EVALUATING GFLOWNET FROM PARTIAL EPISODES FOR STABLE AND FLEXIBLE POLICY-BASED TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Flow Networks (GFlowNets) were developed to learn policies for efficiently sampling combinatorial candidates by interpreting their generative processes as trajectories in directed acyclic graphs. In the value-based training workflow, the objective is to enforce the balance over partial episodes between the flows of the learned policy and the estimated flows of the desired policy, implicitly encouraging policy divergence minimization. The policy-based strategy alternates between estimating that divergence and updating the policy, but reliable estimation of the divergence under directed acyclic graphs remains a major challenge. This work bridges the two perspectives by showing that flow balance also yields a principled policy evaluator that measures the policy divergence, and an evaluation balance objective over partial episodes is proposed for learning the evaluator. As demonstrated on both synthetic and real-world tasks, the flow balance condition not only strengthens the reliability of policy-based training but also broadens its flexibility by seamlessly supporting parameterized backward policies and enabling the integration of offline data-collection techniques.

1 INTRODUCTION

Generative Flow Networks (GFlowNets) are generative models on combinatorial space $\mathcal{X}$, such as graphs formed by organizing nodes and edges in a particular way, or strings composed of alphabets in a specific ordering. GFlowNets aim at sampling $x \in \mathcal{X}$ with probability $\propto R(x)$ where $R(x)$ is a non-negative score function. The task is challenging as $|\mathcal{X}|$ can be too large to compute the normalization constant $Z^* := \sum_{x \in \mathcal{X}} R(x)$ and the distribution modes can be highly isolated due to the combinatorial nature for efficient exploration. In GFlowNets (Bengio et al., 2021; 2023), generating or sampling $x \in \mathcal{X}$ is decomposed into incremental trajectories (episodes) that start from a null state, pass through intermediate states, and end at x as the desired terminating state. These trajectories $\tau \in \mathcal{T}$ can be viewed as the paths along a Directed Acyclic Graph (DAG) with state $s \in \mathcal{S}$ and edge $(s \to s') \in \mathcal{E}$. Positive measures (unnormalized probability) of trajectories are viewed as the amount of *flows* along the DAG, and $R(x)$ is the total flow of trajectories ending at x, so that sampled trajectories will end at x with the probability $\propto R(x)$.

The core of the GFlowNet training problem can be understood as minimizing the discrepancy of forward trajectory distribution $P_F(\tau)$ induced by forward policy $\pi_F(s'|s)$ towards backward trajectory distribution $P_B(\tau) := P_B(\tau|x)R(x)/Z^*$ induced by a given backward policy $\pi_B(s|s')$ and $R(x)$ (Bengio et al., 2023; Malkin et al., 2022b). This is motivated by the fact that in real-world applications, sequential generation must be done in a forward manner. Besides, the marginalization $P^*(x) = \sum_{\tau|x} P_B(\tau)$ trivially holds for all $x \in \mathcal{X}$, so any backward policy can be used to define a target backward trajectory distribution. Since evaluating Z^* and thereby the normalized distribution $P_B(\tau)$ is considered intractable, directly optimizing some distributional divergence of $P_F(\tau)$ and $P_B(\tau)$ is not feasible. To circumvent this, value-based methods reformulate the problem of distributional matching as a flow-matching problem and leverage the balance conditions of flow values to derive training objectives. The basic condition $\mathbb{E}_{P_{\mathcal{D}}(s)}[\log \pi_F(s'|s)F(s)] = \mathbb{E}_{P_{\mathcal{D}}(s)}[\log \pi_B(s|s')F(s')]$ means that the edge flow value $F(s \to s') = \pi_F(s'|s)F(s)$ matches the target edge flow value in the reverse direction, where $P_{\mathcal{D}}(s)$ is the marginal state distribution induced by the data collection policy $\pi_{\mathcal{D}}(s'|s)$. This basic condition leads to various training objectives that implicitly encourage the alignment between forward and backward distributions. These objectives range from edge-wise formulations to subtrajectory-wise variants, providing diverse estimation of target flows and quantification of flow imbalances (Bengio et al., 2023; Madan et al., 2023; Malkin et al., 2022a). Alternatively, policy-based methods (Malkin et al., 2022b; Niu et al., 2024; Zimmermann et al., 2022) introduce a state evaluation function $V(s)$ of the forward policy $\pi_F(\cdot|s)$, which approximates the Kullback-Leibler (KL) divergence between the distribution of forward subtrajectories (partial episodes) starting from state s and that of backward subtrajectories ending at s. Then $V(s)$ is used to update the forward policy $\pi_F(\cdot|s)$. While optimizing policy updating based on $V(s)$ has been well-studied, how to reliably estimate $V(s)$ remains an open problem.

In light of the success of value-based methods that leverage flow balance conditions and the analogy between $F(s)$ as the total amount of trajectory flows passing state s, and $V(s)$ as a measurement of distributional divergence at state s, it is worthwhile to investigate the relationship between the two quantities. It is known that the optimal forward policy π_F and state flow function $F(s)$ can be uniquely determined by the flow balance condition $\mathbb{E}_{\pi_F}[\log F(s)\pi_F(s'|s)] = \mathbb{E}_{\pi_F}[F(s')\pi_B(s|s')]$. We find that, for an arbitrary fixed π_F, the solution to this condition coincides with the ground-truth evaluation function $V(s)$ of π_F. This paves the way to derive balance-based objectives to estimate $V(s)$ reliably. Our contributions are as follows:

- We establish the connection between balance conditions with respect to (w.r.t.) the state flow function F and the evaluation function V. Fixing π_F, the expected balancing conditions of $\log F$ directly lead to a sufficient condition for V, which we call the Subtrajectory Evaluation Balance (Sub-EB) condition.
- We introduce the Sub-EB objective (Sub-EB) for reliably estimating the evaluation function V, where subtrajectories (partial episodes) serve as the basic unit of balance.
- Experimental results on both simulated and real-world datasets, including hypergrid modeling, biological and molecular sequence design, and Bayesian network structure learning, have demonstrated the effectiveness and reliability of Sub-EB for policy evaluation.

2 PRELIMINARIES

We restrict DAGs of GFlowNets to be *graded* as any DAG can be equivalently converted to be graded by adding dummy non-terminating states (Malkin et al., 2022b). In a DAG $\mathcal{G} := (\mathcal{S}, \mathcal{E})$, element $s \in \mathcal{S}$ denotes a state, and element $e \in \mathcal{E}(\subseteq \mathcal{S} \times \mathcal{S})$ denotes a directed edge. We define the index set of time horizons as $[H] := \{0, \dots, H\}$. Being *acyclic* means the state space $\mathcal{S}$ can be partitioned into disjoint subspaces: $\mathcal{S}_0, \dots, \mathcal{S}_{H+1}$, where each element of $\mathcal{S}_h$ is denoted as s_h for $h \in [H+1]$. Being *graded* further implies that actions are only allowed from $\mathcal{S}_h$ to $\mathcal{S}_{h+1}$. For any $s \in \mathcal{S}$, we denote its parent set by $Pa(s) := \{s' | (s' \to s) \in \mathcal{E}\}$ and its child set by $Ch(s) := \{s' | (s \to s') \in \mathcal{E}\}$. 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 $S_{H+1} = \{s_f\}$. The terminating state set, $S_H := \mathcal{X}$ is associated with a score function $R : \mathcal{X} \to \mathbb{R}^+$. Furthermore, the complete trajectory set is defined as $\mathcal{T} := \{\tau = (s_0 \to \cdots \to s_f) | \forall (s \to s') \in \tau : (s \to s') \in \mathcal{E}\}$, $\tau_{i:j}$ denotes a subtrajectory (partial episode) that starts at some state in $\mathcal{S}_i$ and ends at some state $\mathcal{S}_j$ for $i < j \in [H+1]$, $\tau_{i:}$ denotes a subtrajectory from s_i to s_f and $\tau_{:j}$ denotes a subtrajectory from s_0 to s_j. An exemplary DAG and its *graded* version is shown in Fig. 3.

2.1 GFLOWNET TRAINING

GFlowNets aim at sampling with probability $R(x)/Z^*$, where computing Z^* is considered intractable. To achieve this, GFlowNets define a *Markovian* positive measure $F(\tau) : \mathcal{T} \to \mathbb{R}^+$, termed as (trajectory) *flow* (Bengio et al., 2023), so that for any event E and E', $F(E) := \sum_{\tau \in E} F(\tau)$. Then the total flow is $Z := F(s_0) = \sum_x F(x)$. Consequently, $P(E) := F(E)/Z$ and $P(E|E') := \frac{F(E \cap E')}{F(E')}$. In particular, for any $(s \to s') \in \mathcal{E}$, the edge flow and state flow are $F(s \to s') = \sum_{\tau \ni (s \to s')} F(\tau)$ and $F(s) = \sum_{\tau \ni s} F(\tau)$. These induces to the forward and backward policies, $\pi(s'|s) := P(s \to s'|s) = \frac{F(s_h \to s_{h+1})}{F(s_h)}$ and $\pi(s|s') := P(s \to s'|s') = \frac{F(s_h \to s_{h+1})}{F(s_{h+1})}$. Being *Markovian* implies that $P(\tau_{i,j}|s_i) = \prod_{h=i}^{j} \pi(s_{h+1}|s_h)$, $P(\tau_{i,j}|s_j) = \prod_{h=0}^{H} \pi(s_h|s_{h+1})$, and $P(\tau) = P(\tau|s_0) = P(\tau|s_f)$. Consequently, the goal of GFlowNet training is to learn a flow F that matches any desired flow F^* satisfying $F^*(x) = R(x)$. Given $\pi_F(s'|s)$ and $F(s)$ of one flow $F(\tau)$, and $\pi_B(s|s')$ of the desired flow $F^*(\tau) := R(x)P_B(\tau|x)$, a necessary and sufficient condition for achieving this goal is called the Sub-Trajectory Balance (Sub-TB) condition (Malkin et al., 2022a; Madan et al., 2023), which can be written as:

$$\mathbb{E}_{P_\mathcal{D}(\tau_{i:j})}\left[\log\left(F(s_i)P_F(\tau_{i:j}|s_i)\right)\right] = \mathbb{E}_{P_\mathcal{D}(\tau_{i:j})}\left[\log\left(F(s_j)P_B(\tau_{i:j}|s_j)\right)\right] \tag{1}$$

for any $i < j \in [H+1]$, where $F(s_f)P_B(x|s_f) := R(x)$ for $x \in \mathcal{X}$, Here, $P_F(\tau_{i,j}|s_i)$, $P_B(\tau_{i,j}|s_j)$, and $P_\mathcal{D}(\tau_{i,j}) = \sum_{\tau \ni \tau_{i,j}} P_\mathcal{D}(\tau)$ are distributions induced by π_F, π_B and offline data-collection policy $\pi_\mathcal{D}$, respectively. It is assumed that $P_\mathcal{D}(\tau) > 0$ for any $\tau \in \mathcal{T}$ (Malkin et al., 2022b). Besides, backward policy π_B can be arbitrary, as long as $P_B(\tau) := P_B(\tau|x)R(x)/Z^* > 0$ for any $\tau \in \mathcal{T}$. This is because $R(x)/Z^* = \sum_{\tau|x} P_B(\tau)$ trivially holds for all $x \in \mathcal{X}$. The condition can be interpreted as the flow value of a subtrajectory should match the target flow value in the reverse direction, which represents an (approximated) desired flow value.

