-1}\bar{P}_F^t & \mathbf{0}\\ * & 1\end{pmatrix}\mu = \frac{1}{T}\begin{pmatrix}(I-\bar{P}_F)^{-1}\bar{\mu}\\ *\bar{\mu}\end{pmatrix}. \tag{42}$$

By Theorem 11.4 in Grinstead & Snell (2006), $[(I-\bar{P}_F)^{-1}]_{j,i}$ is the expected number of times the chain is in state s^j, starting from s^i, before being absorbed in s^f. And $[(I-\bar{P}_F)^{-1}\bar{\mu}]_j$ is the expected number of times the chain is in state s^j before being absorbed. Since $\forall s \notin \mathcal{S}_0 : \mu(s) = 0$ and $\bar{\mathcal{G}}_F$ is graded, any forward trajectory over sub-graph $\bar{\mathcal{G}}_F$ must start from $s \in \mathcal{S}_0$ and end in $s \in \mathcal{S}_{T-1}$, meaning $\sum_j[(I-\bar{P}_F)^{-1}\bar{\mu}]_j = T$. Thus, $\frac{1}{T}[(I-\bar{P}_F)^{-1}\bar{\mu}]_j$ denotes the fraction of staying in transient state s^j before being absorbed, that is, the probability observing state s^j within T time steps. By the same reasoning, we can conclude that $*\bar{\mu} = 0$ as s^f can not be reached within T time steps.

For backward case, any backward trajectory over sub-graph $\bar{\mathcal{G}}_B$ must start from $s \in \mathcal{S}_{T-1}$ and end in $s \in \mathcal{S}_1$ as $\forall s \notin \mathcal{S}_{T-1} : \rho(s) = 0$ and $\bar{\mathcal{G}}_B$ is graded. Then, a proof procedure for the desired result can be derived similarly, so it is omitted. □

Lemma B.5. *For two forward policy, π_F and π'_F, and two backward policy, π_B and π'_B, we have:*

$$\begin{aligned} D_{TV}(d'_{F,\mu}(\cdot), d_{F,\mu}(\cdot)) &\le D_{TV}^{d'_{F,\mu}}(\pi'_F(s,\cdot), \pi_F(s,\cdot)),\\ D_{TV}(d'_{B,\rho}(\cdot), d_{B,\rho}(\cdot)) &\le D_{TV}^{d'_{B,\rho}}(\pi'_B(s,\cdot), \pi_B(s,\cdot)), \end{aligned} \tag{43}$$

where for three arbitrary distributions p,q and u, $D_{TV}(p(\cdot), q(\cdot)) := \frac{1}{2}\|p(\cdot) - q(\cdot)\|_1$ and $D_{TV}^u(p(\cdot|s), q(\cdot|s)) := \frac{1}{2}\mathbb{E}_{u(s)}\left[\|p(\cdot|s) - q(\cdot|s)\|_1\right]$.

Proof. The proof procedure follows that of Lemma 3 in Achiam et al. (2017). For two forward policy π_F and π'_F, let $\bar{N}_F := (I-\bar{P}_F)^{-1}$ and $\bar{N}'_F := (I-\bar{P}'_F)^{-1}$. Then,

$$\Delta := \bar{P}_F - \bar{P}'_F = (\bar{N}'_F)^{-1} - \bar{N}_F^{-1}, \tag{44}$$

and

$$\bar{N}_F - \bar{N}'_F = \bar{N}_F \Delta \bar{N}'_F. \tag{45}$$

Then,

$$\begin{aligned} \left\|d_{F,\mu} - d'_{F,\mu}\right\|_1 &= \left\|\bar{d}_{F,\mu} - \bar{d}'_{F,\mu}\right\|_1\\ &= \frac{1}{T}\left\|(\bar{N}_F - \bar{N}'_F)\bar{\mu}\right\|_1 = \frac{1}{T}\left\|\bar{N}_F\Delta\bar{d}'_{F,\mu}\right\|_1\\ &\le \frac{1}{T}\left\|\bar{N}_F\right\|_1\left\|\Delta\bar{d}'_{F,\mu}\right\|_1 \le \frac{1}{T}\left(\sum_{t=0}^{T-1}\left\|P_F^t\right\|_1\right)\left\|\Delta\bar{d}'_{F,\mu}\right\|_1\\ &\le \left\|\Delta\bar{d}'_{F,\mu}\right\|_1 = \left\|(\bar{P}_F - \bar{P}'_F)\bar{d}'_{F,\mu}\right\|_1. \end{aligned} \tag{46}$$

Therefore, we have

$$\begin{aligned} \left\|d'_{F,\mu} - d_{F,\mu}\right\|_1 &\le \left\|(\bar{P}'_F - \bar{P}_F)\bar{d}'_{F,\mu}\right\|_1\\ &\le \left\|(\bar{P}'_F - \bar{P}_F)\bar{d}'_{F,\mu}\right\|_1 + \left|(r'_F - r_F)\bar{d}'_{F,\mu}\right| = \left\|(P'_F - P_F)d'_{F,\mu}\right\|_1\\ &= \sum_s\left|\sum_{s'}\left(P'_F(s'|s) - P_F(s'|s)\right)d'_{F,\mu}(s)\right|\\ &\le \sum_{s,s'}\left|P'_F(s'|s) - P_F(s'|s)\right|d'_{F,\mu}(s)\\ &= \sum_{s,a}\left|\pi'_F(s,a) - \pi_F(s,a)\right|d'_{F,\mu}(s) = \mathbb{E}_{d'_{F,\mu}(s)}\left[\|\pi'_F(s,\cdot) - \pi_F(s,\cdot)\|_1\right]. \end{aligned} \tag{47}$$

The result for backward policies can be derived analogously and is omitted here. □

B.3. Derivation of gradients

Proposition B.6. *The gradients of $J_F(\theta)$ and $J_B(\phi)$ w.r.t. θ and ϕ can be written as:*

$$\begin{aligned}
\nabla_\theta J_F(\theta) &= T\,\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[Q_F(s,a)\nabla_\theta \log \pi_F(s,a;\theta)\right] + \mathbb{E}_{\mu(s_0)}[V_F(s_0)\nabla_\theta \log \mu(s_0;\theta)] \\
&= T\,\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[A_F(s,a)\nabla_\theta \log \pi_F(s,a;\theta)\right] + \mathbb{E}_{\mu(s_0)}[V_F(s_0)\nabla_\theta \log \mu(s_0;\theta)], \\
\nabla_\phi J_B(\phi) &= (T-1)\,\mathbb{E}_{d_{B,\rho}(s)\pi_B(s,a)}\left[Q_B(s,a)\nabla_\phi \log \pi_B(s,a;\phi)\right] \\
&= (T-1)\,\mathbb{E}_{d_{B,\rho}(s)\pi_B(s,a)}\left[A_B(s,a)\nabla_\phi \log \pi_B(s,a;\phi)\right].
\end{aligned} \tag{48}$$

Remark B.7. This result implies that an estimated value function, which may differ from the exact one, does not lead to biased gradient estimation.

Proof.

$$\begin{aligned}
\nabla_\theta J_F(\theta) &= \mathbb{E}_{\mu(s_0)}[V_F(s_0)\nabla_\theta \log \mu(s_0;\theta)] + \underbrace{\mathbb{E}_{\mu(s_0)}[\nabla_\theta V_F(s_0;\theta)]}_{(1)} \\
&\overset{(1)}{=} \mathbb{E}_{P_{F,\mu}(s_0)}\left[\nabla_\theta \mathbb{E}_{\pi_F(s_0,a_0;\theta)}[Q_F(s_0,a_0;\theta)]\right] \\
&= \mathbb{E}_{P_{F,\mu}(s_0)}\left[\mathbb{E}_{\pi_F(s_0,a_0)}[Q_F(s_0,a_0)\nabla_\theta \log \pi_F(s_0,a_0;\theta) + \nabla_\theta Q_F(s_0,a_0;\theta)]\right] \\
&= \mathbb{E}_{P_{F,\mu}(s_0\to s_1)}\left[Q_F(s_0,a_0)\nabla_\theta \log \pi_F(s_0,a_0;\theta)\right] + \underbrace{\mathbb{E}_{P_{F,\mu}(s_0\to s_1)}\left[\nabla_\theta R_F(s_0,a_0;\theta) + \nabla_\theta V_F(s_1;\theta)\right]}_{(2)} \\
&\overset{(2)}{=} \underbrace{\mathbb{E}_{P_{F,\mu}(s_0\to s_1)}\left[\nabla_\theta \log \frac{\pi_F(s_0,a_0;\theta)}{\pi_B(s_1,a_0)}\right]}_{(3)} + \mathbb{E}_{P_{F,\mu}(s_1)}\left[\nabla_\theta V_F(s_1;\theta)\right] \\
&\overset{(3)}{=} \mathbb{E}_{P_{F,\mu}(s_0)}\big[\underbrace{\mathbb{E}_{\pi_F(s_0,a_0)}[1\cdot\nabla_\theta \log \pi_F(s_0,a_0;\theta)]}_{=0 \text{ by Lemma A.1}}\big].
\end{aligned} \tag{49}$$

Therefore,

$$\mathbb{E}_{P_{F,\mu}(s_0)}[\nabla_\theta V_F(s_0;\theta)] \overset{(1)}{=} \mathbb{E}_{P_{F,\mu}(s_0\to s_1)}\left[Q_F(s_0,a_0)\nabla_\theta \log \pi_F(s_0,a_0;\theta)\right] + \mathbb{E}_{P_{F,\mu}(s_1)}\left[\nabla_\theta V_F(s_1;\theta)\right]. \tag{50}$$

Keep doing the process, we have

$$\mathbb{E}_{P_{F,\mu}(s_t)}[\nabla_\theta V_F(s_t;\theta)] = \mathbb{E}_{P_{F,\mu}(s_t\to s_{t+1})}\left[Q_F(s_t,a_t)\nabla_\theta \log \pi_F(s_t,a_t;\theta)\right] + \mathbb{E}_{P_{F,\mu}(s_{t+1})}\big[\nabla_\theta \underbrace{V_F(s_{t+1};\theta)}_{V_F(s_T)=0}\big]. \tag{51}$$

Then,

$$\overset{(1)}{=} \mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} Q_F(s_t,a_t)\nabla_\theta \log \pi_F(s_t,a_t;\theta)\right] = \mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[Q_F(s,a)\nabla_\theta \log \pi_F(s,a;\theta)\right]. \tag{52}$$

Besides,

$$\begin{aligned}
&\overset{(1)}{=} \mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[Q_F(s,a)\nabla_\theta \log \pi_F(s,a;\theta)\right] - \mathbb{E}_{d_{F,\mu}(s)}\big[V_F(s)\underbrace{\mathbb{E}_{\pi_F(s,a)}[\nabla_\theta \log \pi_F(s,a;\theta)]}_{=0}\big] \\
&= \mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[A_F(s,a)\nabla_\theta \log \pi_F(s,a;\theta)\right].
\end{aligned} \tag{53}$$

The derivation of $\nabla_\phi J_B(\phi)$ follows the similar way to $\nabla_\theta J_F(\theta)$, and is omitted here. □

B.4. Connection of policy-based training to TB-based training

The gradient of the TB objective w.r.t. θ_F can be written as:

$$\frac{1}{2}\nabla_{\theta_F}\mathcal{L}_{TB}(P_{F,\mu};\theta_F)=\sum_{t=1}^{T}\mathbb{E}_{P_{F,\mu}(\tau)}\left[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{t=1}^{T}\log\frac{P_F(s_t|s_{t-1})}{\widetilde{P}_B(s_{t-1}|s_t)}\right)\right]. \tag{54}$$

In equation (54), each term for $t>1$ can be expanded as:

$$\begin{aligned}
&\mathbb{E}_{P_{F,\mu}(\tau_{\geq t-1})}\left[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{l=t}^{T}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}\right)\right]\\
&+\mathbb{E}_{P_{F,\mu}(\tau_{\leq t})}\left[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{l=1}^{t-1}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}\right)\right]\\
=&\mathbb{E}_{P_{F,\mu}(\tau_{\geq t-1})}\left[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{l=t}^{T}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}\right)\right]\\
&+\mathbb{E}_{P_{F,\mu}(\tau_{\leq t-1})}\left[\left(\sum_{l=1}^{t-1}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}\right)\underbrace{\mathbb{E}_{P_{F,\mu}(s_t|s_{t-1})}[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)]}_{=0\text{ by Lemma A.1}}\right].
\end{aligned} \tag{55}$$