Leveraging the balance condition, value-based methods typically use the Sub-TB objective to optimize the parameterized policy $\pi_F(s'|s;\theta)$ and state-flow logarithm $\log F(s;\theta)$ toward their optimal solutions $\pi_F^*(s'|s) := \pi_B(s'|s)$ and $\log F^*(s)$. The objective is defined as follows:

$$\mathcal{L}_F := \mathbb{E}_{P_\mathcal{D}(\tau)}\Big[\sum_{\tau_{i:j}} w_{j-i}\delta_F(\tau_{i:j})\Big], \quad \delta_F(\tau_{i:j};\theta) := \left(\log\frac{P_F(\tau_{i:j}|s_i;\theta)\log F(s_i;\theta)}{P_B(\tau_{i:j}|s_j;\theta)\log F(s_j;\theta)}\right)^2, \tag{2}$$

where w_{j-i} denotes the non-zero weight coefficient for subtrajectories that consist of $j - i$ edges. For practical gradient-based optimization, $\mathcal{L}_F$ are approximated by $\widehat{\mathcal{L}}_F := \frac{1}{K}\sum_{\tau\in\mathcal{D}}[\sum_{\tau_{i:j}\in\tau} w_{j-i}\delta_F(\tau_{i:j})]$, where $\mathcal{D} := \{\tau^k : \tau^k \sim P_\mathcal{D}(\tau)\}_{k=1}^K$ is the set of samples.

2.2 POLICY-BASED TRAINING

Policy-based training methods for GFlowNets (Niu et al., 2024) resemble the policy gradient algorithm in Reinforcement Learning (RL) and aim to minimize the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(P_F(\tau;\theta)\|P_B(\tau))$ as in traditional variational approaches (Malkin et al., 2022b; Zimmermann et al., 2021; 2022). The method follows the **actor-critic** framework Agarwal et al. (2021).

Critic In each training round, the critic (evaluation function) $V^\dagger$ of actor π_F is first computed to capture the policy gaps in terms of KL divergences over subtrajectories, which is defined as $\forall h \in [H]$[1]:

$$V^\dagger(s_h;\theta) := \mathbb{E}_{P_F(\tau_{h:}|s_h;\theta)}\left[\sum_{i=h}^{H} R(s_i,s_{i+1};\theta)\right] = \log F^*(s_h) - D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h)\|P_B(\tau_{h:}|s_h)) \tag{3}$$

where $R(s_i,s_{i+1};\theta) := \log\frac{\pi_F(s_{i+1}|s_i;\theta)}{\widetilde{\pi}_B(s_i|s_{i+1})}$, and $\widetilde{\pi}_B = \pi_B$ expect that $\widetilde{\pi}_B(x|s_f) := R(x)$. The second equality can be easily verified Niu et al. (2024), and its explicit derivation is also provided in (21) in the Appendix.

Actor To optimize π_F to minimize $D_{\mathrm{KL}}(P_F(\tau;\theta)\|P_B(\tau))$, it is noted that $\nabla_\theta V^\dagger(s_0;\theta) = \nabla_\theta(P_F(\tau;\theta)\|P_B(\tau))$ since $\log F^*(s_0)$ is a constant. Further applying the policy gradient theorems in RL (Agarwal et al., 2019), the gradients of the KL divergence w.r.t π_F can be simplified and expressed as the following expectation:

$$\nabla_\theta V^\dagger(s_0;\theta) := \mathbb{E}_{P_F(\tau)}\left[\sum_{h=0}^{H} A^\gamma(s_h,s_{h+1})\nabla_\theta\log\pi_F(s_{h+1}|s_h;\theta)\right], \tag{4}$$

where $A^\gamma(s_h,s_{h+1}) = \sum_{i=h}^{H}\gamma^{i-h}(R_F(s_i\to s_{i+1}) + V(s_{i+1}) - V(s_i))$. When increasing the hyperparameter γ from 0 to 1, $A^\gamma(s_h,s_{h+1})$ evolves from $R_F(s_h\to s_{h+1}) + V(s_{h+1}) - V(s_h)$ to $\sum_{i=h}^{H} R_F(s_i\to s_{i+1}) - V(s_h)$. This enables the variance-bias trade-off during stochastic gradient descent (Schulman et al., 2016; 2017). Given π_F, analytical computation of its $V^\dagger$ is usually not feasible as $|\mathcal{T}|$ can be enormous, so $V^\dagger$ is directly modeled by an evaluation function V with learnable parameters. We will discuss the learning objective of V in the next section.

[1]Niu et al. (2024) consider an additional total flow estimator $Z(\theta)$ to scale down the magnitude of $V^\dagger$. For notion compactness, we defer the discussion of integrating Z into our method to Appendix A.4.

3 Subtrajectory Evaluation Balance

As discussed in the previous sections, the true evaluation function $V^\dagger$ plays a central role in policy-based methods. In this section, we first introduce the Sub-EB condition that characterizes the relationship between π_F and a parameterized evaluation V, which closely resembles the Sub-TB conditions between π_F and F. We then present the Sub-EB objective for learning V. Further, we extend these results with corresponding conditions and objectives for backward policies and backward evaluation functions.

3.1 The balance condition and objective for forward policies

Given a forward policy π_F, the Subtrajectory Evaluation Balance (Sub-EB) condition for the associated evaluation function V can be written as:

$$\mathbb{E}_{P_F(\tau_{i:j})}\left[\log\left(P_F(\tau_{i:j}|s_i)\exp\dot{V}(s_i)\right)\right] = \mathbb{E}_{P_F(\tau_{i:j})}\left[\log\left(P_B(\tau_{i:j}|s_j)\exp\dot{V}(s_j)\right)\right] \tag{5}$$

for any $i < j \in [H+1]$, where $\dot{V} := -V$ and $P_B(x|s_f)\exp\dot{V}(s_f) := R(x)$ for $x \in \mathcal{X}$. It should be noted that the expectation is taken w.r.t. the subtrajectory distribution induced by π_F while it is taken w.r.t. the subtrajectory distribution induced by $\pi_{\mathcal{D}}$ in the Sub-TB condition.

Theorem 3.1. *Suppose V is an evaluation function over $\mathcal{S}$ and F^* is the desired flow. Given a forward policy π_F,*

$$\forall h \in [H] : -V(s_h) = \log F^*(s_h) - D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h)\|P_B(\tau_{h:}|s_h)), \tag{6}$$

if and only if V satisfies the Sub-EB condition (5).

The corresponding proof can be found in Appendix A.1.

Theorem 3.2. *Suppose F is a state flow function over $\mathcal{S}$ and π_F is a forward policy. Then,*

$$\forall h \in [H] : \log F(s_h) = \log F^*(s_h) - D_{\mathrm{KL}}(P_F^*(\tau_{h:}|s_h)\|P_B(\tau_{h:}|s_h)) \tag{7}$$

and π_F is equal to π_F^* if and only if F and π_F satisfy the Sub-TB condition (1).

The corresponding proofs can be found in Appendix A.2. Here, $\pi_F^*(s'|s) := \frac{F^*(s'\to s)}{F^*(s)}$. While the sufficiency and necessity of the Sub-TB condition have been studied in prior work (Bengio et al., 2023; Malkin et al., 2022a), Theorem 3.2 offers an alternative perspective that more clearly elucidates the connection between the flow function and the evaluation function. It should be noted that the minimum of the KL term above is zero as both $P_F(\tau)$ and $P_B(\tau)$ are Markovian. By Proposition 23 in Bengio et al. (2023), the trajectory flow $F(\tau) = P_F(\tau)Z$ that achieves the zero KL term is unique.

Leveraging the balancing condition, we define the Sub-EB objective for optimizing a parameterized evaluation function $V(\cdot\,;\phi)$ as:

$$\mathcal{L}_V(\phi) := \mathbb{E}_{P_F(\tau)}\Big[\sum_{\tau_{i:j}} w_{j-i}\delta_V(\tau_{i:j};\phi)\Big],\ \delta_V(\tau_{i:j};\phi) = \left(\log\frac{P_F(\tau_{i:j}|s_i)\exp\dot{V}(s_i;\phi)}{P_B(\tau_{i:j}|s_j;\phi)\exp\dot{V}(s_j;\phi)}\right)^2, \tag{8}$$

where w_{j-i} is a weight constant for sub-trajectories that consist of $j-i$ edges. While the traditional λ-Temporal-Difference (TD) objective (54) detailed in Appendix A.5 (Niu et al., 2024; Schulman et al., 2016) focuses on learning $V(s_h;\phi)$ only from events starting at step h and edge-wise mismatches $\delta_V(s \to s')$, the Sub-EB objective incorporates events both before and after h and leverages subtrajectory-wise mismatches, yielding more balanced learning of $V(s_h;\phi)$. Moreover, Sub-EB allows freely weighting schemes, whereas the scheme of λ-TD is restricted to the λ-decay form. A detailed comparison between our Sub-EB and λ-TD objectives is provided in Appendix A.5.