Thus,

$$\begin{aligned}
\frac{1}{2}\nabla_{\theta_F}\mathcal{L}_{TB}(P_{F,\mu};\theta_F)&=\sum_{t=1}^{T}\mathbb{E}_{P_{F,\mu}(\tau)}\left[\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{l=t}^{T}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}\right)\right]\\
&\quad-C\sum_{t=0}^{T-1}\underbrace{\mathbb{E}_{P_{F,\mu}(\tau)}\left[\nabla_{\theta_F}\log P_F(s_t,a_t;\theta_F)\right]}_{=0\text{ by Lemma A.1}}\\
&=\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=1}^{T}\nabla_{\theta_F}\log P_F(s_t|s_{t-1};\theta_F)\left(\sum_{l=t}^{T}\log\frac{P_F(s_l|s_{l-1})}{\widetilde{P}_B(s_{l-1}|s_l)}-C\right)\right],
\end{aligned} \tag{56}$$

where C is an added baseline and constant w.r.t. θ_F for variance reduction during gradient estimation. As shown in Appendix B.3, the gradient of J_F w.r.t. θ_F can be written as:

$$\begin{aligned}
\nabla_{\theta_F}J_F(\theta_F)&=T\,\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[A_F(s,a)\nabla_{\theta_F}\log\pi_F(s,a;\theta_F)\right]\\
&=\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1}\nabla_{\theta_F}\log\pi_F(s_t,a_t;\theta_F)Q_F(s_t,a_t)\right]\\
&=\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1}\nabla_{\theta_F}\log\pi_F(s_t,a_t;\theta_F)\mathbb{E}_{P_{F,\mu}(\tau_{>t+1}|s_t,a_t)}\left[\sum_{l=t}^{T-1}R_F(s_l,a_l)\middle|s_t,a_t\right]\right]\\
&=\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1}\nabla_{\theta_F}\log\pi_F(s_t,a_t;\theta_F)\left(\sum_{l=t}^{T-1}R_F(s_l,a_l)\right)\right]\\
&=\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1}\nabla_{\theta_F}\log\pi_F(s_t,a_t;\theta_F)\left(\sum_{l=t}^{T-1}R_F(s_l,a_l)-C\right)\right].
\end{aligned} \tag{57}$$

This result implies that: when we update the forward policy by the estimation of $\nabla_{\theta_F}\mathcal{L}_{TB}(\theta_F)$ based on a batch of sampled trajectories, we approximate $Q_F(s_t,a_t)$ empirically by $\widehat{Q}_F(s_t,a_t)=\sum_{l=t}^{T-1}R_F(s_l,a_l)$ for each sample, and can further reduce the estimation variance by some unbiased constant baseline C. By comparison, the RL formulation generalizes the constant to an unbiased functional baseline $\widetilde{V}_F(s;\eta)$, which is the approximation of exact $V_F(s)$. This enables to approximate

$Q_F(s_t, a_t)$ and $A_F(s_t, a_t)$ functionally by $\widehat{Q}_F(s_t, a_t) = R(s_t, a_t) + \widetilde{V}_F(s_{t+1})$ and $\widehat{A}_F(s_t, a_t) = \widehat{Q}_F(s_t, a_t) - \widetilde{V}_F(s_t)$. Here, $\widehat{A}_F(s_t, a_t)$ can further be generalized to $\sum_{l=t}^{T-1} \lambda^{l-t} \left(\widehat{Q}_F(s_l, a_l) - \widetilde{V}_F(s_l)\right)$, allowing flexible bias-variance trade-off for gradient estimation (Appendix B.6).

B.5. Connection between policy-based training and Soft-Q-learning

In the following text, we discuss the relationship between our policy-based method and Soft-Q-learning (Haarnoja et al., 2018), one of the most representative Maximum-Entropy (MaxEnt) RL methods.

Firstly, we introduce their connection when the total flow estimator $\log Z$ is **fixed**. We can expand $-J_F$ as:

$$\begin{aligned} -J_F &= T\,\mathbb{E}_{d_{F,\mu}(s),\pi_F(s,a)}\left[\log \pi_B(s', a) - \log \pi_F(s, a)\right] \\ &= T\,\mathbb{E}_{d_{F,\mu}(s),\pi_F(s,a)}\left[\log \pi_B(s', a)) + \mathbb{E}_{\pi_F(s,a)}[-\log \pi_F(s, a)]\right] \\ &= T\,\mathbb{E}_{d_{F,\mu}(s),\pi_F(s,a)}\left[\log \pi_B(s', a)) + \mathcal{H}(\pi_F(s, \cdot))\right], \end{aligned} \tag{58}$$

where $a = (s \to s')$, and $\mathcal{H}$ denotes the entropy of a distribution. The equation above implies that fixing the total flow estimator $\log Z$, maximizing $-J_F$ w.r.t. π_F can be interpreted as a MaxEnt RL problem, where $\log \pi_B(s', a)$ is the **static** reward w.r.t. $\pi_F(s, a)$. We define $Q_F^S(s, a) := \mathbb{E}_{P_F(\tau_{>t+1}|s_t,a_t)}\Big[\pi_B(s_{t+1}, a_t) + \sum_{l=t+1}^{T-1} \pi_B(s_{l+1}, a_l) + \mathcal{H}(\pi_F(s_l, \cdot))|s_t = s, a_t = a\Big] = -Q_F(s, a) + \log \pi_F(s, a)$ and $V_F^S(s, a) := \mathbb{E}_{P_F(\tau_{>t}|s_t)}\Big[\sum_{l=t}^{T-1} \pi_B(s_{l+1}, a_l) + \mathcal{H}(\pi_F(s_l, \cdot))|s_t = s\Big] = -V_F(s, a)$, which implies that $Q_F^S(s, a) = \log \pi_B(s', a) + V_F^S(s')$. We use a parameterized function $\widetilde{Q}_F^S$ as the estimator of Q_F^S, and define $\widetilde{V}_F^S(s) := \log \sum_a \exp\{\widetilde{Q}_F^S(s, a)\}$ as the estimator of V_F^S. Then, we define $\pi_F(s, a; \theta) := \frac{\exp \widetilde{Q}_F^S(s,a;\theta)}{\exp \widetilde{V}_F^S(s;\theta)}$, which implies that:

$$\widetilde{Q}_F^S(s, a) = \widetilde{V}_F^S(s) + \log \pi_F(s, a). \tag{59}$$

In Soft-Q-learning, we define the target function as:

$$\widehat{Q}_F^S(s, a) := \log \pi_B(s', a) + \widetilde{V}_F^S(s'). \tag{60}$$

Then, π_F is updated by the gradient of the following objective:

$$\frac{T}{2}\mathbb{E}_{d_{\mathcal{D}}(s),\pi_{\mathcal{D}}(s,a)}\left[\left(\widetilde{Q}_F^S(s, a; \theta) - \widehat{Q}_F^S(s, a)\right)^2\right]. \tag{61}$$

It has been shown that the policy gradients for static rewards plus the gradients of the policy entropy (or KL divergence from some reference policy) are equivalent to the gradients of the corresponding soft Q estimator (Schulman et al., 2017a). Likewise, we demonstrate our policy-based method with $\lambda = 0$ is equivalent to Soft-Q-learning with $\pi_{\mathcal{D}}$ equal to π_F, without any adaption. Noting Soft-Q-learning is off-policy, the gradients of equation (61) w.r.t. θ can be written as:

$$\begin{aligned} &\frac{T}{2}\nabla_\theta \mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[\left(\widetilde{Q}_F^S(s, a; \theta) - (\log P_B(s', a) + \widetilde{V}_F^S(s'))\right)^2\right] \\ &= T\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[\nabla_\theta \widetilde{Q}_F^S(s, a; \theta)\left(\widetilde{Q}_F^S(s, a; \theta) - \log P_B(s', a) - \widetilde{V}_F^S(s')\right)\right] \\ &= T\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[\nabla_\theta \left(\log \pi_F(s, a) + \widetilde{V}_F^S(s)\right)\left(\widetilde{V}_F^S(s) + R_F(s, a) - \widetilde{V}_F^S(s')\right)\right] \\ &= T\mathbb{E}_{d_{F,\mu}(s)\pi_F(s,a)}\left[\nabla_\theta \left(\log \pi_F(s, a) + \widetilde{V}_F^S(s)\right)\hat{\delta}_F(s, a)\right] \\ &= \mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_F(s_t, a_t; \theta)\hat{\delta}_F(s_t, a_t)\right] + \mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \widetilde{V}_F^S(s_t; \theta)\hat{\delta}_F(s_t, a_t)\right], \end{aligned} \tag{62}$$

where $\hat{\delta}_F(s, a) := -\widetilde{V}_F(s) + R_F(s, a) + \widetilde{V}_F(s')$, and $\widetilde{V}_F := -\widetilde{V}_F^S$. Compared to formula (65) with $\lambda = 0$, a clear equivalence can be established.

We further connect the Soft-Q-learning objective (61) to the Flow Matching (FM) objective (Bengio et al., 2021) and explain the role that $\log Z$ plays during training. When $\widetilde{Q}_F^S$ achieves the optimal point, we have $\widetilde{Q}_F^S = \widehat{Q}_F^S$. Consequently, for the desired flow F, $\widetilde{V}_F^S(s^f) = 0$ by definition, $\widetilde{Q}_F^S(x, a_{T-1}) = \log \pi_B(x, a_{T-1}) + 0 = \log \frac{F(x \to s^f)}{Z}$, and $\widetilde{V}_F^S(x) = \log \sum_a \exp Q_F^S(x, a) = \log \frac{F(x)}{Z}$ $(= \log \frac{R(x)}{Z})$. Accordingly, $\widetilde{Q}_F^S(s_{T-2}, a_{T-2}) = \widetilde{V}_F^S(x) + \log \pi_B(x, a_{T-2}) = \log \frac{F(x)}{Z} + \log \frac{F(s_{T-2} \to x)}{F(x)} = \log \frac{F(s_{T-2} \to x)}{Z}$, and $\widetilde{V}_F^S(s_{T-2}) = \log \sum_a \exp Q_F^S(s_{T-2}, a) = \log \frac{F(s_{T-2})}{Z}$. Continuing this process, it can be verified that $\widetilde{Q}_F^S(s, a) = \log \frac{F(s \to s')}{Z}$ and $\widetilde{V}_F^S(s) = \log \frac{F(s)}{Z}$ when $\widetilde{Q}^S$ achieve the optimum. Based on the above optimum condition of $\widetilde{Q}^S(s, a)$ and the fact that $\widetilde{Q}^S(s, a) \in \mathbb{R}$ is a parametrized function with no assumption over its output form during training, we can safely substitute it by $F^{\log}(s \to s') := \widetilde{Q}^S(s, a) + \log Z \in \mathbb{R}$, where $F^{\log}(s \to s')$ is the estimator for the logarithm of the desired edge flow, $\log F(s \to s')$, and no assumption over its output form during training is made as well. Then, objective (61) can be equivalently rewritten as:

$$\mathbb{E}_{P_{\mathcal{D}}(\tau)}\left[\sum_{t=1}^{T-1}\left(F^{\log}(s_{t-1} \to s_t) - \log\left(P_B(s_{t-1}|s_t)\sum_{s_{t+1}} \exp F^{\log}(s_t \to s_{t+1})\right)\right)^2\right], \tag{63}$$

This objective is similar to the FM objective, which can be written as:

$$\mathbb{E}_{P_{\mathcal{D}}(\tau)}\left[\sum_{t=1}^{T-1}\left(\log\left(\sum_{s_{t-1}} \exp F^{\log}(s_{t-1} \to s_t)\right) - \log\left(\sum_{s_t} \exp F^{\log}(s_t \to s_{t+1})\right)\right)^2\right]. \tag{64}$$

The reason is that the optimal solution of the objective (63) satisfies $\exp F^{\log}(s_{t-1} \to s_t) = P_B(s_{t-1}|s_t)\sum_{s_{t+1}} \exp F^{\log}(s_t \to s_{t+1})$. Taking summation over s_{t-1} of this equation, we have $\sum_{s_{t-1}} \exp F^{\log}(s_{t-1} \to s_t) = \sum_{s_{t+1}} \exp F^{\log}(s_t \to s_{t+1})$, the optimal solution of the objective (64). We can see that $\log Z$ serves as a baseline for modeling $\log F$. Without $\log Z$, we need to approximate $\log F$ by $F^{\log}$ directly. This often leads to numerical issues. For example, let's suppose a small perturbation of $\log F$, denoted as ϵ. Then the flow difference is $\exp(\log F + \epsilon) - F = (\exp \epsilon - 1)F$. As values of F can be exponentially large, especially for nodes near the root, the flow difference can be large even if ϵ is small, making approximation to F by $\exp F^{\log}$ very difficult. By contrast, TB-based methods and our policy-based method allow the updating of $\log Z$ to dynamically scale down the value of $\widetilde{Q}$ during training. This also complements the claims by Malkin et al. (2022a), who show that the TB-based methods is more efficient than flow-matching and DB-based methods.

B.6. Model parameter updating rules

In the following context, we will explain the updating rules for P_F and μ within the vanilla policy-based method, also called the Actor-Critic method, and the TRPO method. The updating rules for P_B follow those of P_F analogously.

Actor-Critic First of all, we split the parameter θ into θ_F and θ_Z corresponding to π_F and μ. Since computing the exact V_F is usually intractable, we use $\widetilde{V}_F$ parametrized by η as the functional approximation. Given a batch of trajectories samples, we compute the sampling averaging approximation of the following gradient estimators to update θ_F, θ_Z and η as proposed by Schulman et al. (2016) and Tsitsiklis & Van Roy (1996):

$$\begin{aligned}
&\mathbb{E}_{P_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} \widehat{A}_F^{\lambda}(s_t, a_t)\nabla_{\theta_F} \log \pi_F(s_t, a_t; \theta_F)\right] + \mathbb{E}_{\mu(s_0)}\left[\widehat{V}_F^{\lambda}(s_0)\nabla_{\theta_Z} \log \mu(s_0; \theta_Z)\right], \\
&\mathbb{E}_{P_{F,\mu}(s_t)}\left[\sum_{t=0}^{T-1} \nabla_{\eta}(\widehat{V}_F^{\lambda}(s_t) - \widetilde{V}_F(s_t; \eta))^2\right],
\end{aligned} \tag{65}$$

where $\lambda \in [0, 1]$,

$$\begin{aligned}
&\widehat{A}_F^{\lambda}(s_t, a_t) := \sum_{l=t}^{T-1} \lambda^{l-t}\hat{\delta}_F(s_l, a_l), \quad \widehat{V}_F^{\lambda}(s_t) := \sum_{l=t}^{T-1} \lambda^{l-t}\hat{\delta}_F(s_l, a_l) + \widetilde{V}_F(s_t), \\
&\hat{\delta}_F(s_t, a_t) := R_F(s_t, a_t) + \widetilde{V}_F(s_{t+1}) - \widetilde{V}_F(s_t),
\end{aligned} \tag{66}$$

$\widehat{A}_F$ is called **critic** and π_F is called **actor**. It can be verified that $\widehat{A}^1_F(s_t,a_t)=\sum_{l=t}^{T-1}R_F(s_l,a_l)-\widetilde{V}_F(s_t;\eta)$ renders an unbiased estimator of $\nabla_{\theta_F}J(\theta)$ as the first term is an unbiased estimation of Q_F and $\widetilde{V}_F(\cdot;\eta)$ does not introduce estimation bias (Remark B.7); $\widehat{A}^0_F(s_t,a_t)=R(s_t,a_t)+\widetilde{V}_F(s_{t+1})-\widetilde{V}_F(s_t)$ provided an direct functional approximation of $A_F(s_t,a_t)$, which usually render biased estimation with lower variance as $\widetilde{V}_F$ may not equal to V_F exactly. Thus, λ enables the **variance-bias trade-off** for robust gradient estimation. Likewise, $\widehat{V}^1_F(s_t)=\sum_{l=t}^{T-1}R(s_l,a_l)$ and $\widehat{V}^0_F(s_t)=R(s_t,a_t)+\widetilde{V}_F(s_{t+1})$ for each τ, corresponding to unbiased and biased estimation of $V_F(s_t)$. Denoting the estimated gradient w.r.t. (θ_F,θ_Z,η) as $(\hat{g}_F,\hat{g}_Z,\hat{g}_V)$, these parameters are updated by $(\theta'_F,\theta'_Z,\eta')\leftarrow(\theta_F-\alpha_F\hat{g}_F,\theta_Z-\alpha_Z\hat{g}_Z,\eta-\alpha_V\hat{g}_V)$.

TRPO Parameters θ_Z and η are updated in the same way as the actor-critic method. Parameter θ_F is updated by the linear approximation of objective (6):

$$\begin{aligned}\min_{\theta'_F}\quad & T\,g_F^\top(\theta'_F-\theta_F)\\ \text{s.t.}\quad & \frac{1}{2}(\theta'_F-\theta_F)^\top H_F(\theta'_F-\theta_F)\le\zeta_F,\end{aligned}\tag{67}$$

with

$$g_F=\nabla_{\theta'_F}\mathbb{E}_{d_{F,\mu}(s;\theta),\pi_F(s,a;\theta'_F)}\left[\widehat{A}^\lambda_F(s,a;\theta_F)\right],\quad H_F=\nabla^2_{\theta'_F}D_{KL}^{d_{F,\mu}(\cdot;\theta)}\left(\pi_F(s,a;\theta_F),\pi_F(s,a;\theta'_F)\right).\tag{68}$$

Let's denote the Lagrangian formulation of the above problem as $L(d_F,\kappa):=Tg_F^\top d_F-\kappa(d_F^\top H_F d_F-\zeta_F)$ with Lagrangian constant κ and $d_F:=\theta'_F-\theta_F$. By the optimal conditions of $L(d_F,\kappa)$, $\nabla_\kappa L(d_F,\kappa)=0$ and $\nabla_{d_F}L(d_F,\kappa)=0$, we have $d_F=\frac{1}{\kappa}H_F^{-1}g_F$ and $\kappa=\left(\frac{g_F^\top H_F^{-1}g_F}{2\zeta_F}\right)^{0.5}$. Thus, the maximal updating step of model parameters is: $\theta'_F\leftarrow\theta_F-\left(\frac{2\zeta_F}{\hat{g}_F^\top\widehat{H}_F^{-1}\hat{g}_F}\right)^{0.5}\widehat{H}_F^{-1}\hat{g}_F$. When the dimension of θ'_F is high, computing $\widehat{H}_F^{-1}$ is time-demanding. Thus, we adopt the conjugate gradient method to estimate $\widehat{H}_F^{-1}\hat{g}_F$ based on $\widehat{H}_F\hat{g}_F$ (Hestenes et al., 1952). Besides, following Schulman et al. (2015), we perform a line search of updating step size to improve performance, instead of taking the maximal step.

C. Performance analysis

Lemma C.1. *(Descent lemma (Beck, 2017)) Supposing $f(\cdot)$ is a β-smooth function, then for any θ and θ':*

$$f(\theta')\le f(\theta)+\langle\nabla_\theta f(\theta),\theta'-\theta\rangle+\frac{\beta}{2}\|\theta-\theta'\|_2^2.\tag{69}$$

Lemma C.2. *Given two forward policies (π'_F,π_F) with $D_{KL}^{d'_{F,\mu}}(\pi'_F(s,\cdot),\pi_F(s,\cdot))\le\zeta_F$ or two backward (π'_B,π_B) with $D_{KL}^{d'_{B,\rho}}(\pi'_B(s,\cdot),\pi_B(s,\cdot))\le\zeta_B$, we have*

$$\begin{aligned}\frac{1}{T}(J'_F-J_F)&\le\mathbb{E}_{d'_{F,\mu}(s),\pi'_F(s,a)}[A_F(s,a)]+\zeta_F,\\ \frac{1}{T-1}(J'_B-J_B)&\le\mathbb{E}_{d'_{B,\rho}(s),\pi'_B(s,a)}[A_B(s,a)]+\zeta_B.\end{aligned}\tag{70}$$

Proof. The proof procedure is analogous to that of Schulman et al. (2015) and Rengarajan et al. (2022). By the definition of A_F,

$$\begin{aligned}\mathbb{E}_{P'_F(\tau|s_0)}\left[\sum_{t=0}^{T-1}A_F(s_t,a_t)\right]&=\mathbb{E}_{P'_F(\tau|s_0)}\left[\sum_{t=0}^{T-1}\left(R_F(s_t,a_t)+V_F(s_{t+1})-V_F(s_t)\right)\right]\\ &=\mathbb{E}_{P'_F(\tau|s_0)}\left[\sum_{t=0}^{T-1}R_F(s_t,a_t)\right]+\mathbb{E}_{P'_F(\tau|s_0)}[\underbrace{V_F(s_T)}_{=0}]-V_F(s_0)\\ &=\mathbb{E}_{P'_F(\tau|s_0)}\left[\sum_{t=0}^{T-1}\left(R'_F(s_t,a_t)+R_F(s_t,a_t)-R'_F(s_t,a_t)\right)\right]-V_F(s_0)\\ &=V'_F(s_0)-V_F(s_0)+\mathbb{E}_{P'_F(\tau|s_0)}\left[\sum_{t=0}^{T-1}(R_F(s_t,a_t)-R'_F(s_t,a_t))\right].\end{aligned}\tag{71}$$

Thus,

$$\begin{aligned} J'_F - J_F &= \mathbb{E}_{P'_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} A_F(s_t, a_t)\right] + \mathbb{E}_{P'_{F,\mu}(\tau)}\left[\sum_{t=0}^{T-1} D_{KL}(\pi'_F(s_t,\cdot), \pi_F(s_t,\cdot))\right] \\ &= T\,\mathbb{E}_{d'_{F,\mu}(s),\pi'_F(s,a)}[A_F(s,a)] + T\,D_{KL}^{d'_{F,\mu}}(\pi'_F(s,\cdot), \pi_F(s,\cdot)) \\ &\le T\left(\mathbb{E}_{d'_{F,\mu}(s),\pi'_F(s,a)}[A_F(s,a)] + \zeta_F\right). \end{aligned} \tag{72}$$

Using the fact that $V'_B(s_0) = 0$ and backward rewards are accumulated from $T-1$ back to 1, the results for the backward case can be derived by a similar procedure as in the forward case, so it is omitted here. □

C.1. Proof of Theorem 3.5

Proof. Firstly,

$$\begin{aligned} J_F^G &= J_F + (J_F^G - J_F) \\ &= J_F + \mathbb{E}_{P_{F,\mu}(\tau)}\left[\log \frac{P_F(\tau|s_0)Z}{P_G(\tau|x)R(x)} - \log \frac{P_F(\tau|s_0)Z}{P_B(\tau|x)R(x)}\right] \\ &= J_F + \mathbb{E}_{P_{F,\mu}(\tau)}\left[\log \frac{P_B(\tau|x)}{P_G(\tau|x)}\right] + \mathbb{E}_{P_{B,\rho}(\tau)}\left[\log \frac{P_B(\tau|x)}{P_G(\tau|x)}\right] - \mathbb{E}_{P_{B,\rho}(\tau)}\left[\log \frac{P_B(\tau|x)}{P_G(\tau|x)}\right] \\ &= J_F + J_B^G + \sum_{\tau} \left(P_{F,\mu}(\tau) - P_{B,\rho}(\tau)\right) R_B^G(\tau), \end{aligned} \tag{73}$$

where $R_B^G(\tau) := \log \frac{P_B(\tau|x)}{P_G(\tau|x)} = \sum_{t=1}^{T-1} R_B^G(s_t, a_t)$. Then,

$$\begin{aligned} J_F^G &= J_F + J_B^G + \left\langle P_{F,\mu}(\cdot) - P_{B,\rho}(\cdot), R_B^G(\cdot)\right\rangle \\ &\le J_F + J_B^G + \left\|P_{F,\mu}(\cdot) - P_{B,\rho}(\cdot)\right\|_1 \left\|R_B^G(\cdot)\right\|_\infty \\ &\le J_F + J_B^G + (T-1)\,\left\|P_{F,\mu}(\cdot) - P_{B,\rho}(\cdot)\right\|_1 R_B^{G,\max}, \end{aligned} \tag{74}$$

where the first inequality holds by Hölder's inequality, and the second inequality holds by $R_B^{G,\max} := \max_{s,a}\left|R_B^G(s,a)\right| \ge \frac{1}{T-1}\max_\tau \left|R_B^G(\tau)\right|$. By Pinsker's inequality:

$$\left\|P_{F,\mu}(\cdot) - P_{B,\rho}(\cdot)\right\|_1 \le \sqrt{\frac{1}{2} D_{KL}(P_{F,\mu}(\tau), P_{B,\rho}(\tau))}. \tag{75}$$

Besides,

$$\begin{aligned} D_{KL}(P_{F,\mu}(\tau), P_{B,\rho}(\tau)) &= \mathbb{E}_{P_{F,\mu}(\tau)}\left[\log \frac{P_{F,\mu}(\tau|x)P_{F,\mu}^\top(x)}{P_B(\tau|x)P_{F,\mu}^\top(x)}\right] \\ &\le \mathbb{E}_{P_{F,\mu}(\tau)}\left[\log \frac{P_{F,\mu}(\tau|x)}{P_B(\tau|x)}\right] + \underbrace{\mathbb{E}_{P_{F,\mu}^\top(x)}\left[\log \frac{P_{F,\mu}^\top(x)}{R(x)/Z^*}\right]}_{\ge 0} \\ &= D_{KL}(P_{F,\mu}(\tau), P_B(\tau)) \\ &= D_{KL}^\mu(P_F(\tau|s_0), P_B(\tau|s_0)) + \underbrace{D_{KL}(\mu(s_0), P_B(s_0))}_{=0} \\ &= D_{KL}^\mu(P_F(\tau|s_0), \widetilde{P}_B(\tau|s_0)) - \log Z + \log Z^* \\ &= J_F + \log Z^* - \log Z. \end{aligned} \tag{76}$$

Then, we have:

$$J_F^G \le J_F + J_B^G + (T-1)\,R_B^{G,\max}\sqrt{\frac{1}{2}(J_F + \log Z^* - \log Z)}. \tag{77}$$

□

C.2. Proof of Theorem 3.6

Proof. By Lemma C.2 and the definition of ζ_F:

$$\frac{1}{T}(J'_F - J_F) \leq \mathbb{E}_{d'_{F,\mu}(s),\pi'_F(s,a)}[A_F(s,a)] + \zeta_F. \tag{78}$$

Let $\bar{A}_F \in R^{|\mathcal{S}|}$ denote the vector components of $\mathbb{E}_{\pi'_F(s,a)}[A_F(s,a)]$. Then, we have:

$$\begin{aligned}\mathbb{E}_{d'_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] &= \left\langle d'_{F,\mu}, \bar{A}_F \right\rangle \\ &= \left\langle d_{F,\mu}, \bar{A}_F \right\rangle + \left\langle d'_{F,\mu} - d_{F,\mu}, \bar{A}_F \right\rangle \\ &\leq \mathbb{E}_{d_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] + \left\| d'_{F,\mu} - d_{F,\mu} \right\|_1 \left\| \bar{A}_F \right\|_\infty, \end{aligned} \tag{79}$$

where the last inequality holds by Hölder's inequality. By Lemma B.5 and the definition of ϵ_F:

$$\mathbb{E}_{d'_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] \leq \mathbb{E}_{d_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] + 2\mathbb{E}_{d'_{F,\mu}(s)}\left[D_{TV}(\pi'_F(s,\cdot), \pi_F(s,\cdot))\right]\epsilon_F. \tag{80}$$