It should be noted that the Sub-EB objective is specifically designed for learning V that approximates $V^\dagger$ of the current π_F. During each optimization iteration, its gradient w.r.t. ϕ is computed to update $V(\cdot\,;\phi)$, while parameter θ is frozen. In contrast, the Sub-TB objective 2 is used to jointly update $\pi_F(\cdot|\cdot\,;\theta)$ and $\log F(\cdot\,;\theta)$.[2] We summarize the workflow of our policy-based method for GFlowNet training in Algorithm 1. Here, in analogy to $\mathcal{L}_F$ and $\widehat{\mathcal{L}}_F$, we use $\widehat{\mathcal{L}}_V$ and $\widehat{\nabla}_\theta V^\dagger(s_0;\theta)$ to denote the approximated versions based on sampled trajectories (and V).

[2]Here, θ and ϕ are introduced as separate parameter sets. θ corresponds to functions updated jointly with π_F, while ϕ corresponds to functions that are not.

3.2 Parameterized backward policy

In this and the following sections, we further discuss two key advantages, introduced by the Sub-EB condition, that improve the flexibility of policy-based training. The λ-TD objective (54) requires π_B to remain fixed throughout optimization, as V^λ is treated as constant w.r.t. ϕ. To address this limitation, Niu et al. (2024) proposed a two-phase algorithm for addressing this issue. Each training iteration includes two phases. In the forward phase, we sample $\mathcal{D} \sim P_F(\tau)$, update V based on $\mathcal{D}$, and update π_F based on V and $\mathcal{D}$. In the backward phase, we sample $\mathcal{D}' \sim P_B(\tau|x)P_F(x)$, update the evaluation function W that approximates the true evaluation function $W^\dagger$ w.r.t. π_B, and update π_B based on W. The definitions of $W^\dagger$ and W are detailed in Section 3.3. In parallel, Gritsaev et al.; Jang et al. (2024) adopted an additional objective for π_B in the forward phase. When applied to the policy-based framework, their approach first samples a batch $\mathcal{D} \sim P_F(\tau)$ and updates π_B by maximizing its log-likelihood, $\sum_{\tau\in\mathcal{D}} \log P_B(\tau)$. Then, with π_B held fixed, they update the value function V and subsequently optimize π_F based on the updated V, using the same batch $\mathcal{D}$. Compared to all these algorithms, both the Sub-TB (2) and Sub-EB (8) objectives allow for updating parameterized π_B without introducing a separate backward phase or an additional objective. To be more specific, π_B is jointly updated with π_F and $F^{\log}$ for the Sub-TB objective, and π_B is jointly updated with V for the Sub-EB objective. This leads to a more streamlined and efficient training process while enabling the backward policy to adapt dynamically during optimization.

3.3 Offline policy-based training

Both the single-phase method and the two-phase method by Niu et al. (2024) operate in an **online** manner, meaning that we can not use a policy $\pi_{\mathcal{D}}$, different from the current forward policy π_F, during training. To overcome this limitation, we introduce an **offline** policy-based method made possible by the flexibility of the Sub-EB objective. Supposing π_F is fixed, we can define the evaluation function $W^\dagger$ of π_B as $W^\dagger(s_0) := -\log F(s_0)$ and $\forall h \in [H] \setminus \{0\}$:

$$W^\dagger(s_h;\phi) := \mathbb{E}_{P_B(\tau_{:h}|s_h;\phi)}\left[\sum_{i=0}^{h-1} R(s_{i+1}, s_i;\phi)\right] = D_{\mathrm{KL}}(P_B(\tau_{:h}|s_h)\|P_F(\tau_{:h}|s_h)) - \log F(s_h) \quad (9)$$

where $R(s_{i+1}, s_i;\phi) := \log \frac{\pi_B(s_i|s_{i+1};\phi)}{\pi_F(s_{i+1}|s_i)}$, It can be easily verified that. Since $\nabla_\phi \log F(x) = 0$ and $\nabla_\phi \mathbb{E}_{P_{\mathcal{D}}(x)}[W^\dagger(x;\phi)] = \nabla_\phi \mathbb{E}_{P_{\mathcal{D}(x)}}[D_{\mathrm{KL}}(P_B(\tau|x;\phi)\|P_F(\tau|x))]$, minimizing $\mathbb{E}_{P_{\mathcal{D}}}[W^\dagger(x)]$ can be a surrogate to minimizing the expected KL divergence. In analogy to the forward case, we use a parameterized evaluation function W to approximate $W^\dagger$. Then, given a backward policy π_B, the backward Sub-EB condition for W can be written as:

$$\mathbb{E}_{P_B^{\mathcal{D}}(\tau_{i:j})}\left[\log\left(P_F(\tau_{i:j}|s_i)\exp \dot{W}(s_i)\right)\right] = \mathbb{E}_{P_B^{\mathcal{D}}(\tau_{i:j})}\left[\log\left(P_B(\tau_{i:j}|s_j)\exp \dot{W}(s_j)\right)\right] \quad (10)$$

for any $i < j \in [H+1]$, where $\dot{W} := -W$, $P_B(x|s_f)\exp \dot{W}(s_f) := R(x)$ for $x \in \mathcal{X}$, and $P_B^{\mathcal{D}}(\tau_{i,j})$ denotes the marginal distribution induced by $P_B^{\mathcal{D}}(\tau) := P_B(\tau|x)P_{\mathcal{D}}(x)$.

Theorem 3.3. *Suppose W is an evaluation function over $\mathcal{S}\backslash\{s_f\}$. Given a backward policy π_B,*

$$\forall h \in [H-1] : -W(s_{h+1}) = \log F(s_{h+1}) - D_{\mathrm{KL}}(P_B(\tau_{:h+1}|s_{h+1})\|P_F(\tau_{:h+1}|s_{h+1})), \quad (11)$$

and $-W(x) = \log R(x)$ if and only if W satisfies the backward Sub-EB condition (10).

The corresponding proof can be found in Appendix A.3. As shown in (45) there, the right-hand side of (11) is equal to $W^\dagger$ defined in (9). When π_B and π_F are at their optima, the KL term in the expression of W is zero. Consequently, $F(x) = R(x)$ and $F(s_h) = F^*(s_h)$ for any $h \in [H]$ thereby fulfilling the goal of GFlowNet training. Based on the backward Sub-EB condition, we present the backward Sub-EB objective for $W(\cdot;\theta)$ as follows:

$$\mathcal{L}_W := \mathbb{E}_{P_B^{\mathcal{D}}(\tau)}\Big[\sum_{\tau_{i:j}} w_{j-i}\delta_W(\tau_{i:j})\Big],\ \delta_W(\tau_{i:j};\theta) = \left(\log \frac{P_F(\tau_{i:j}|s_i;\theta)\exp \dot{W}(s_i;\theta)}{P_B(\tau_{i:j}|s_j)\exp \dot{W}(s_j;\theta)}\right)^2. \quad (12)$$

The workflow of our offline policy-based method for GFlowNet training is presented in Algorithm 2, where we use $\widehat{\mathcal{L}}_W$ and $\widehat{\nabla}_\theta W^\dagger(s_0;\theta)$ to denote the approximated $\mathcal{L}_W$ and $\nabla_\theta W^\dagger(s_0;\theta)$ based on sampled trajectories (and W).

Algorithm 1 Online Policy-based Workflow

```
Require: $\pi_F(\cdot|\cdot;\theta)$, $\pi_B(\cdot|\cdot;\phi)$, $V(\cdot;\phi)$, batch size $K$, number of total iterations $N$
  for $n = 1, \ldots, N$ do
    $\mathcal{D} \leftarrow \{\tau^k | \tau^k \sim P_F(\tau)\}_{k=1}^K$
    Based on $\mathcal{D}$, update $\phi$ by $\nabla_\phi \widehat{\mathcal{L}}_V(\phi)$.
    Based on $\mathcal{D}$ and $V$, update $\theta$ by $\widehat{\nabla}_\theta V^\dagger(s_0;\theta)$
  end for
  return $\pi_F(\cdot|\cdot;\theta)$, $\pi_B(\cdot|\cdot;\phi)$, $V(\cdot;\phi)$
```

Algorithm 2 Offline Policy-based Workflow

```
Require: $\pi_B(\cdot|\cdot;\phi)$, $\pi_F(\cdot|\cdot;\theta)$, $W(\cdot;\theta)$, $\pi_{\mathcal{D}}$, batch size $K$, number of total iterations $N$
  for $n = 1, \ldots, N$ do
    $\mathcal{D}^\top \leftarrow \{x^k | x^k \in \tau^k, \tau^k \sim P_{\mathcal{D}}(\tau)\}_{k=1}^K$
    $\mathcal{D} \leftarrow \{\tau^k | x^k \in \mathcal{D}^\top, \tau^k \sim P_B(\tau|x^k)\}_{k=1}^K$
    Based on $\mathcal{D}$, update $\theta$ by $\nabla_\theta \widehat{\mathcal{L}}_W(\theta)$.
    Based on $\mathcal{D}$ and $W$, update $\phi$ by $\widehat{\nabla}_\phi \mathbb{E}_{P_{\mathcal{D}}(x)}[W^\dagger(x;\phi)]$
  end for
  return $\pi_B(\cdot|\cdot;\phi)$, $\pi_F(\cdot|\cdot;\theta)$, $W(\cdot;\theta)$
```