By Pinsker's inequality,

$$D_{TV}(\pi'_F(s,\cdot), \pi_F(s,\cdot)) \leq \left(\frac{1}{2} D_{KL}(\pi'_F(s,\cdot), \pi_F(s,\cdot)\right)^{0.5}.$$

By Jensen's inequality and the definition of ζ_F,

$$\mathbb{E}_{d'_{F,\mu}(s)}\left[\left(\frac{1}{2} D_{KL}(\pi'_F(s,\cdot), \pi_F(s,\cdot)\right)^{0.5}\right] \leq \left(\frac{1}{2}\mathbb{E}_{d'_{F,\mu}(s)}\left[D_{KL}(\pi'_F(s,\cdot), \pi_F(s,\cdot)\right]\right)^{0.5} \leq \left(\frac{\zeta_F}{2}\right)^{0.5}.$$

Thus, we have:

$$\mathbb{E}_{d'_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] \leq \mathbb{E}_{d_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] + (2\zeta_F)^{0.5}\epsilon_F. \tag{81}$$

Combing inequalities (81) and (78), we have:

$$\frac{1}{T}(J'_F - J_F) \leq E_{d_{F,\mu}(s)\pi'_F(s,a)}[A_F(s,a)] + \zeta_F + (2\zeta_F)^{0.5}\epsilon_F. \tag{82}$$

□

C.3. Proof of Theorem 3.7

Proof. By Lemma C.1,

$$J_F(\theta_{n+1}) \leq J_F(\theta_n) + \langle \nabla_{\theta_n} J_F(\theta_n), \theta_{n+1} - \theta_n \rangle + \frac{\beta}{2}\left\|\theta_{n+1} - \theta_n\right\|_2^2. \tag{83}$$

Thus,

$$\begin{aligned} -\langle \nabla_{\theta_n} J_F(\theta_n), \theta_{n+1} - \theta_n \rangle &\leq J_F(\theta_n) - J_F(\theta_{n+1}) + \frac{\beta}{2}\left\|\theta_{n+1} - \theta_n\right\|_2^2, \\ \alpha\left\langle \nabla_{\theta_n} J_F(\theta_n), \widehat{\nabla}_{\theta_n} J_F(\theta_n)\right\rangle &\leq J_F(\theta_n) - J_F(\theta_{n+1}) + \frac{\beta\alpha^2}{2}\left\|\widehat{\nabla}_{\theta_n} J_F(\theta_n)\right\|_2^2. \end{aligned}$$

Conditioning on θ_n, taking expectations over both sides and noting that $\mathbb{E}_{P(\cdot|\theta_n)}\left[\left\langle \nabla_{\theta_n} J_F(\theta_n), \widehat{\nabla}_{\theta_n} J_F(\theta_n)\right\rangle\right] = \left\langle \nabla_{\theta_n} J_F(\theta_n), \mathbb{E}_{P(\cdot|\theta_n)}[\widehat{\nabla}_{\theta_n} J_F(\theta_n)]\right\rangle = \left\|\nabla_{\theta_n} J_F(\theta_n)\right\|_2^2$, we have:

$$\alpha\left\|\nabla_{\theta_n} J_F(\theta_n)\right\|_2^2 \leq J_F(\theta_n) - \mathbb{E}_{P(\theta_{n+1}|\theta_n)}\left[J_F(\theta_{n+1})\right] + \frac{\beta\alpha^2}{2}\mathbb{E}_{P(\cdot|\theta_n)}\left[\left\|\widehat{\nabla}_{\theta_n} J_F(\theta_n)\right\|_2^2\right]. \tag{84}$$

By the assumption that $\mathbb{E}_{P(\cdot|\theta)}\left[\left\|\widehat{\nabla}_\theta J_F(\theta) - \nabla_\theta J_F(\theta)\right\|_2^2\right] = \mathbb{E}_{P(\cdot|\theta)}\left[\left\|\widehat{\nabla}_\theta J_F(\theta)\right\|_2^2\right] - \left\|\nabla_\theta J_F(\theta)\right\|_2^2 \leq \sigma_F$, we have:

$$\alpha\left\|\nabla_{\theta_n} J_F(\theta_n)\right\|_2^2 \leq J_F(\theta_n) - \mathbb{E}_{P(\theta_{n+1}|\theta_n)}\left[J_F(\theta_{n+1})\right] + \frac{\beta\alpha^2}{2}\left\|\nabla_{\theta_n} J_F(\theta_n)\right\|_2^2 + \frac{\beta\alpha^2\sigma_F}{2}. \tag{85}$$

Consequently, we have:

$$\left(\alpha - \frac{\beta\alpha^2}{2}\right) \mathbb{E}_{P(\theta_{0:N-1})}\left[\sum_{n=0}^{N-1} \|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \frac{N\beta\alpha^2\sigma_F}{2} + \mathbb{E}_{P(\theta_{0:N})}\left[\sum_{n=0}^{N-1} J_F(\theta_n) - J_F(\theta_{n+1})\right],$$

$$\left(\alpha - \frac{\beta\alpha^2}{2}\right) \sum_{n=0}^{N-1} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \frac{N\beta\alpha^2\sigma_F}{2} + \mathbb{E}_{P(\theta_{0:N})}\left[J_F(\theta_0) - J_F(\theta_N)\right],$$

$$\left(\alpha - \frac{\beta\alpha^2}{2}\right) N \min_{n\in\{0,\ldots,N-1\}} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \frac{N\beta\alpha^2\sigma_F}{2} + \mathbb{E}_{P(\theta_0)}\left[J_F(\theta_0)\right] - \mathbb{E}_{P(\theta_N)}\left[J_F(\theta_N)\right]. \tag{86}$$

Setting $\alpha = \sqrt{2/(\beta N)}$, we have:

$$\left(\sqrt{(2N)/\beta} - 1\right) \min_{n\in\{0,\ldots,N-1\}} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \sigma_F + \mathbb{E}_{P(\theta_0)}\left[J_F(\theta_0)\right] - \mathbb{E}_{P(\theta_N)}\left[J_F(\theta_N)\right]. \tag{87}$$

Since $J_F(\theta) + \log Z^* - \log Z(\theta) = D_{KL}^{\mu(\cdot;\theta)}(P_F(\tau|s_0;\theta), P_B(\tau|s_0))$, and $J_F(\theta^*) = 0$ with $\log Z^* = \log Z(\theta^*)$ for optimal parameter θ^*, then $J_F(\theta_N) + \log Z^* - \log Z(\theta_N) \geq J_F(\theta^*)$ and we have:

$$\begin{aligned}
\min_{n\in\{0,\ldots,N-1\}} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] &\leq \frac{\sigma_F + \mathbb{E}_{P(\theta_0)}\left[J_F(\theta_0)\right] - \mathbb{E}_{P(\theta_N)}\left[J_F(\theta_N) + \log Z^* - \log Z(\theta_N)\right]}{\left(\sqrt{(2N)/\beta} - 1\right)} \\
&\quad + \frac{\mathbb{E}_{P(\theta_N)}\left[\log Z^* - \log Z(\theta_N)\right]}{\left(\sqrt{(2N)/\beta} - 1\right)} \\
&\leq \frac{\sigma_F + \mathbb{E}_{P(\theta_0)}\left[J_F(\theta_0)\right] + \mathbb{E}_{P(\theta_N)}\left[|\log Z^* - \log Z(\theta_N)|\right]}{\left(\sqrt{(2N)/\beta} - 1\right)}.
\end{aligned} \tag{88}$$

By the assumption that $|\log Z - \log Z^*| \leq \sigma_Z$, we have:

$$\min_{n\in\{0,\ldots,N-1\}} \mathbb{E}_{P(\theta_n)}\left[\|\nabla_{\theta_n} J_F(\theta_n)\|_2^2\right] \leq \frac{\sigma_F + \sigma_Z + \mathbb{E}_{P(\theta_0)}[J_F(\theta_0)]}{\left(\sqrt{(2N)/\beta} - 1\right)}. \tag{89}$$

□

D. Additional discussion about policy-based and value-based methods

The goal of traditional RL is to learn a policy π that achieves the optimality in the expected accumulated reward J_π (for GFlowNet training, corresponding to the distance between $P_F(\tau)$ and $P_B(\tau)$, $\text{Dist}(P_F(\tau), P_B(\tau))$) addressing the challenge of the exploration-exploitation (Exp-Exp) dilemma. While valued-based methods are usually off-policy allowing to explicitly balance the **Exp-Exp trade-off** by designing $P_\mathcal{D}$, the objectives of the valued-based methods are optimized to encourage the improvement of J_π but they do not directly solve the optimization formulation with J_π. Policy-based methods directly optimize J_π w.r.t. π, enabling optimization techniques that tackle the **Exp-Exp trade-off** implicitly but efficiently. Our joint framework manages to inherit both advantages of the value-based and the policy-based methods by keeping the optimization formulation of J_π and allowing explicit design of P_G as $P_\mathcal{D}$. We provide more detailed explanations of our arguments as follows:

- The **Exp-Exp dilemma** is the main challenge in decision-making including different reinforcement learning (RL) formulations. RL is guided by reward functions. To learn the desired policies, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing rewards (exploitation). But to discover such actions, it has to try actions that it has not been selected before (exploration) at the expense of an exploitation opportunity (Sutton & Barto, 2018). Therefore, both policy-based and value-based methods face the fundamental challenge and try to overcome them in different ways.
- In RL, the goal is to learn a policy π that achieves the optimality in the expected accumulated reward J_π. The value-based methods, represented by Q-learning and Soft-Q-learning, do not optimize J_π w.r.t. π directly. They

leverage the fact that the optimal policy should satisfy the Bellman equation. By minimizing the mismatch of the Bellman equation, which typically takes an off-policy form $\mathbb{E}_{(s,a)\sim P_{\mathcal{D}}}[(Q_\pi(s,a) - \widehat{Q}_\pi(s,a))^2]$ (Haarnoja et al., 2018), the corresponding J_π is encouraged to be improved. The improvement, however, is not guaranteed, since they do not directly solve the optimization formulation with J_π. So the core of the value-based methods turns to explicitly design a sampler (agent), $P_{\mathcal{D}}$, that can effectively identify the state-action pair that gives rise to the mismatch the most (exploration) while allowing revisiting the state-action pair that has already been found to be effective (exploitation), and how to represent the target function $\widehat{Q}_\pi$ that approximates the optimal function Q^* while balancing the trade-off properly as well. To exemplify that both exploration and exploitation are important, let's take the hyper-grid experiment as an example, where modes are highly separated but do not exactly lie on the margin of the grids. In the extreme case, a purely explorative sampler will always favor taking actions that lead to visiting the marginal coordinates, which, however, yield low rewards.

- Likewise, original GFlowNet methods do not optimize the distance between $P_F(\tau)$ and $P_B(\tau)$ directly (so that $P_F^\top(x) = P_B(x)$), but optimizes the flow mismatch associated with, P_F, which implicitly encourages the minimization of the distance and the core of training efficiency is to design a sampler $P_{\mathcal{D}}$ that effectively balance the **Exp-Exp trade-off**. This motivates back-and-forth local search (Kim et al., 2023b), Thompson sampling (Rector-Brooks et al., 2023) and temperature conditioning (Kim et al., 2023a). Besides, Detailed Balance (DB) objective can be understood as favoring exploitation as the 'target' edge flow is $P_B(s|s')F(s')$, where $F(s')$ is learned from the data collected so far and represents our partial knowledge about the environment. The Trajectory Balance (TB) objective can be understood as favoring exploration as the target trajectory flow is $P_B(\tau|x)R(x)$ which can be fixed w.r.t. P_F and regarded as pure environment feedback. So, finding better 'target' flow representations that properly balance the **Exp-Exp trade-off** is one of the common motivations for sub-trajectory balance (Madan et al., 2023), forward-looking (Pan et al., 2023), and energy decomposition (Kim et al., 2023a) approaches.

- Policy-based methods reformulate the problem of balancing the Bellman equation into optimizing J_π w.r.t. π directly. On one hand, the nature of the policy-based methods requires to be on-policy (i.e. $P_{\mathcal{D}} = \pi$), so we can not explicitly design $P_{\mathcal{D}}$ to overcome the exploration-exploitation dilemma. On the other hand, it saves us from the difficulty in sampler design. The policy-based methods compute the gradients of J_π w.r.t. π and learn π by gradient-based strategies. So the problem of designing a sampler is converted to gradient-based optimization with robust gradient estimation. Related techniques include variance reduction techniques, improvement of gradient descent directions like natural policy gradients and mirror policy descent, and conservative policy updates such as TRPO and PPO. These methods implicitly address **Exp-Exp trade-off**. Because a policy π that always favors either exploration or exploitation will not render the maximum or minimum J_π unless it is equal to the optimal policy. Besides, the intuition of conservative policy updates like TRPO is that we keep the policy unchanged to prevent it from getting trapped into local optima (exploration) unless we find a better point in the trust region (exploitation).