4 Related works

Value-based GFlowNet Training Existing works on value-based GFlowNet training can be categorized into two directions. The first one focuses on designing training objectives to characterize target flow values, thereby improving the estimation of flow imbalance. For example, the Detailed Balance (DB) objective (Bengio et al., 2021; 2023) aims at minimizing the mismatch between the logarithms of forward edge flow and backward target edge flow expressed as $F^{\log}(s) + \log \pi_F(s'|s)$ and $F^{\log}(s') + \log \pi_B(s|s')$, respectively. Malkin et al. (2022a) proposed the Trajectory Balance (TB) objective that optimizes the mismatch between the logarithms of forward trajectory flow and the backward trajectory flow. Sub-TB objective (Madan et al., 2023) generalizes both DB and TB by minimizing the flow logarithm mismatch of subtrajectories across varying lengths. DB and TB objectives are equivalent to the Sub-TB objective when non-zero weights are only assigned to edges or complete trajectories. Building on Sub-TB objects, various improvements are also proposed. Kim et al. (2023a) introduced temperature-conditional objectives, whose goal turns to make $P_F(x) \propto R^\beta(x)$ with positive scalar $\beta < 1$. Taking R to the exponent β reduces its sharpness, making it easier to be matched. As $P_F(x)$ does not match $R(x)$ in this case, this representation is specifically useful when the focus is solely on mode seeking. For all the objectives mentioned above, backward policy $\pi_B(s|s')$ can be chosen freely for any intermediate edges $(s \to s')$ and only $\pi(x|s_f)$ is fixed to $R(x)$. Target flow values of intermediate edges or subtrajectories do not directly reflect the ground-truth knowledge about the reward function $\mathcal{R}$. However, under the special cases that the covered object space of $\mathcal{R}$ can be extended from $\mathcal{X}$ to $\mathcal{S}$, Pan et al. (2023) and Jang et al. (2023) improved the formulation of the DB and Sub-TB objectives, propagating partial knowledge of $\mathcal{R}$ directly to intermediate edges. Due to the similarity of the Sub-TB objective and our Sub-EB objective, these improvements can also be easily adapted to Sub-EB, facilitating the estimation of the evaluation function V. *A key characteristic* of value-based methods is the data-collection policy $\pi_{\mathcal{D}}$ that can be off-policy, meaning it can differ from π_F. This flexibility leads to numerous efforts at designing $\pi_{\mathcal{D}}$ (Kim et al., 2023b; Rector-Brooks et al., 2023; Shen et al., 2023). The goal is to effectively identify edges that precede highly-rewarded terminating states (exploration) while allowing revisiting the edges already found to yield high reward (exploitation). The most widely used approach is α-greedy design that mixes π_F with a uniform policy by factor α. These efforts, however, fail to achieve *deep exploration*, which requires considering not only immediate information gain but also the long-term consequences of a transition (edge) in future learning (Osband et al., 2019). Although there have been many theoretical advances to efficient exploration design from an RL perspective (Azar et al., 2017; Jin et al., 2018; Ménard et al., 2021), their expensive computational cost approaches limit their applicability in practical problems.

Policy-based GFlowNet Training As mentioned above, designing a data-collection policy that is both computationally efficient and capable of *deep exploration* remains a significant challenge. Moreover, what is truly needed for training efficiency is identifying the edges of high flow-imbalance rather than the edges that lead to high terminating rewards. However, the flow imbalance is closely related to π_F and changes whenever π_F is updated. Empirical evidence shows that on-policy training, meaning $\pi_{\mathcal{D}}$ is equal to π_F can result in faster convergence under many conditions (Atanackovic & Bengio, 2024). Accordingly, policy-based GFlowNet methods (Malkin et al., 2022b; Niu et al., 2024; Zimmermann et al., 2022) conduct on-policy training, which typically corresponds to optimizing the KL divergence between $P_F(\tau)$ and the unnormalized distribution $P_B(\tau|x)R(x)(= P_B(\tau)Z^*)$, which has gradient equivalence to the divergence between $P_F(\tau)$ and

$P_B(\tau)$. Removing the need to design a data-collection policy, the main challenge of policy-based methods shifts to robust estimation of the divergence and its gradients, that is, balancing the trade-off between variance and bias of the estimators. Malkin et al. (2022b) and Silva et al. (2024) construct estimators empirically. During gradient-based optimization, these estimators are computed solely from the training data sampled in the current iteration. These estimators typically exhibit low bias but high variance. From the perspective of policy gradient algorithms in RL, Niu et al. (2024) proposed estimators on a parameterized evaluation function V, which enables leveraging sampled data from all previous iterations. These estimators generally have high bias but low variance. Combining empirical and parameterized approaches, Niu et al. (2024) introduced a tunable hyperparameter to control **bias-variance trade-off** explicitly, resulting in significant performance gains. As the effectiveness of this policy-based method critically depends on how V is learned, our paper is a subsequent work to address this key challenge.

As GFlowNet training is closely related to RL, we provide more detailed discussions on GFlowNet training from an RL perspective in Appendix A.6 and A.7 . Value-based methods follow the soft Q-learning framework Haarnoja et al. (2017), whereas policy-based methods, based on policy gradients, operates within the soft actor–critic framework Haarnoja et al. (2018).

5 Experiments

We compare the empirical policy-based method of Silva et al. (2024), which uses the Control Variate technique for variance reduction during gradient estimation, RL-like policy-based method with V estimated by the λ-TD objective in GFlowNet training (Niu et al., 2024) and V by our proposed Sub-EB objective, referred to as CV, RL and Sub-EB, respectively. Since the Sub-TB method (Madan et al., 2023) and the value-based RL method (Muchnausen DQN, denoted as Q-Much) Tiapkin et al. (2024) are closely related to Sub-EB, we also include them as representative baselines for value-based methods. To design the data collection policy $\pi_{\mathcal{D}}$ in the Sub-TB method, we follow a common choice, where $\pi_{\mathcal{D}}$ is equal to π_F with probability $(1-\alpha)$ and a uniform policy with probability α (Shen et al., 2023; Rector-Brooks et al., 2023). The common setups of π_B are a uniform policy or a parameterized policy. By default, we follow the first setup, as the parameterized policy does not carry ground-truth information about the reward function. For both Sub-EB and Sub-TB, the weight coefficient w_{j-i} for $i<j\in[H+1]$ is set to $\lambda^{j-i}/\sum_{i<j\in[H+1]}\lambda^{j-i}$ following Madan et al. (2023).

We choose total variation D_{TV} and Jensen–Shannon divergence D_{JSD} between $P_F(x)$ and $P^*(x)$ as the metrics for performance comparison between competing methods. Their definition is detailed in Appendix B. We have conducted three sets of experiments. The first set is conducted in simulated environments, referred to as 'Hypergrids'. The second set focuses on biological and molecular sequence design tasks using real-world datasets. The third set involves real-world applications of GFlowNet in Bayesian Network (BN) structure learning. More experimental details can be found in Appendix B. The implementation code is provided as supplementary material.

Hypergrids Hypergrid experiments are widely used for testing GFlowNet performance (Malkin et al., 2022b; Niu et al., 2024). Here, states are the coordinate tuples of an D-dimensional hypercubic grid with heights equal to H. The detailed description of the generative process is provided in Appendix B.1. We perform experiments on 256×256, $128\times128\times128$ and $64\times64\times64$ grids. We use dynamic programming (Malkin et al., 2022a) to explicitly compute $P_F(x)$ for learned π_F, and compute the exact D_{TV} and D_{JSD} between $P_F(x)$ and $P^*(x)$. Experimental results are depicted in Fig. 1 for D_{TV} and Fig. 5 in Appendix B.1 for D_{JSD} respectively.

On the 256×256 grid, it can be observed that replacing the λ-TD objective for learning V_ϕ by the proposed Sub-EB objective, the stability and convergence rate of the policy-based method are significantly improved. While the final performances of the two policy-based methods (RL and Sub-EB) are close, they both outperform Sub-TB and CV. These experimental results strongly support the effectiveness of the Sub-EB objective in enabling more reliable learning of the evaluation function V, leading to improved stability and faster convergence during policy-based GFlowNet training. The empirical gradient estimator constructed solely from the current training batch is not adequate for reliably guiding policy-based training. On the $128\times128\times128$ grid, the stability and convergence rate of Sub-EB are again much better than RL. While all three methods achieve similar final performance,

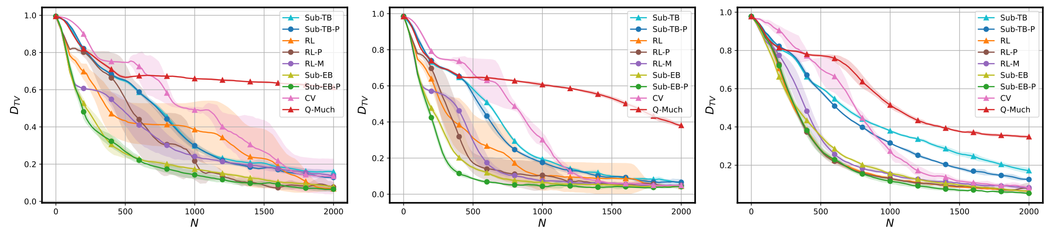

Figure 1: Plots of the means and standard deviations (represented by the shaded area) of D_{TV} for different training methods with parameterized π_B and uniform π_B on the 256×256 (left) and 128×128 (middle) and $64 \times 64 \times 64$ (right) grids, based on five randomly started runs for each method. By default, metric values are recorded every 20 iterations over $N = 2000$ training iterations and smoothed by a sliding window of length 5 for all plotted curves in this paper.

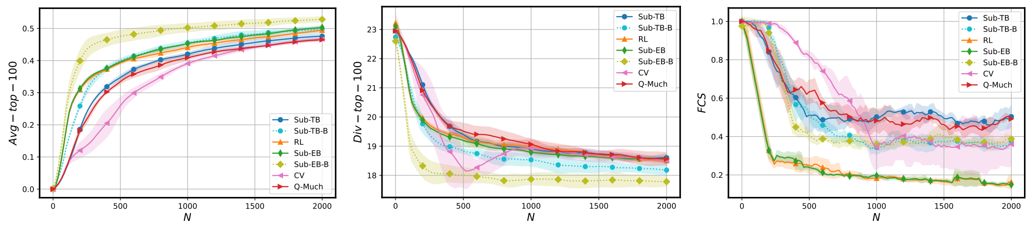

Figure 2: Plots of the mean and standard deviation values (represented by the shaded area) of average reward (left), diversity (right) and FCS (right) of the top 100 unique candidate graphs over 10 nodes, based on five randomly started runs for each method.

both RL and Sub-EB outperform Sub-TB and CV in terms of convergence rate. These findings further validate the effectiveness of our Sub-EB objective. Finally, on the $64 \times 64 \times 64$ grid, both RL and Sub-EB outperform Sub-TB and CV, but the behavior of RL and Sub-EB is very close. This can be ascribed to the fact that the stability of RL is good enough for this experiment, so the advantages brought by the Sub-TB objectives is not obvious. Besides, since hypergrids are homogeneous w.r.t. each dimension, and the minimum distance between modes only depends on N, the environment height N can have more effect on the modeling difficulty than the environment dimension D (Niu et al., 2024).

Ablation study on π_B To demonstrate that the Sub-EB objective naturally accommodates a parameterized backward policy. We compare the performance of the different methods with parameterized and uniform π_B on 256×256 and 128×128 and $64 \times 64 \times 64$ grids. We use '-P' to denote the method with parameterized π_B. RL-P uses the two-phase algorithm by Niu et al. (2024) and RL-M uses the approach by Gritsaev et al.. As shown in Fig. 1 and Fig. 5 in Appendix B.1, Sub-EB-P achieves the best performance and training stability among all the evaluated methods. This confirms that the Sub-EB objective well accommodates backward policies, which are parameterized and updated jointly with evaluation functions. We also conduct an ablation study λ, which is deferred to Appendix B.1.

Sequence design In this set of experiments, we use GFlowNets to generate biological and molecular sequences of length D, which are composed of M building blocks (Shen et al., 2023). We use nucleotide sequence datasets (SIX6 and PHO4), and molecular sequence datasets (QM9 and sEH) from Shen et al. (2023). The detailed description of the generative process and experimental results are provided in Appendix B.2.

BN structure learning In this experiment set, we focus on real-world studies of Bayesian Network (BN) structure learning (Malkin et al., 2022b; Niu et al., 2024). Here, the object space $\mathcal{X}$ corresponds to the space of BN structures. The detailed description of the generative process is provided in Appendix B.3. We consider three cases with 5, 10, and 15 nodes, where the sizes of $\mathcal{X}$ are approximately 2.95×10^4, 4.18×10^{18}, and 2.38×10^{35}, respectively. As in sequence design experiments, we also augment Sub-TB and the offline Sub-EB (Algorithm 2) with the local search technique Kim et al. (2023b) for designing $P_{\mathcal{D}}$, yielding the variants Sub-TB-B and Sub-EB-B. While such off-policy techniques that explicitly encourage the exploration of high rewards may not benefit overall distribution modeling, they can be valuable when the focus is on mode discovery during training.

We present the results and discussion for the two large-scale cases below and defer the small-scale case to Appendix B.3. For the large-scale cases, either $P_F(x)$ or $P^*(x)$ is computationally infeasible. Instead, we report the average reward of the top 100 unique graphs that are discovered during the training process. Since effective distribution modeling performance implies not only optimality but also diversity of generated candidates, we also compute the mean pairwise Hamming distance among these 100 graphs as a measure of diversity. It should be noted that neither excessively high nor excessively low diversity is desirable: the former corresponds to near-random generation, while the latter indicates that generation gets stuck in a limited set of structures. Thanks to the works by (Silva et al., 2025), we use Flow Consistency in Sub-graphs (FCS) as the unbiased estimation of D_{TV}. For FCS, we randomly sampled 32 batches of terminal states of size up to 128 for the Monte Carlo estimator. The experimental results are presented in Fig. 2 and Fig. 12 in the Appendix B.3. Among RL, Sub-EB, Sub-TB, Q-Much and CV, the results indicate that Sub-EB achieves the highest average reward, and both RL and Sub-EB converge faster than Sub-TB. All three methods attain similar diversity. More importantly, only RL and Sub-EB obtain strong distribution-modeling performance as measured by the FCS metric. Taking together all these findings, we can conclude that all methods achieve appropriate distribution modeling, and Sub-EB performs the best. This supports that Sub-EB not only enables reliable policy-based training but also scales effectively to large combinatorial spaces, providing both high-quality and diverse solutions. For the two variants, Sub-TB-B and Sub-EB-B, it can be observed that Sub-EB-B achieves the highest average reward among all five methods, accompanied by a more noticeable drop in diversity. Given that the local search component explicitly prioritizes high-reward regions of the solution space, such a trade-off—significant reward improvement at the expense of reduced diversity—is expected. In contrast, the Sub-TB-B does achieve a higher average reward compared to its non-augmented counterpart (Sub-TB) with a moderate decrease in diversity. However, the trade-off becomes much less pronounced. Without the local search technique, Sub-EB already achieves a comparable average reward and higher diversity than Sub-TB-B. Overall, Sub-EB-B proves to be more effective than Sub-TB-B, aligning well with our expectations of the optimality-diversity trade-off. This finding further supports the superiority of the policy-based methods, and validates that the Sub-EB objective enables the integration of offline techniques within policy-based frameworks.

Molecular graph design In this set of experiments, we consider the molecular graph design task (with $|\mathcal{X}| \approx 10^{16}$) described in Bengio et al. (2021). The sequence design task based on the sEH dataset (with $|\mathcal{X}| \approx 3.4 \times 10^7$) (Shen et al., 2023) is simplified from these tasks. A detailed description of the generative process and the corresponding experimental results for these graph design tasks are provided in the Appendix B.4. Sub-EB achieves the best overall performance on large-scale molecular graph design, providing higher average rewards, faster convergence, and competitive diversity compared to RL, and Sub-TB.

6 Discussion and Conclusion

In this work, we have established the connection between the state flow function $F(s)$ and the evaluation function $V(s)$. Built upon that, a new objective, called Sub-EB, is proposed for learning the evaluation function $V(s)$. Through three sets of experiments, we provide empirical evidence that the new Sub-EB objective enables more stable and flexible learning of V than the λ-TD objective in GFlowNet training, thereby improving the performance of the RL-like policy-based methods. In principle, the Sub-EB objective allows flexible choices of weight coefficients. Further investigation of designing optimal weight coefficients is left for future work.

7 Reproducibility Statement

Implementation details such as model configurations and hyperparameter choices are provided in Appendix B. Our implementation is based on the *torchgfn* package (Lahlou et al., 2023), and the code is also included in the supplementary materials.

References

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Lazar Atanackovic and Emmanuel Bengio. Investigating generalization behaviours of generative flow networks. In *ICML 2024 Workshop on Structured Probabilistic Inference* $\{\backslash\&\}$ *Generative Modeling*, 2024.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pp. 263–272. PMLR, 2017.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pp. 518–528. PMLR, 2022.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. In *Uncertainty in Artificial Intelligence*, pp. 997–1021. PMLR, 2024.

Timofei Gritsaev, Morozov Nikita, Samsonov Sergey, and Daniil Tiapkin. Optimizing backward policies in gflownets via trajectory likelihood maximization. In *The Thirteenth International Conference on Learning Representations*.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 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.

Hyosoon Jang, Minsu Kim, and Sungsoo Ahn. Learning energy decompositions for partial inference of gflownets. In *The Twelfth International Conference on Learning Representations*, 2023.

Hyosoon Jang, Yunhui Jang, Minsu Kim, Jinkyoo Park, and Sungsoo Ahn. Pessimistic backward policy for gflownets. *Advances in Neural Information Processing Systems*, 2024.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

Minsu Kim, Joohwan Ko, Dinghuai Zhang, Ling Pan, Taeyoung Yun, Woo Chang Kim, Jinkyoo Park, and Yoshua Bengio. Learning to scale logits for temperature-conditional gflownets. In *NeurIPS 2023 AI for Science Workshop*, 2023a.

Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search gflownets. In *The Twelfth International Conference on Learning Representations*, 2023b.

Jack Kuipers, Giusi Moffa, and David Heckerman. Addendum on the scoring of gaussian directed acyclic graphical models. 2014.

Salem Lahlou, Joseph D Viviano, and Victor Schmidt. torchgfn: A pytorch gflownet library. *arXiv preprint arXiv:2305.14594*, 2023.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022a.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. Gflownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2022b.

Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Emilie Kaufmann, Edouard Leurent, and Michal Valko. Fast active learning for pure exploration in reinforcement learning. In *International Conference on Machine Learning*, pp. 7599–7608. PMLR, 2021.

Puhua Niu, Shili Wu, Mingzhou Fan, and Xiaoning Qian. Gflownet training by policy gradients. In *Forty-first International Conference on Machine Learning*, 2024.

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, pp. 26878–26890. PMLR, 2023.

Jarrid Rector-Brooks, Kanika Madan, Moksh Jain, Maksym Korablyov, Cheng-Hao Liu, Sarath Chandar, Nikolay Malkin, and Yoshua Bengio. Thompson sampling for improved exploration in gflownets. In *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2023.

Robert W Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V: Proceedings of the Fifth Australian Conference, Held at the Royal Melbourne Institute of Technology, August 24–26, 1976*, pp. 28–43. Springer, 2006.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International Conference on Machine Learning*, pp. 30956–30975. PMLR, 2023.

Tiago Silva, de Souza da Silva Eliezer, and Mesquita Diego. On divergence measures for training gflownets. *Advances in Neural Information Processing Systems*, 2024.

Tiago Silva, Alves Rodrigo Barreto, da Silva Eliezer de Souza, Souza Amauri H, Garg Vikas, Kaski Samuel, and Mesquita Diego. When do gflownets learn the right distribution? In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Daniil Tiapkin, Morozov Nikita, Naumov Alexey, and Vetrov P Dmitry. Generative flow networks as entropy-regularized rl. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024.

Heiko Zimmermann, Hao Wu, Babak Esmaeili, and Jan-Willem van de Meent. Nested variational inference. *Advances in Neural Information Processing Systems*, 34:20423–20435, 2021.

Heiko Zimmermann, Fredrik Lindsten, Jan-Willem van de Meent, and Christian A Naesseth. A variational perspective on generative flow networks. *Transactions on Machine Learning Research*, 2022.

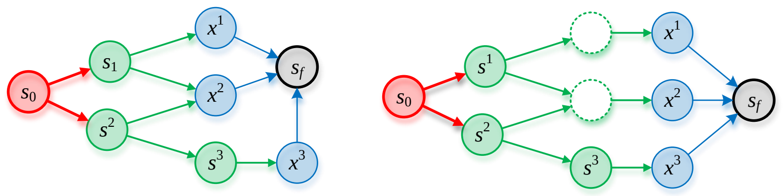

Figure 3: A graphical illustration of a DAG (left) and its graded version (right). Dotted circles represent dummy states, added during the conversion to a graded DAG.

LLM Usage Disclosure

LLMs were used only for text refinement (grammar and style). All scientific content was developed and verified by the authors.

A Theoretical analyses

A.1 Proof of Theorem 3.1

Proof. We first prove the **sufficiency** of the Sub-EB condition (5). Assume that V is an evaluation function over $\mathcal{S}$ that satisfies the Sub-EB condition for a given forward policy π_F. Then, for any $h \in [H]$:

$$\mathbb{E}_{P_F(s_h)\pi_F(s_{h+1}|s_h)}\left[\log \pi_B(s_h|s_{h+1}) + \dot{V}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{V}(s_h)\right] = 0 \tag{13}$$

$$\Downarrow$$

$$\mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \pi_B(s_h|s_{h+1}) + \dot{V}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{V}(s_h)\right] = 0 \tag{14}$$

$$\Downarrow$$

$$\dot{V}(s_h) = \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \dot{V}(s_{h+1})\right]. \tag{15}$$

Here, the second equation holds by our assumption that any valid π_F should introduce a flow F, which is a positive measure over trajectories over $\mathcal{G}$. Consequently, the corresponding state probability $P_F(s)$ is strictly positive for any $s \in \mathcal{S}$.