- Our joint training framework manages to inherit the advantages of both value-based methods and policy-based methods. It keeps optimizing J_π directly and makes the associated gradient-based optimization techniques applicable. In the meanwhile, we can explicitly design guided policy as the design of $P_{\mathcal{D}}$ in the off-policy case, to integrate expert knowledge about the environment and balance the **Exp-Exp trade-off** explicitly.

E. Additional experimental settings and results

In all experiments, we follow a regular way of designing off-policy sampler,$P_{\mathcal{D}}$ for value-based methods: $P_{\mathcal{D}}$ is a mixture of the learned forward policy and a uniform policy where the mix-in factor of the uniform policy starts at 1.0 and decays exponentially at a rate of γ after each training iteration, where γ is set to 0.99 based on the results of the ablation study. In TB-Sub, the objective is a convex combination of the sub-trajectory balance losses following Madan et al. (2023), where the hyperparameter that controls the weights assigned to sub-trajectories of different lengths, is set to 0.9 selected from $\{0.80, 0.85, 0.90, 0.95, 0.99\}$. For our policy-based methods, we set the value of hyper-parameter λ to 0.99 for the forward policy gradients based on the results of the ablation study. The backward policy gradient is estimated unbiasedly, meaning the hyper-parameter value is 1. Trust region hyper-parameter ζ_F is set to 0.01 select from $\{0.01, 0.02, 0.03, 0.04, 0.05\}$.

We use the Adam optimizer for model optimization. The learning rates of forward and backward policy are equal to 1×10^{-3}, which is selected from $\{5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$ by TB-U. The learning rates of value functions are set to 5×10^{-3}, which is selected from $\{1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}\}$ by RL-U. The learning rates of total flow estimator is

1×10^{-1}, which is selected from $\{1 \times 10^{-1}, 5 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}\}$ by TB-U. The sample batch size is set to 128 for each optimization iteration. For all experiments, we report the performance with five different random seeds.

E.1. Evaluation metrics

The total variation D_{TV} between $P_F^\top(x)$ and $P^*(x)$ is defined as:

$$D_{TV}(P_F^\top, P^*) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P_F^\top(x) - P^*(x)|. \tag{90}$$

The total variation is similar to the average l_1-distance used in prior works, which can be computed by $\frac{1}{|\mathcal{X}|} \sum_x |P^*(x) - P_F^\top(x)|$. However, the average l_1-distance may be inappropriate as $|\mathcal{X}|$ is usually large ($> 10^4$) and $\sum_x |P^*(x) - P_F^\top(x)| \leq 2$, resulting in the average l_1-distance being heavily scaled down by $|\mathcal{X}|$.

The Jensen–Shannon divergence D_{JSD} between $P_F^\top(x)$ and $P^*(x)$ is defined as:

$$D_{JSD}(P_F^\top, P^*) = \frac{1}{2} D_{KL}(P_F^\top, P^M) + \frac{1}{2} D_{KL}(P^*, P^M), \quad P^M = P_F^\top + P^*. \tag{91}$$

Following Shen et al. (2023), the mode accuracy Acc of $P_F^\top(x)$ w.r.t. $P^*(x)$ is defined as:

$$Acc(P_F^\top, P^*) = \min \left(\frac{\mathbb{E}_{P_F^\top(x)}[R(x)]}{\mathbb{E}_{P^*(x)}[R(x)]}, 1 \right). \tag{92}$$

For biological and molecular sequence experiments, we also count the number of modes, that is, the number of modes discovered during training. At every 10 training iterations, we sample $|\mathcal{X}^{\text{mode}}|$ terminating states by the current learned $P_F(\cdot|\cdot)$ and store the states which are modes and have not been discovered before; then we count the total number of discovered unique modes for evaluation. (Shen et al., 2023; Kim et al., 2023b). The mode set $\mathcal{X}^{\text{mode}}$ is defined as the set of terminating states whose rewards are in the top 0.5%, 0.5%, 0.5%, and 0.1% of all rewards for the SIX6, QM9, PHO4 and sEH datasets respectively.

E.2. Hyper-grid modeling

Environment In this environment, $\mathcal{S} \setminus \{s^f\}$ is equal to $\{s = ([s]_0, \ldots, [s]_d, \ldots, [s]_D) | [s]_d \in \{0, \ldots, N-1\}\}$ $(= \{1, \ldots, N-1\}^D)$, where the initial state $s^0 = (0, \ldots, 0)$, and the final state s^f can be represented by any invalid coordinate tuple of the hyper-grid, denoted as $(-1, \ldots, -1)$ in our implementation. For state $s \in \mathcal{S} \setminus \{s^f\}$, we have $D+1$ possible actions in $\mathcal{A}(s)$: (1) increment the coordinate $[s]_d$ by one, arriving at $s' = ([s]_0, \ldots, [s]_d + 1, \ldots)$; (2) choose stopping actions $(s \to s^f)$, terminating the process and returning s as the terminating coordinate tuple x. In this environment, $\mathcal{G}$ is not a graded DAG, and $\mathcal{S} \setminus \{s^f\} = \mathcal{X}$ as all coordinate tuples can be returned as the terminating states. The reward $R(x)$ is defined as:

$$R(x) = R_0 + R_1 \prod_{d=1}^{D} \mathbb{I}\left[\left| \frac{[s]_d}{N-1} - 0.5 \right| \in (0.25, 0.5] \right] + R_2 \prod_{d=1}^{D} \mathbb{I}\left[\left| \frac{[s]_d}{N-1} - 0.5 \right| \in (0.3, 0.4] \right], \tag{93}$$

where $R_0 = 10^{-2}$, $R_1 = 0.5$ and $R_2 = 2$ in our experiment. Conditioning on x, we use an unnormalized conditional guided trajectory distribution $\widetilde{P}_G(\tau|x)$ for backward policy design, which is defined as:

$$\begin{aligned}
\widetilde{P}_G(\tau | x \to s^f) &:= P_f(\tau_{\preceq x}) = \prod_{t=1}^{T-1} P_f(s_t | s_{t-1}), \\
\forall s_t \neq s^f : P_f(s_t | s_{t-1}) &:= \begin{cases} \frac{P_F(s_t|s_{t-1})}{\sum_{s: s \neq s^f} P_F(s|s_{t-1}) + \epsilon^f} & \text{if } R(s_{t-1}) \leq R_0 \\ P_F(s_t|s_{t-1}) & \text{otherwise} \end{cases}, \\
P_f(s^f | s_{t-1}) &:= \begin{cases} \frac{\epsilon^f}{\sum_{s: s \neq s^f} P_F(s|s_{t-1}) + \epsilon^f} & \text{if } R(s_{t-1}) \leq R_0 \\ P_F(s^f|s_{t-1}) & \text{otherwise} \end{cases},
\end{aligned} \tag{94}$$

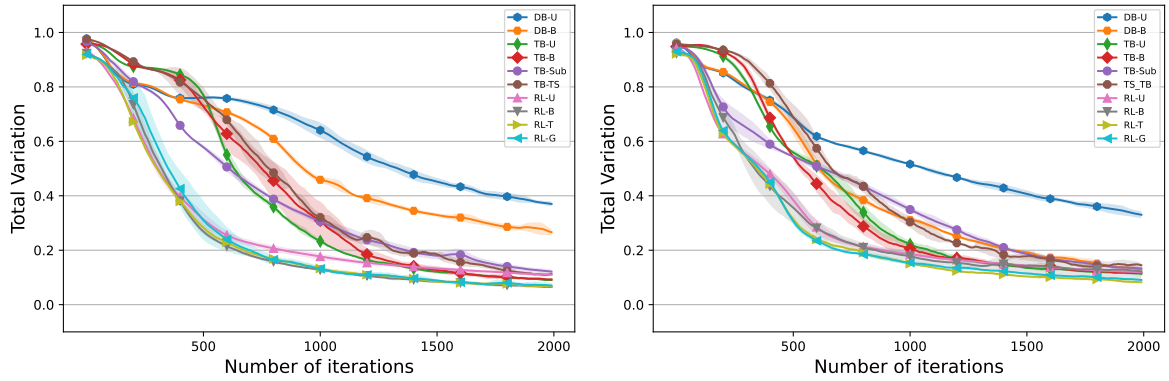

Figure 4. Training curves by D_{TV} between $P_F^{\top}$ and P^* for $64 \times 64 \times 64$ (left) and $32 \times 32 \times 32 \times 32$ hyper-grids (right). The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

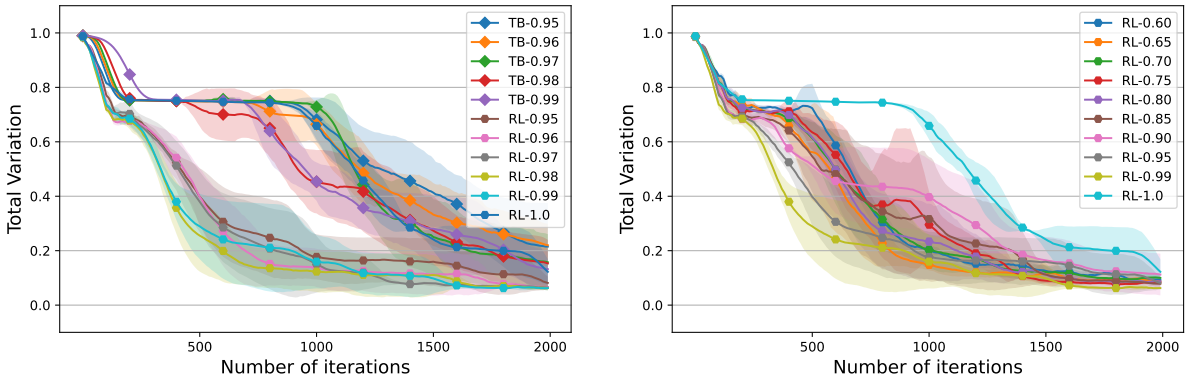

Figure 5. Performance comparison between RL-U with different λ values and TB-U with different γ values. The curves are plotted based on their mean and standard deviation values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

where $\epsilon^f = 10^{-5}$, the corresponding normalized distribution can be understood as $P_G(\tau|x \to s^f) = P_f(\tau|x \to s^f) \propto P_f(\tau_{\preceq x})$, and $P_f(\tau) = \prod_{t=1}^{T} P_f(s_t|s_{t-1})$. Similar to the proof of Proposition 3.1, it can be verified that $\nabla_\phi D^\rho_{KL}(P_B(\tau|x \to s^f;\phi), P_G(\tau|x \to s^f)) = \nabla_\phi D^\rho_{KL}(P_B(\tau|x \to s^f;\phi), \widetilde{P}_G(\tau|x \to s^f))$. As all the coordinate tuples can be terminating states (i.e., s^f is the child of all the other states) the expression above means that P_f assigns a low probability to the event of the terminating state being a state with a low reward. In this way, we discourage the generative process from stopping early at low reward coordinate tuples.

Model Architecture Forward policy $P_F(\cdot|\cdot)$ is parametrized by a neural network with 4 hidden layers and the hidden dimension is 256. Backward policy $P_B(\cdot|\cdot)$ is fixed to be uniform over valid actions or parameterized in the same way as P_F. Coordinate tuples are transformed by K-hot encoding before being fed into neural networks.

Additional Experiment Results The obtained results from five runs on $64 \times 64 \times 64$ and $32 \times 32 \times 32 \times 32$ grids are summarized in Fig. 4 and Table 2. The graphical illustrations of $P_F^{\top}(x)$ are shown in Figs. 14 and 15. We can observe similar performance trends to those in 256×256 and 128×128 grids: both TB-based methods and our policy-based method are better than DB-based method, and our policy-based methods achieve much faster convergence than TB-based methods. While these trends are less obvious than in 256×256 and 128×128 grids, this phenomenon can be ascribed to the fact that the environment height N has more influence on the modeling difficulty than the environment dimension D. The reason is that hyper-grids are homogeneous w.r.t. each dimension, and the minimum distance between modes only depends on N.