Based on (15), the previous definition $\log \pi_B(x|s_f) + \dot{V}(s_f) := R(x)$, we have:

$$\begin{aligned}
\dot{V}(x) &= \mathbb{E}_{\pi_F(s_f|x)}\left[\log \frac{R(x)}{\pi_F(s_f|x)}\right] \\
&= \mathbb{E}_{\pi_F(s_f|x)}\left[\log \frac{F^*(x \to s_f)}{\pi_F(s_f|x)}\right] \\
&= \mathbb{E}_{P_F(x \to s_f|x)}\left[\log \frac{P_B(x \to s_f)Z^*}{P_F(x \to s_f|x)}\right] \\
&= \mathbb{E}_{P_F(x \to s_f|x)}\left[\log \frac{P_B(x \to s_f|x)}{P_F(x \to s_f|x)}\right] + \log P_B(x)Z^* \\
&= \log F^*(x) - D_{\mathrm{KL}}(P_F(\tau_{H:}|s_H)\|P_B(\tau_{H:}|s_H)),
\end{aligned} \tag{16}$$

$$\begin{aligned}
\dot{V}(s_{H-1}) &= \mathbb{E}_{\pi_F(x|s_{H-1})}\left[\log\frac{\pi_B(s_{H-1}|x)}{\pi_F(x|s_{H-1})} + \mathbb{E}_{P_F(x\to s_f|x)}\left[\log\frac{P_B(x\to s_f)Z^*}{P_F(x\to s_f|x)}\right]\right] \\
&= \mathbb{E}_{P_F(\tau_{H-1:}|s_{H-1})}\left[\log\frac{P_B(\tau_{H-1:})Z^*}{P_F(\tau_{H-1:}|s_{H-1})}\right] \\
&= \mathbb{E}_{P_F(\tau_{H-1:}|s_{H-1})}\left[\log\frac{P_B(\tau_{H-1:}|s_{H-1})}{P_F(\tau_{H-1:}|s_{H-1})}\right] + \log P_B(s_{H-1})Z^* \\
&= \log F^*(s_{H-1}) - D_{\mathrm{KL}}(P_F(\tau_{H-1:}|s_{H-1})\|P_B(\tau_{H-1:}|s_{H-1})), \qquad (17) \\
&\vdots \\
\dot{V}(s_h) &= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log\frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \mathbb{E}_{P_F(\tau_{h+1:}|s_{h+1})}\left[\log\frac{P_B(\tau_{h+1:})Z^*}{P_F(\tau_{h+1:}|s_{h+1})}\right]\right] \\
&= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\log\frac{P_B(\tau_{h:})Z^*}{P_F(\tau_{h:}|s_h)}\right] \qquad (18) \\
&= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\log\frac{P_B(\tau_{h:}|s_h)}{P_F(\tau_{h:}|s_h)}\right] + \log P_B(s_h)Z^* \\
&= \log F^*(s_h) - D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h)\|P_B(\tau_{h:}|s_h)), \qquad (19) \\
&\vdots \\
\dot{V}(s_0) &= \mathbb{E}_{P_F(\tau|s_h)}\left[\log\frac{P_B(\tau)Z^*}{P_F(\tau)}\right] \\
&= \mathbb{E}_{P_F(\tau)}\left[\log\frac{P_B(\tau)}{P_F(\tau)}\right] + \log Z^* \\
&= \log F^*(s_0) - D_{\mathrm{KL}}(P_F(\tau)\|P_B(\tau)). \qquad (20)
\end{aligned}$$

It should be noted that the definition of evaluation function (3) coincides with the equation (18) in that

$$\begin{aligned}
\dot{V}(s_h) = \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\sum_{i=h}^{H}\log\frac{\widetilde{\pi}_B(s_i|s_{i+1})}{\pi_F(s_{i+1}|s_i)}\right] &= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\sum_{i=h}^{H}\log\frac{\pi_B(s_i|s_{i+1})}{\pi_F(s_{i+1}|s_i)}\right] + \log Z^* \\
= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\log\frac{P_B(\tau_{h:})Z^*}{P_F(\tau_{h:}|s_h)}\right]. \qquad (21)
\end{aligned}$$

Now, we prove the **necessity** of the Sub-EB condition (5). Assume that V is the evaluation function of a given forward policy π_F. Then, $\forall h \in [H]$:

$$\begin{aligned}
\dot{V}(s_h) &= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\sum_{i=h}^{H}\log\frac{\pi_B(s_i|s_{i+1})}{\pi_F(s_{i+1}|s_i)}\right] + \log Z^* \\
&= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log\frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \mathbb{E}_{P_F(\tau_{h+1:}|s_{h+1})}\left[\log\frac{P_B(\tau_{h+1:})Z^*}{P_F(\tau_{h+1:}|s_{h+1})}\right]\right] \\
&= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log\frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \dot{V}(s_{h+1})\right]. \qquad (22)
\end{aligned}$$

Therefore,

$$\mathbb{E}_{P_F(s_h)\pi_F(s_{h+1}|s_h)}\left[\log \pi_B(s_h|s_{h+1}) + \dot{V}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{V}(s_h)\right] = 0 \tag{23}$$

$$\Downarrow$$

$$\sum_{l=i}^{j-1} \mathbb{E}_{P_F(s_l \to s_{l+1})}\left[\log \pi_B(s_l|s_{l+1}) + \dot{V}(s_{l+1}) - \dot{V}(s_l) - \log \pi_F(s_{l+1}|s_l)\right] = 0 \tag{24}$$

$$\Downarrow$$

$$\mathbb{E}_{P_F(\tau_{i:j})}\left[\sum_{l=i}^{j-1} \log \pi_B(s_l|s_{l+1}) + \dot{V}(s_j) - \dot{V}(s_i) - \sum_{l=i}^{j-1} \log \pi_F(s_{l+1}|s_l)\right] = 0 \tag{25}$$

$$\Downarrow$$

$$\mathbb{E}_{P_F(\tau_{i:j})}\left[\log P_B(\tau_{i:j}|s_j) + \dot{V}(s_j) - \log P_F(\tau_{i:j}|s_i) - \dot{V}(s_i)\right] = 0 \tag{26}$$

for any $i < j \in [H+1]$. □

A.2 Proof of Theorem 3.2

Proof. We first prove the **sufficiency** of the Sub-TB condition (1). Assume that F is a state flow function over $\mathcal{S}$, and π_F is a forward policy, such that they satisfy the Sub-TB condition. Then, for any $h \in [H]$:

$$\mathbb{E}_{P_{\mathcal{D}}(s_h \to s_{h+1})}\left[\log \pi_B(s_h|s_{h+1})F(s_{h+1}) - \log \pi_F(s_{h+1}|s_h)F(s_h)\right] = 0 \tag{27}$$

$$\Downarrow$$

$$\log \pi_B(s_h|s_{h+1})F(s_{h+1}) - \log \pi_F(s_{h+1}|s_h)F(s_h) = 0 \tag{28}$$

$$\Downarrow$$

$$\log F(s_h) = \log \frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \log F(s_{h+1}). \tag{29}$$

Here, the second equation holds due to the following two reasons. First, by our assumption that the trajectory distribution $P_{\mathcal{D}}$, which is induced by $\pi_{\mathcal{D}}$, assigns non-zero probability to all trajectories in $\mathcal{T}$. Thus, the marginal probability $P_{\mathcal{D}}(s \to s')$ is strictly positive for any $(s \to s') \in \mathcal{E}$. Second, $\pi_{\mathcal{D}}$ is arbitrarily constructed and may differ from π_F. Let

$$\pi^{\dagger}(s'|s) := \frac{\pi_B(s|s')F(s')}{\sum_{s'} \pi_B(s|s')F(s')}. \tag{30}$$

Summing both sides of (28) over s_{h+1} yields $F(s_h) = \sum_{s_{h+1}} \log F(s_h)\pi_F(s_{h+1}|s_h) = \sum_{s_{h+1}} \log \pi_B(s_h|s_{h+1})F(s_{h+1})$. Inserting this into (28), we arrive at $\pi_F = \pi^{\dagger}$. Then, taking the expectation of (29) w.r.t. π_F, we have for any $h \in [H]$:

$$\begin{aligned}\log F(s_h) &= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \log F(s_{h+1})\right] &(31)\\ &= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \frac{\pi^{\dagger}(s_{h+1}|s_h)}{\pi_F(s_{h+1}|s_h)}\right] + \log \sum_{s_{h+1}} \pi_B(s_h|s_{h+1})F(s_{h+1}) \\ &= -D_{\mathrm{KL}}(\pi_F(\cdot|s_h)\|\pi^{\dagger}(\cdot|s_h)) + \log \sum_{s_{h+1}} \pi_B(s_h|s_{h+1})F(s_{h+1}). &(32)\end{aligned}$$

Note that the second term in the last equality is independent of $\pi_F(\cdot|s_h)$. Then, given $\pi_B(s_h|\cdot)$ and $F(s_{h+1})$, $F(s_h)$ is maximized at s_h when $\pi(\cdot|s_h) = \pi^{\dagger}(\cdot|s_h)$. Accordingly, given $\pi_B(s_{h+1}|\cdot)$ and $F(s_{h+2})$, $F(s_{h+1})$ is maximized at s_{h+1} when $\pi_F(\cdot|s_{h+1}) = \pi^{\dagger}(\cdot|s_{h+1})$. Keeping doing this recursion from $h = 0$ to $h = H$, we get the conclusion that F is maximized when $\pi_F = \pi^{\dagger}$ for any $(s \to s') \in \mathcal{E}$. When achieving the maximum, we have $\log F(x) = \log \sum_{s_f} \pi(x|s_f)F(s_f) := \log R(x) = \log F^*(x)$, $\log F(s_{H-1}) = \log \sum_x \pi_B(s_{H-1}|x)F^*(x) = \log F^*(s_{H-1}), \ldots, \log F(s_0) = \log \sum_{s_1} \pi_B(s_0|s_1)F^*(s_1) = F^*(s_0)$. Therefore, $F(s) =$

$F^*(s)$ for any $s \in \mathcal{S}$. Combining this with (28), we have $\pi_F = \pi_F^*$, where $\pi_F^*(s'|s) = \pi_B(s|s')F^*(s')/F^*(s)$ is the desired forward policy induced by π_B. Moreover, based on (31) and a derivation analogous to that of $\dot{V}$, we have, for Markovian P_B and any $h \in [H]$:

$$\log F(s_h) := \log F^*(s_h) - D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h) \| P_B(\tau_{h:}|s_h)). \tag{33}$$

Finally, we conclude:

$$\log F(s_h) = \log F^*(s_h) - \min D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h) \| P_B(\tau_{h:}|s_h)) = \log F^*(s_h), \tag{34}$$

Now we prove the **necessity** of the Sub-TB condition (1). Assume that F and π_F satisfy the equation (34) above. Then the equations (33) and (31) must also hold for F and π_F, and we have, for any $h \in [H]$:

$$\begin{aligned}
\log F(s_h) &= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\log \frac{P_B(\tau_{h:})Z^*}{P_F(\tau_{h:}|s_h)}\right] \\
&= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\sum_{i=h}^{H} \log \frac{\pi_B(s_i|s_{i+1})}{\pi_F(s_{i+1}|s_i)}\right] + \log Z^* \\
&= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \mathbb{E}_{P_F(\tau_{h+1:}|s_{h+1})}\left[\log \frac{P_B(\tau_{h+1:})Z^*}{P_F(\tau_{h+1:}|s_{h+1})}\right]\right] \\
&= \mathbb{E}_{\pi_F(s_{h+1}|s_h)}\left[\log \frac{\pi_B(s_h|s_{h+1})}{\pi_F(s_{h+1}|s_h)} + \log F(s_{h+1})\right].
\end{aligned} \tag{35}$$

We can rewrite the above equation as follows:

$$\log F(s_h) := -D_{\mathrm{KL}}(\pi_F(\cdot|s_h) \| \pi^\dagger(\cdot|s_h)) + \log \sum_{s_{h+1}} \pi_B(s_h|s_{h+1})F(s_{h+1}), \quad \forall h \in [H]. \tag{36}$$

Since $\pi_F(\cdot|s_h)$ is independent of $\pi_B(s_h|\cdot)$ and $F(s_{h+1})$, the assumption that $\log F(s_h)$ is maximized at s_h indicates $\pi_F(\cdot|s_h) = \pi^\dagger(\cdot|s_h)$. Therefore, we recover $\forall h \in [H]$:

$$\log \pi_F(s_{h+1}|s_h)F(s_h) = \log \pi_B(s_h|s_{h+1})F(s_{h+1}), \tag{37}$$

and the Sub-TB condition can be easily derived from it. □

A.3 Proof of Theorem 3.3

Proof. We first prove the **sufficiency** of the backward Sub-EB condition (10). Assume that W is an evaluation function over $\mathcal{S}\backslash\{s_f\}$ that satisfies the backward Sub-EB condition for a given backward policy π_B. Then, for any $h \in [H-1]$:

$$\mathbb{E}_{\pi_B(s_h|s_{h+1})P_B(s_{h+1})}\left[\log \pi_B(s_h|s_{h+1}) + \dot{W}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{W}(s_h)\right] = 0 \tag{38}$$

$$\Downarrow$$

$$\mathbb{E}_{\pi_B(s_h|s_{h+1})}\left[\log \pi_B(s_h|s_{h+1}) + \dot{W}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{W}(s_h)\right] = 0 \tag{39}$$

$$\Downarrow$$

$$\dot{W}(s_{h+1}) = \mathbb{E}_{\pi_B(s_h|s_{h+1})}\left[\log \frac{\pi_F(s_{h+1}|s_h)}{\pi_B(s_h|s_{h+1})} + \dot{W}(s_h)\right]. \tag{40}$$

Here, the second equation holds by our assumption that any valid π_B and R should introduce a flow F^*, which is a positive measure over trajectories over $\mathcal{G}$. Consequently, the corresponding state probability $P_B(s)$ is strictly positive for any $s \in \mathcal{S}$.

Based on (40), we have:

$$\begin{aligned}
\dot{W}(s_1) &= \mathbb{E}_{\pi_B(s_0|s_1)}\left[\log \frac{\pi_F(s_1|s_0)}{\pi_B(s_0|s_1)}\right] + \dot{W}(s_0) \\
&= \mathbb{E}_{P_B(s_0 \to s_1|s_1)}\left[\log \frac{P_F(s_0 \to s_1)}{P_B(s_0 \to s_1|s_1)}\right] + \log F(s_0) \\
&= \mathbb{E}_{P_B(s_0 \to s_1|s_1)}\left[\log \frac{P_F(s_0 \to s_1|s_1)}{P_B(s_0 \to s_1|s_1)}\right] + \log F(s_0) P_F(s_1) \\
&= \log F(s_1) - D_{\mathrm{KL}}(P_B(\tau_{:1}|s_1) \| P_F(\tau_{:1}|s_1)), \qquad (41) \\
&\vdots \\
\dot{W}(s_{h+1}) &= \mathbb{E}_{\pi_B(s_h|s_{h+1})}\left[\log \frac{\pi_F(s_{h+1}|s_h)}{\pi_B(s_h|s_{h+1})} + \mathbb{E}_{P_B(\tau_{:h}|s_h)}\left[\log \frac{P_F(\tau_{:h})}{P_B(\tau_{:h}|s_h)}\right]\right] + \dot{W}(s_0) \\
&= \mathbb{E}_{P_B(\tau_{:h+1}|s_{h+1})}\left[\log \frac{P_F(\tau_{:h+1})}{P_B(\tau_{:h+1}|s_{h+1})}\right] + \log F(s_0) \qquad (42) \\
&= \mathbb{E}_{P_B(\tau_{:h+1}|s_{h+1})}\left[\log \frac{P_F(\tau_{:h+1}|s_{h+1})}{P_B(\tau_{:h+1}|s_{h+1})}\right] + \log F(s_0) P_F(s_{h+1}) \\
&= \log F(s_{h+1}) - D_{\mathrm{KL}}(P_F(\tau_{:h+1}|s_{h+1}) \| P_B(\tau_{:h}|s_{h+1})), \qquad (43) \\
&\vdots \\
\dot{W}(x) &= \mathbb{E}_{\pi_B(s_{H-1}|x)}\left[\log \frac{\pi_F(x|s_{H-1})}{\pi_B(s_{H-1}|x)} + \mathbb{E}_{P_B(\tau_{:H-1}|s_{H-1})}\left[\log \frac{P_F(\tau_{:H-1})}{P_B(\tau_{:H-1}|s_{H-1})}\right]\right] + \dot{W}(s_0) \\
&= \mathbb{E}_{P_B(\tau|x)}\left[\log \frac{P_F(\tau)}{P_B(\tau|x)}\right] + \log F(s_0) \\
&= \mathbb{E}_{P_B(\tau|x)}\left[\log \frac{P_F(\tau|x)}{P_B(\tau|x)}\right] + \log F(s_0) P_F(x) \\
&= \log F(x) - D_{\mathrm{KL}}(P_B(\tau|x) \| P_F(\tau|x)). \qquad (44)
\end{aligned}$$

It should be noted that the definition of evaluation function (9) coincides with the equation (42) in that

$$\begin{aligned}
\dot{W}(s_{h+1}) = \mathbb{E}_{P_B(\tau|x)}\left[\sum_{i=0}^{h} \log \frac{\widetilde{\pi}_F(s_{i+1}|s_i)}{\pi_B(s_i|s_{i+1})}\right] &= \mathbb{E}_{P_B(\tau|x)}\left[\sum_{i=0}^{h} \log \frac{\pi_F(s_{i+1}|s_i)}{\pi_B(s_i|s_{i+1})}\right] + \log F(s_0) \\
&= \mathbb{E}_{P_F(\tau_{h:}|s_h)}\left[\log \frac{P_F(\tau_{:h+1})}{P_B(\tau_{:h+1}|s_{h+1})}\right] + \log F(s_0). \qquad (45)
\end{aligned}$$

In particular, when the backward Sub-EB condition is satisfied for $h = H$, we have

$$\mathbb{E}_{P_{\mathcal{D}}(x)}[\log \pi_F(s_f|x) + \dot{W}(x) - \log \pi_B(x|s_f) - \dot{W}(s_f)] = 0 \qquad (46)$$

$$\Downarrow$$

$$\mathbb{E}_{P_{\mathcal{D}}(x)}[\dot{W}(x) - \log R(x)] = 0. \qquad (47)$$

As $P_{\mathcal{D}}$ can be chosen arbitrarily, $\dot{W}(x) = R(x)$ is further implied.