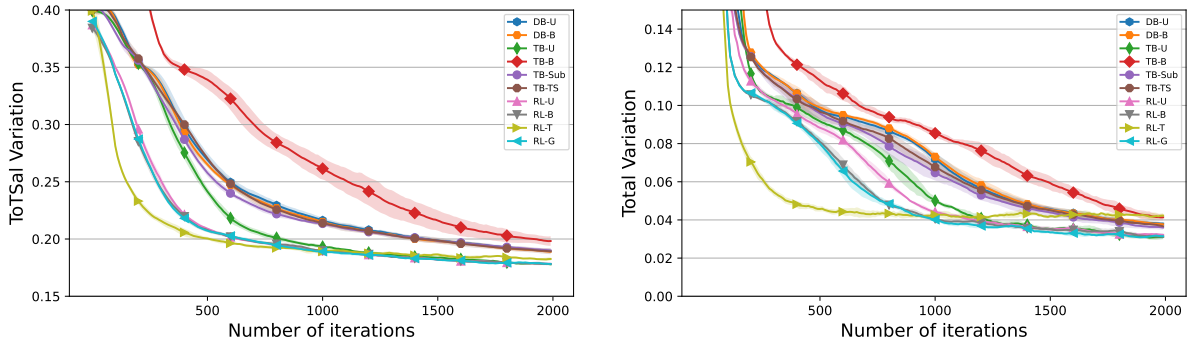

Figure 6. Training curves by D_{TV} between $P_F^{\top}$ and P^* for SIX6 (left) and QM9 (right). The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

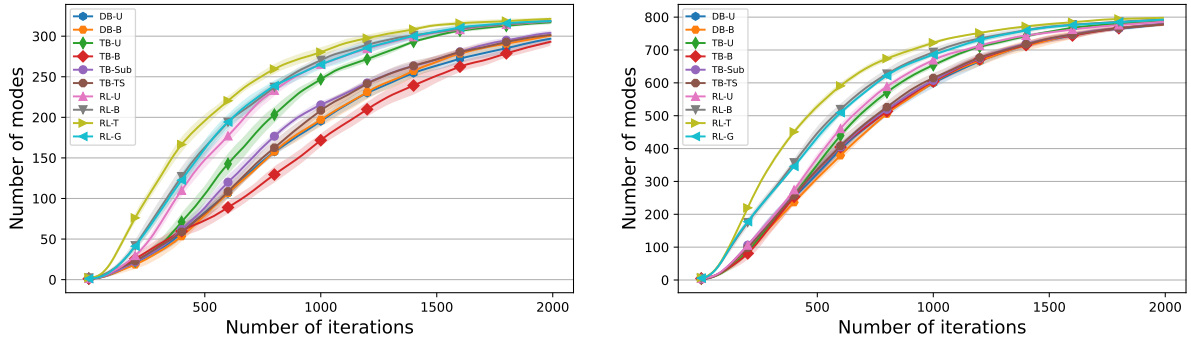

Figure 7. Training curves by the number of modes discovered during training for SIX6 (left) and QM9 (right). The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

E.3. Biological and Molecular Sequence design

Environment In this environment, $\mathcal{S} \setminus s^f = \{-1, 0, \ldots, N-1\}^D$ with element s corresponds to a sequence composed of integers ranging from -1 to $N-1$. The set $\{0, \ldots, N-1\}$ denotes the N nucleotide types or molecular building blocks, and the integer -1 represents that the corresponding position within s is unfilled. The initial state $s^0 = (-1, \ldots, -1)$ represents an empty sequence and the final state $s^f = (N, \ldots, N)$. For $s_t \in \mathcal{S}_t$, there are t elements in $\{0, \ldots, N-1\}$ and the rest equal to -1. There are $N \cdot (D-t)$ actions in $\mathcal{A}(s)$ that correspond to fill in one of the empty slots by one integer in $\{0, \ldots, N-1\}$. The generative process will not stop until sequences are fulfilled. By definition, $\mathcal{G}$ is a graded DAG and $\mathcal{S}_D = \mathcal{X} = \{0, \ldots, N-1\}^D$. We use the reward values provided in the dataset directly. Following Shen et al. (2023), we compute reward exponents $R^\beta(x)$ with hyper-parameter β set to 3, 5, 3, 6 and normalize the reward exponents to $[1 \times 10^{-3}, 10]$, $[1 \times 10^{-3}, 10]$, $[0, 10]$ and $[1 \times 10^{-3}, 10]$ for the SIX6, QM9, PHO4 and sEH datasets respectively. The guided distribution design also follows Shen et al. (2023). For content completeness, we provide the definitions as:

$$P_G(\tau|x) = \prod_{t=1}^{T} P_G(s_t|s_{t-1}, x), \quad P_G(s_t|s_{t-1}, x) = \frac{\text{score}(s_t|x)}{\sum_{s' \in Ch(s_{t-1})} \text{score}(s'|x)},$$
$$\text{score}(s|x) := \begin{cases} \text{mean}(\{R(x')|s \in x', x' \in \mathcal{X}^{\text{replay}}\}) & \text{if } s \in x \\ 0 & \text{otherwise} \end{cases} \tag{95}$$

where $\mathcal{X}^{\text{replay}}$ corresponds to a replay buffer that stores the sampled terminating states during training.

Model Architecture Policies are constructed in the same way as the hyper-grid modeling experiment. Integer sequences are transformed by K-hot encoding before being fed into neural networks.

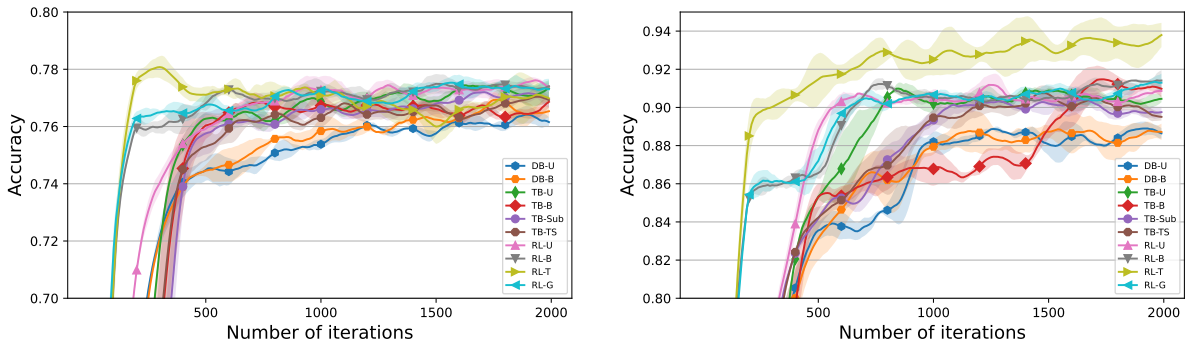

Figure 8. Training curves by Acc of $P_F^{\top}$ w.r.t. P^* for PHO4 and sEH. The curves are plotted based on their mean and standard deviation values. The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

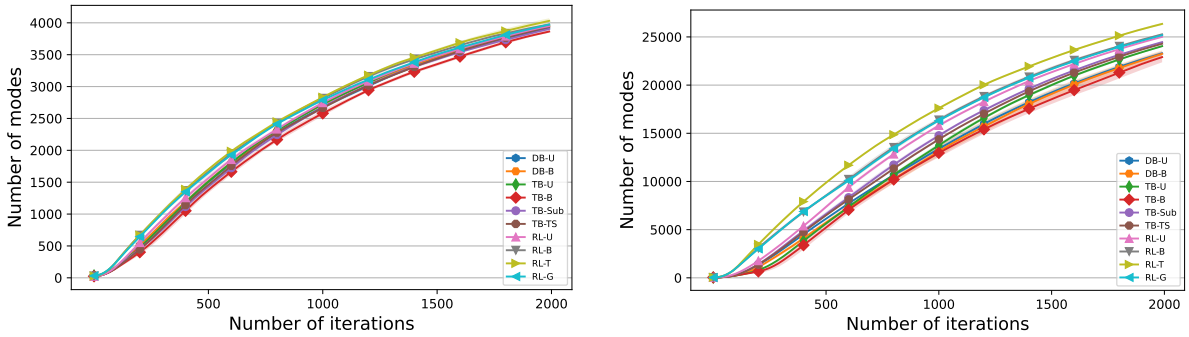

Figure 9. Training curves by the number of modes discovered over training for PHO4 and sEH. The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

Additional Experiment Results For PHO4 and sEH dataset, the exact computation of $P_F^{\top}$ by dynamic programming is expensive. Thus, we only plot the training curves by Acc and the number of modes as shown in Figs. 8 and 9, and the means and standard deviations of metric values at the last iteration are provided in Table 5. We can observe similar performance trends as in the QM9 and SIX6 datasets. Our policy-based methods achieve faster convergence rates and better converged Acc values than the TB-based and DB-based methods. While the converged Acc of RL-T is slightly worse than the other policy-based methods, it achieves the fastest convergence rate. RL-G and RL-B, both employing a parametrized π_B, demonstrate similar performance and converge faster than RL-U, which utilizes a uniform π_B.

E.4. Bayesian Network Structure Learning

In this experiment, we investigate GFlowNets for BN structure learning following the settings adopted in Malkin et al. (2022b). The set $\mathcal{X}$ corresponds to a set of BN structures, which are also DAGs. BN structure learning can be understood as approximating $P(x|\mathcal{D}) \propto R(x)$ given a dataset $\mathcal{D}$. Given a set of nodes, the state space for GFlowNets is the set of all possible DAGs over the node set. The actions correspond to adding edges over a DAG without introducing a cycle. The generative process of a BN structure is interpreted as starting from an empty graph, an action is taken to decide to add an edge or terminate the generative process at the current graph structure.

Environment A Bayesian Network is a probabilistic model that represents the joint distribution of N random variable and the joint distribution factorizes according to the network structure x:

$$P(y_1, \ldots, y_N) = \prod_{n=1}^{N} P(y_n | Pa_x(y_n)) \tag{96}$$

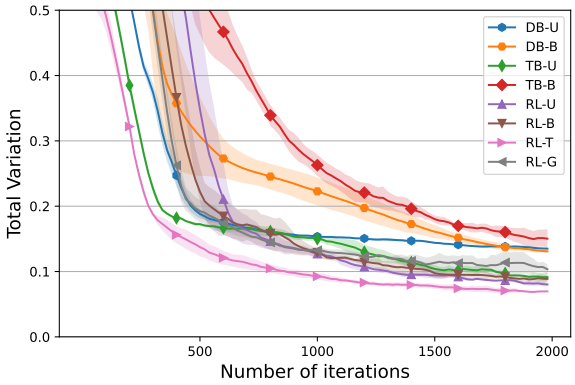

Figure 10. Training curves by D_{TV} of $P_F^{\top}$ w.r.t. P^* for the BN structure learning experiment. The curves are plotted based on means and standard deviations of metric values across five runs and smoothed by a sliding window of length 10. Metric values are computed every 10 iterations.

where $Pa_x(y_n)$ denote the set of parent nodes of y_n according to graph x. As the structure of any graph can be represented by its adjacency matrix, the state space can be defined as $\mathcal{S} := \left\{s|\mathcal{C}(s) = 0, s \in \{0,1\}^{N\times N}\right\}$ where $\mathcal{C}$ corresponds to the acyclic graph constraint (Deleu et al., 2022), the initial state $s^0 = \mathbf{0}^{N\times N}$ and specially $s^f := -\mathbf{1}^{N\times N}$ in our implementation. For each state s, $a \in \mathcal{A}(s)$ can be any action that turns one of 0 values of s to be 1 (i.e. adding an edge) while keeping $\mathcal{C}(s') = 0$ for the resulting graph s', or equal to $(s \to s^f)$ that stopping the generative process and return $x = s$ as the terminating state. By definition, the corresponding $\mathcal{G}$ in this environment is not graded. Given observation dataset $\mathcal{D}_y$ of $y_{1:N}$, the structure learning problem can be understood as approximating $P(x|\mathcal{D}_y) \propto P(x, \mathcal{D}_y) = P(\mathcal{D}_y|x)P(x)$. Without additional information about graph structure x, $P(x)$ is often assumed to be uniform. Thus, $P(x|\mathcal{D}_y) \propto P(\mathcal{D}_y|x)$ and the reward function is defined as $R(x) \propto P(\mathcal{D}_y|x)$. Distribution $P(\mathcal{D}_y|x)$ is also called graph score and we use BGe score (Kuipers et al., 2014) in our experiment. Following Malkin et al. (2022b), the ground-truth graph structure and the corresponding observation dataset $\mathcal{D}_y$ are simulated from Erdős–Rényi model. The guided distribution design follows the hyper-grid experiment. A low probability value, 10^{-5} is assigned to the transition probability of $(s \to s^f)$ if $|\log R(s) - \log R^{\max}| \leq 5$.

Model Architecture Policies are constructed in the same way as the hyper-grid modeling experiments, but adjacency matrices are fed into neural networks directly without encoding. As reported in (Deleu et al., 2022), the distribution can be very peaky between adjacent graph structures. The reward $R(x)$ (typically $\approx e^{80}$) in this experiment is much larger than those (typically ≈ 10) in the previous two sets of experiments. These facts give rise to numerical issues for reliable estimation of value function. Thus, we compute the gradients w.r.t. J_B empirically as the gradients w.r.t. TB-based objective, that is, V_B is not utilized during training. Besides, $\log Z$ is also very large, so we set the learning rate to 1, which is selected from $\{0.1, 0.3, 0.5, 0.8, 1.0\}$ by TB-U.

Experiment Results The number of possible DAGs grows exponentially with the number of nodes. Thus, we test the same benchmark in Malkin et al. (2022b) with the number of nodes set to 5 and the corresponding total numbers of DAGs is about 2.92×10^4. The number of edges in the ground-truth DAG is set to 5, and the size of the observation dataset is set to 100. The experimental results across five runs are shown in Fig. 10 and Table 6. The graphical illustrations of $P_F^{\top}(x)$ are shown in Fig. 18. As expected, performance trends similar to those in the previous two sets of experiments are observed. The converged D_{TV} values of all the policy-based methods are better than those of the value-based methods, with only TB-U achieving comparable performance. Besides, RL-T achieves the best converged D_{TV} value and has the fastest convergence among all the methods. These results further demonstrate the effectiveness of our policy-based methods for GFlowNet training.

E.5. Tables of converged metric values

	256×256			128×128		
Method	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Time↓	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Time↓
DB-U	6.050 ± 0.129	24.61 ± 1.147	37.2	3.233 ± 0.138	8.357 ± 0.471	18.5
DB-B	4.459 ± 0.135	14.42 ± 0.656	46.8	2.621 ± 0.122	5.621 ± 0.502	21.3
TB-U	0.728 ± 0.095	0.763 ± 0.131	**30.9**	0.449 ± 0.078	0.338 ± 0.062	**14.3**
TB-B	2.101 ± 1.052	6.612 ± 5.093	32.5	0.441 ± 0.080	0.355 ± 0.105	16.4
TB-Sub	1.461 ± 0.058	2.915 ± 0.285	71.5	0.450 ± 0.032	0.367 ± 0.060	31.3
TB-TS	1.277 ± 0.268	1.714 ± 0.627	38.5	0.481 ± 0.050	0.364 ± 0.058	20.4
RL-U	0.621 ± 0.057	0.770 ± 0.096	44.4	0.440 ± 0.077	0.390 ± 0.053	18.3
RL-B	0.704 ± 0.190	1.064 ± 0.286	68.5	0.490 ± 0.088	0.467 ± 0.082	35.2
RL-T	0.708 ± 0.058	0.774 ± 0.054	90.2	0.503 ± 0.043	0.426 ± 0.053	58.3
RL-G	$\mathbf{0.439 \pm 0.037}$	$\mathbf{0.541 \pm 0.017}$	69.8	$\mathbf{0.427 \pm 0.158}$	$\mathbf{0.353 \pm 0.180}$	35.4

Table 1. Converged metric values of different methods for the modeling of 256×256 and 128×128 grids. Training time costs are provided in minutes.

	$64 \times 64 \times 64$			$32 \times 32 \times 32 \times 32$		
Method	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Time↓	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Time↓
DB-U	3.687 ± 0.132	10.83 ± 0.611	19.3	3.254 ± 0.151	7.570 ± 0.682	19.1
DB-B	2.606 ± 0.083	6.559 ± 0.486	21.5	1.248 ± 0.041	1.684 ± 0.119	20.4
TB-U	0.870 ± 0.065	0.737 ± 0.082	**16.1**	1.086 ± 0.078	1.123 ± 0.110	**17.3**
TB-B	0.909 ± 0.069	0.753 ± 0.101	17.4	1.191 ± 0.112	1.258 ± 0.184	18.6
TB-Sub	1.185 ± 0.106	1.397 ± 0.276	27.5	1.319 ± 0.085	1.961 ± 0.171	25.2
TB-TS	1.164 ± 0.222	1.255 ± 0.410	21.3	1.390 ± 0.161	1.781 ± 0.295	21.2
RL-U	1.082 ± 0.060	1.287 ± 0.105	17.4	1.180 ± 0.049	1.455 ± 0.111	18.5
RL-B	$\mathbf{0.647 \pm 0.111}$	$\mathbf{0.440 \pm 0.132}$	30.2	1.203 ± 0.330	1.304 ± 0.723	34.4
RL-T	0.654 ± 0.069	0.456 ± 0.084	49.0	$\mathbf{0.838 \pm 0.105}$	$\mathbf{0.591 \pm 0.140}$	59.3
RL-G	0.670 ± 0.095	0.527 ± 0.198	31.8	0.888 ± 0.068	0.666 ± 0.091	36.4

Table 2. Converged metric values of different methods for the modeling of $64 \times 64 \times 64$ and $32 \times 32 \times 32 \times 32$ grids. Training time costs are provided in minutes.

	SIX6			
Method	$Acc \uparrow (\times 10^2)$	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Number of modes↑
DB-U	86.26 ± 1.20	1.889 ± 0.021	3.261 ± 0.052	298.24 ± 6.84
DB-B	86.26 ± 1.51	1.886 ± 0.020	3.263 ± 0.064	302.00 ± 4.97
TB-U	93.88 ± 1.13	$\mathbf{1.767 \pm 0.015}$	$\mathbf{2.915 \pm 0.025}$	317.92 ± 2.74
TB-B	90.94 ± 1.46	1.998 ± 0.055	3.542 ± 0.146	295.36 ± 4.69
TB-Sub	88.57 ± 0.91	1.893 ± 0.021	3.261 ± 0.056	305.26 ± 3.22
TB-TS	88.66 ± 1.04	1.887 ± 0.029	3.241 ± 0.050	302.58 ± 1.27
RL-U	94.30 ± 1.22	1.779 ± 0.018	2.956 ± 0.025	318.84 ± 3.08
RL-B	94.68 ± 1.62	1.786 ± 0.019	2.967 ± 0.036	319.14 ± 1.96
RL-T	93.40 ± 1.01	1.832 ± 0.022	3.174 ± 0.049	$\mathbf{321.74 \pm 1.72}$
RL-G	$\mathbf{94.70 \pm 1.65}$	1.782 ± 0.020	2.951 ± 0.025	318.96 ± 3.03

Table 3. Converged metric values of different methods for the SIX6 datasets.

	QM9			
Method	$Acc \uparrow (\times 10^2)$	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Number of modes↑
DB-U	96.30 ± 1.07	1.889 ± 0.021	3.261 ± 0.052	780.2 ± 4.32
DB-B	96.93 ± 1.07	1.886 ± 0.020	3.263 ± 0.064	780.0 ± 6.04
TB-U	98.65 ± 1.10	$\mathbf{1.767 \pm 0.015}$	$\mathbf{2.915 \pm 0.025}$	788.8 ± 2.78
TB-B	98.36 ± 1.08	1.998 ± 0.055	3.542 ± 0.146	781.4 ± 3.72
TB-Sub	97.00 ± 1.17	1.893 ± 0.021	3.261 ± 0.056	781.8 ± 4.87
TB-TS	96.83 ± 0.76	1.887 ± 0.029	3.241 ± 0.050	780.6 ± 4.16
RL-U	98.79 ± 1.08	1.779 ± 0.018	2.956 ± 0.025	787.0 ± 5.29
RL-B	$\mathbf{99.10 \pm 0.90}$	1.786 ± 0.019	2.967 ± 0.036	792.8 ± 2.95
RL-T	98.74 ± 1.27	1.832 ± 0.022	3.174 ± 0.049	$\mathbf{798.2 \pm 0.84}$
RL-G	90.04 ± 1.20	1.782 ± 0.020	2.951 ± 0.025	794.0 ± 2.16

Table 4. Converged metric values of different methods for the QM9 datasets.

	PHO4		sEH	
Method	$Acc \uparrow (\times 10^2)$	Number of modes↑	$Acc \uparrow$	Number of modes↑
DB-U	76.08 ± 0.23	3957.33 ± 18.61	88.45 ± 0.61	23502.97 ± 260.03
DB-B	76.61 ± 0.32	3975.00 ± 64.00	88.58 ± 0.81	23454.68 ± 422.14
TB-U	77.47 ± 0.29	3993.33 ± 9.87	90.49 ± 0.48	24250.63 ± 114.85
TB-B	77.00 ± 0.14	3901.33 ± 26.08	90.84 ± 0.85	23139.13 ± 634.65
TB-Sub	76.94 ± 0.24	3939.33 ± 18.61	89.81 ± 0.61	24652.47 ± 74.43
TB-TS	76.78 ± 0.14	3973.33 ± 38.68	89.57 ± 0.24	24491.93 ± 90.99
RL-U	77.31 ± 0.30	3984.67 ± 24.79	90.86 ± 0.42	25200.67 ± 24.61
RL-B	$\mathbf{77.55 \pm 0.25}$	4004.67 ± 15.01	91.36 ± 0.85	25454.03 ± 142.64
RL-T	76.81 ± 0.48	$\mathbf{4062.67 \pm 47.69}$	$\mathbf{93.98 \pm 0.64}$	$\mathbf{26530.30 \pm 142.27}$
RL-G	77.26 ± 0.55	4005.67 ± 12.50	91.28 ± 0.79	25368.07 ± 52.27

Table 5. Converged metric values of different methods for the PHO4 and sEH datasets

Method	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$	Method	$D_{TV} \downarrow (\times 10^{-1})$	$D_{JSD} \downarrow (\times 10^{-2})$
DB-U	1.346 ± 0.030	4.863 ± 0.420	DB-B	1.284 ± 0.027	7.533 ± 1.873
TB-U	0.898 ± 0.124	3.121 ± 0.332	TB-B	1.521 ± 0.281	13.301 ± 2.107
RL-U	0.831 ± 0.079	7.436 ± 1.508	RL-B	0.929 ± 0.078	6.114 ± 1.179
RL-T	$\mathbf{0.698 \pm 0.052}$	$\mathbf{1.890 \pm 0.307}$	RL-G	0.964 ± 0.174	5.712 ± 0.845

Table 6. Converged metric values of different methods for the BN structure learning experiment, where the ground-truth BN has 5 nodes and 5 edges.

E.6. Graphical representation of $P_F^\top$

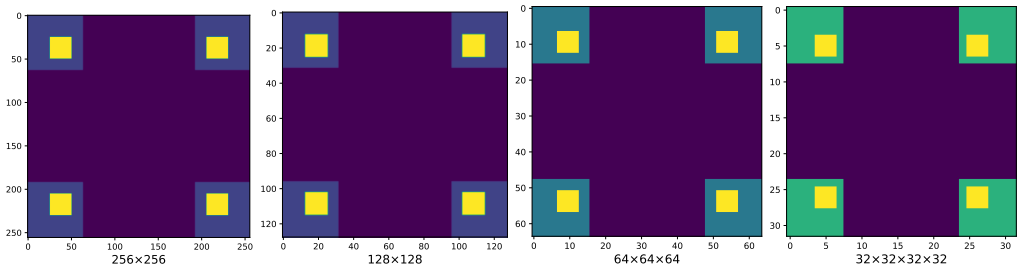

Figure 11. Graphical representation of P^* for different hyper-grids. For visualization easiness, the ground-truth marginal distributions of two dimensions are plotted for $64 \times 64 \times 64$ and $32 \times 32 \times 32 \times 32$ grids.

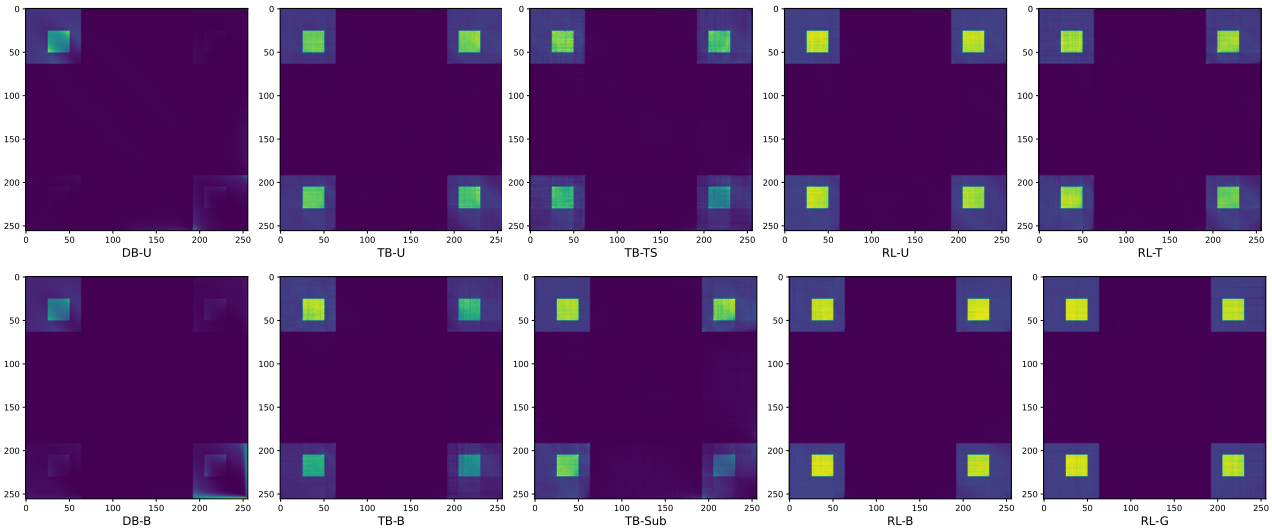

Figure 12. Graphical illustrations of $P_F^\top(x)$ averaged across 5 runs of corresponding training strategies for a 256×256 hyper-grid.

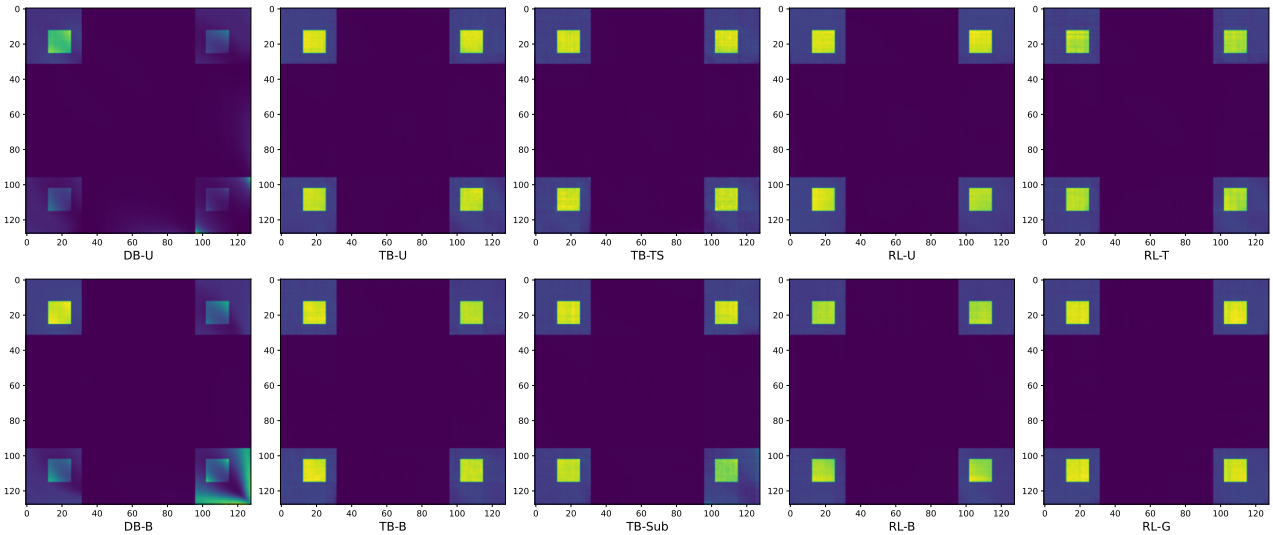

Figure 13. Graphical illustrations of $P_F^\top(x)$ averaged across 5 runs of corresponding training strategies for a 128×128 hyper-grid.

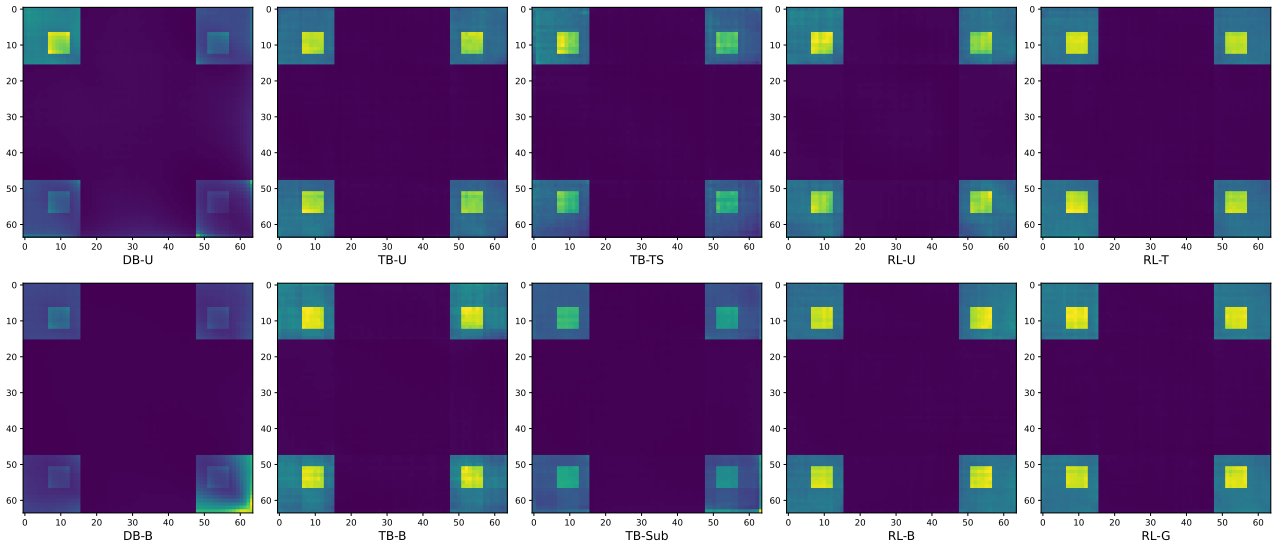

Figure 14. Graphical illustrations of $P_F^{\top}(x)$ averaged across 5 runs of corresponding training strategies for a $64 \times 64 \times 64$ hyper-grid. For visualization easiness, only the marginals of two dimensions are plotted.

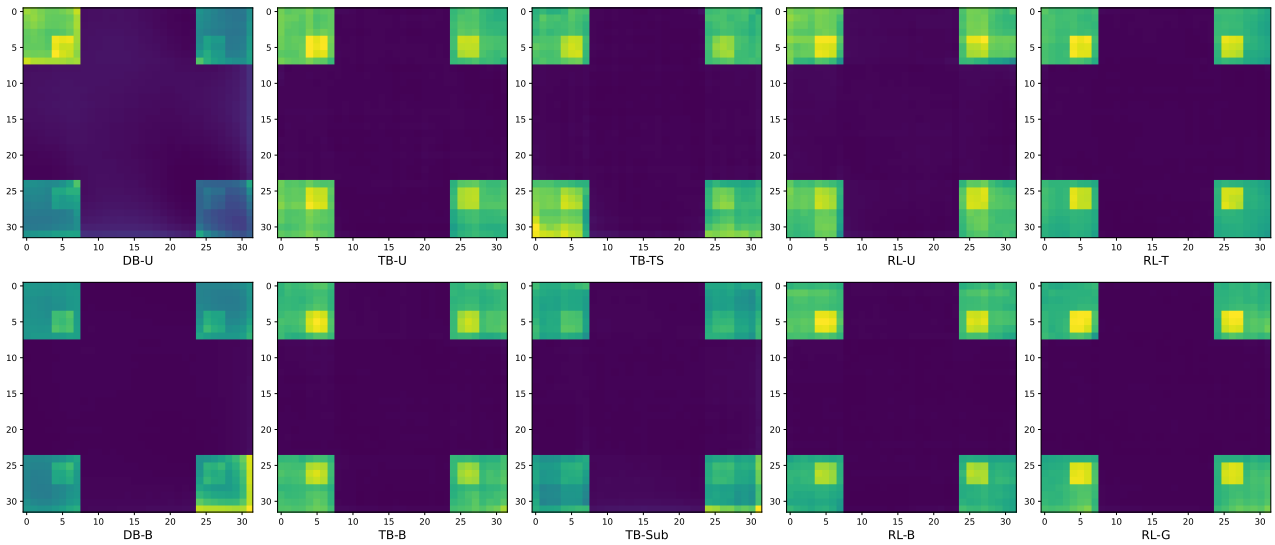

Figure 15. Graphical illustrations of $P_F^{\top}(x)$ averaged across 5 runs of corresponding training strategies for a $32 \times 32 \times 32 \times 32$ hyper-grid. For visualization easiness, only the marginals of 2 dimensions are plotted.

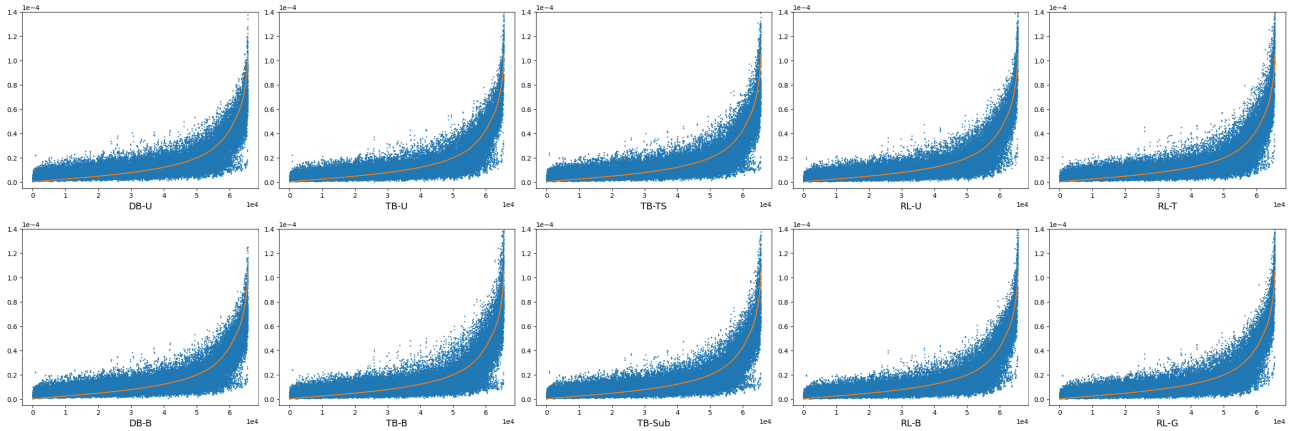

Figure 16. In each plot, the orange line represents the $P^*(x)$ of all sequences in the SIX6 dataset, with its values plotted in ascending order. The blue dots represent corresponding values of $P_F^\top(x)$, averaged over five runs of the corresponding training strategy.

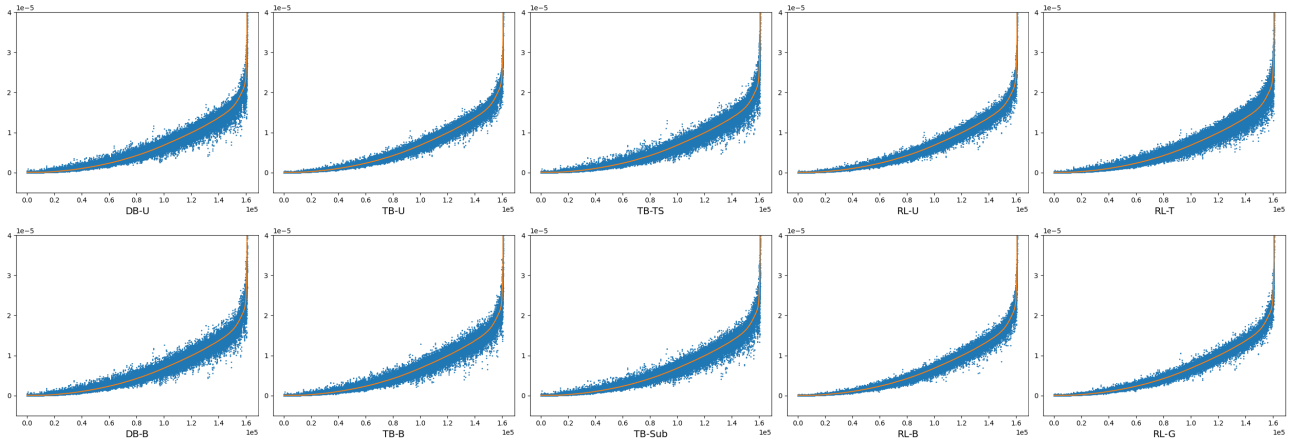

Figure 17. In each plot, the orange line represents the $P^*(x)$ of all sequences in the QM9 dataset, with its values plotted in ascending order. The blue dots represent corresponding values of $P_F^\top(x)$, averaged over five runs of the corresponding training strategy.

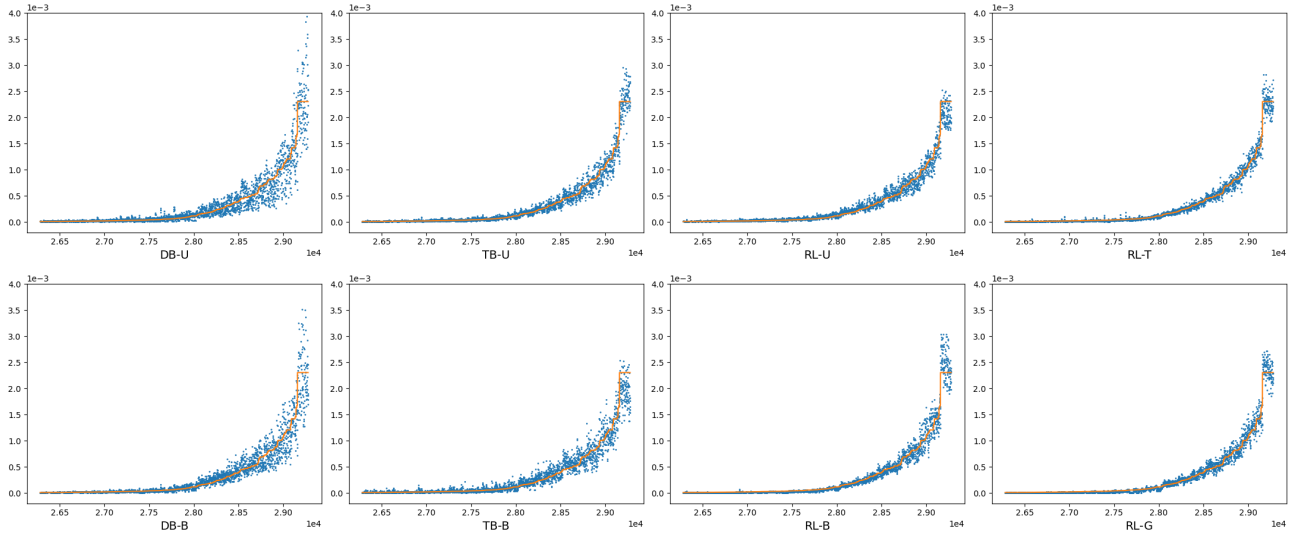

Figure 18. In each plot, the orange line represents the $P^*(x)$ of all BN structures, with its values plotted in ascending order. The blue dots represent corresponding values of $P_F^\top(x)$, averaged over five runs of the corresponding training strategy. Only the ground-truth and corresponding learned values for the top 3000 structures are plotted as the remaining values are nearly zero.