Now, we prove the **necessity** of backward the Sub-EB condition (10). Assume that W is the evaluation function of a given forward policy π_B. Then, $\forall h \in [H-1]$:

$$\begin{aligned}
\dot{W}(s_{h+1}) &= \mathbb{E}_{P_B(\tau_{:h+1}|s_{h+1})}\left[\sum_{i=0}^{h} \log \frac{\pi_F(s_{i+1}|s_i)}{\pi_B(s_i|s_{i+1})}\right] + \dot{W}(s_0) \\
&= \mathbb{E}_{\pi_B(s_h|s_{h+1})}\left[\log \frac{\pi_F(s_{h+1}|s_h)}{\pi_B(s_h|s_{h+1})} + \mathbb{E}_{P_B(\tau_{:h}|s_h)}\left[\log \frac{P_F(\tau_{:h})F(s_0)}{P_B(\tau_{:h}|s_h)}\right]\right] \\
&= \mathbb{E}_{\pi_B(s_h|s_{h+1})}\left[\log \frac{\pi_F(s_{h+1}|s_h)}{\pi_B(s_h|s_{h+1})} + \dot{W}(s_h)\right]. \qquad (48)
\end{aligned}$$

and $\dot{W}(x) = \log R(x)$. Therefore,

$$\mathbb{E}_{P_B^{\mathcal{D}}(s_{h+1})\pi_B(s_h|s_{h+1})}\left[\log \pi_B(s_h|s_{h+1}) + \dot{W}(s_{h+1}) - \log \pi_F(s_{h+1}|s_h) - \dot{W}(s_h)\right] = 0 \tag{49}$$

$$\Downarrow$$

$$\sum_{l=i}^{j-1} \mathbb{E}_{P_B^{\mathcal{D}}(s_l \to s_{l+1})}\left[\log \pi_B(s_l|s_{l+1}) + \dot{W}(s_{l+1}) - \dot{W}(s_l) - \log \pi_F(s_{l+1}|s_l)\right] = 0 \tag{50}$$

$$\Downarrow$$

$$\mathbb{E}_{P_B^{\mathcal{D}}(\tau_{i:j})}\left[\sum_{l=i}^{j-1} \log \pi_B(s_l|s_{l+1}) + \dot{W}(s_j) - \dot{W}(s_i) - \sum_{l=i}^{j-1} \log \pi_F(s_{l+1}|s_l)\right] = 0 \tag{51}$$

$$\Downarrow$$

$$\mathbb{E}_{P_B^{\mathcal{D}}(\tau_{i:j})}\left[\log P_B(\tau_{i:j}|s_j) + \dot{W}(s_j) - \log P_F(\tau_{i:j}|s_i) - \dot{W}(s_i)\right] = 0 \tag{52}$$

for any $i < j \in [H+1]$ □

A.4 The complete version of the evaluation function

Taking into consideration the total flow estimator Z, $V^\dagger$ still takes the form of (3), only differing at $\widetilde{\pi}_B(x|s_f)$, which is redefined as $R(x)/Z$. In this case, it can be verified that $V^\dagger = D_{\mathrm{KL}}(P_F(\tau)\|P_B(\tau)) + \log \frac{Z}{Z^*}$. Then, the gradient of $V^\dagger(s_0;\theta)$ is equal to that of $D_{\mathrm{KL}}(P_F(\tau;\theta)\|P_B(\tau)) + \frac{1}{2}(\log Z - \log Z^*)^2$. The forms of the Sub-EB condition and objective remain unchanged except that $P_B(x|s_f)\exp \dot{V}(s_f)$ is refined as $R(x)/Z$.

Corollary A.1 (Corollary to Theorem 3.1)**.** *Suppose V is an evaluation function over $\mathcal{S}$ and F^* is the desired flow. Given a forward policy π_F,*

$$\forall h \in [H] : -V(s_h) = \log \frac{F^*(s_h)}{Z} - D_{\mathrm{KL}}(P_F(\tau_{h:}|s_h)\|P_B(\tau_{h:}|s_h)), \tag{53}$$

if and only if V satisfies the Sub-EB condition (5).

Proof. The proof can be done by replacing $R(x)$ in the proof of Theorem 3.1 by $R(x)/Z$ □

Finally, in Algorithm 1, the approximated $\nabla_\theta \mathbb{E}_{\mu(s_0;\theta)}\mu[V^\dagger(s_0;\theta)]$ (Niu et al., 2024) are computed to update both $\pi_F(\cdot|\cdot;\theta)$ and $Z(\theta)$, where $\mu(s_0;\theta) := Z(\theta)/Z$.

A.5 Comparison between λ-TD and Sub-EB objectives

The traditional λ-TD objective for $V(\cdot\,;\phi)$ can be expressed as follows:

$$\mathbb{E}_{P_F(\tau)}\left[\sum_{h=0}^{H}\left(V^\lambda(s_h) - V(s_h;\phi)\right)^2\right], \quad V^\lambda(s_h) := V(s_h) + \sum_{i=h}^{H} \lambda^{i-h}\delta_V(s_i \to s_{i+1}), \tag{54}$$

where V^λ is considered as constant when computing gradients w.r.t. ϕ, and $\delta_V(s_i \to s_{i+1})$ is equal to $\delta_V(\tau_{i,i+1})$ as defined in (8). Without consideration of gradient computation, the expression of the objective value can be simplified as:

$$\mathbb{E}_{P_F(\tau)}\left[\sum_{h=0}^{H}\sum_{i=h}^{H} \lambda^{i-h}(\delta_V(s_i \to s_{i+1}))^2\right]. \tag{55}$$

In comparison, the expression of the Sub-EB objective value is:

$$\mathbb{E}_{P_F(\tau)}\left[\sum_{\tau_{i,j}\,:\, i<j\in[H+1]} w_{j-i}(\delta_V(\tau_{i:j}))^2\right]. \tag{56}$$

It can be observed that the λ-TD objective only considers the events that start at step h for learning $V(s_h;\phi)$, and edges-wise mismatch $\delta_V(s_i \to s_{i+1})$. In contrast, the Sub-EB objective incorporates information from both events that start at h and those that end at h by considering subtrajectory-wise mismatches $\delta_V(\tau_{i:(\cdot)})$ that start at $i(\geq h)$ and $\delta_V(\tau_{(\cdot):j})$ that end at $j(\leq h)$. This results in a more balanced and reliable learning of $V(s_h;\phi)$. Besides, the form of w can be freely chosen, while it must be λ^{i-h} in the λ-TD objective (Schulman et al., 2016).

A.6 Additional related works

RL methods can be roughly categorized into two main framework (Sutton & Barto, 2018): the first is (soft) Q-learning, and the second is the actor–critic framework.

In the first framework, the core idea of Soft Q-learning (Haarnoja et al., 2017) is to learn a function Q that minimizes the mismatch of the *offline* Bellman equation for the transition environment $\mathcal{G}$ with edge reward $\log \pi_B(s|s')$ and $\log \pi_B(x|s_f) + V(s_f) := \log R(x)$. This objective of Q can be written as

$$\mathbb{E}_{P_{\mathcal{D}}(s\to s')}[\delta_Q(s\to s')^2], \quad \delta_Q(s \to s') := \log \pi_B(s|s') + V(s') - Q(s,s'), \tag{57}$$

with $\pi_F(s'|s) := \frac{\exp Q(s,s')}{\exp V(s)}$ and $V(s) := \log \sum_{s'} \exp Q(s,s')$. Any function Q that achieves zero mismatch is guaranteed to equal the optimal soft Q-function Q^* with the corresponding $V = V^*$ and $\pi_F = \pi_F^*$. Tiapkin et al. (2024) showed that if we treat $Q(s,s')$ as $\log F(s\to s')$ so that $V(s) = \log \sum_{s'} F(s \to s') = \log F(s)$, then the Bellman objective transforms into the DB objective. The distinction lies only in parameterization: one may parameterize (π_F, V) directly and represent $Q(s,s')$ as $V(s)\pi_F(s'|s)$, instead of parameterizing $Q(s,s')$ and deriving V and π_F from it. They further proved that the optimal solutions of the Bellman objective coincide with those of the DB objective from the perspective of Soft Q-learning. As acknowledged by the authors, their proof only applies to the DB objective with fixed π_B. To address this limitation, Deleu et al. (2024) established an equivalence between path-consistency learning (a generalized form of soft Q learning) and the Sub-TB objective from a gradient-based perspective. Compared to Deleu et al. (2024), our Theorem 3.2 offers a more direct and explicit connection along this RL direction. The major challenge in RL is balancing the exploration-exploitation trade-off (Sutton & Barto, 2018). Returning to GFlowNet, the DB objective tends to favor exploitation as the target flow logarithm is $\log F(s') + \log P_B(s|s')$, where $\log F$ encodes the learned partial knowledge about the task, resulting in **biased** but **low-variance** task feedback. In contrast, the TB objective encourages exploration as the target flow logarithm $\log(P_B(\tau)R(x))$ is independent of $\log F$, and serves as **unbiased** but **high-variance** feedback. The main advantage of the Sub-TB objective is that it enables a tunable **trade-off** between exploration and exploitation by adjusting the weights assigned to subtrajectories of different lengths, thereby having better performances This behavior is empirically demonstrated on hypergrid tasks, as shown in Fig. 4

Our work and Niu et al. (2024) is based on theory of policy-gradient Agarwal et al. (2021), which operates under the actor–critic framework Haarnoja et al. (2018). In the framework, we learn a function V that minimizes (but not necessarily) the *online* Bellman objective of V[3]:

$$\mathbb{E}_{P_F(s\to s')}[\delta_S(s\to s')^2], \quad \delta_S(s \to s') := \log \pi_F(s'|s) + V(s) - Q(s,s'), \tag{58}$$

with $Q(s,s') := \log \pi_B(s|s') + V(s')$ and $Q(x, s_f) := \log R(x)$, where $\mathbb{E}_{P_F(s\to s')}[\ldots]$ above can be generalized into $\mathbb{E}_{P_{\mathcal{D}}(s)}[\mathbb{E}_{\pi_F(s'|s)}[\ldots]]$, but the inner online expectation still must be maintained. **At this point, the two RL directions begin to diverge** (Schulman et al., 2017). When the V

[3]As the offline Bellman objective under the first framework can be written as the DB objective, a special case of the Sub-TB objective, the online Bellman objective for the second framework can be expressed as a special case of the Sub-EB objective.

achieves the optimal solution of the Bellman objective (denoted as $V^\dagger$), we have

$$\begin{aligned} V(s) = V^\dagger(s) &= \mathbb{E}_{\pi_F(s'|s)}[Q(s,s') - \log \pi_F(s'|s)] \\ &= \mathbb{E}_{\pi_F(s'|s)}\left[\frac{\exp Q(s,s')}{\log \sum_{s'} \exp Q(s,s')} - \log \pi_F(s'|s) + \log \sum_{s'} \exp Q(s,s')\right] \\ &= -D_{KL}\left(\pi_F(\cdot|s) \| \pi_Q(\cdot|s)\right) + \log \sum_{s'} \exp Q(s,s'), \quad \pi_Q(s'|s) := \frac{\exp Q(s,s')}{\sum_{s'} \exp Q(s,s')}. \end{aligned}$$

Defining $F(s \to s') := \exp Q(s,s')$ yields expressions that coincide with those used in the main text. This clarifies why $V^\dagger$ (and its learned approximate V) serves as a critic: it evaluates how far the π_F is from the local optimal policy as $Q(s,s')$ may still deviate from the global optimal one $Q^*(s,s')$. The divergence is then minimized w.r.t. actor π_F using critic V, so that $V^\dagger$ of π_F moves closer to the optimal one V^*. This can be achieved by simply setting $\pi_F(\cdot|s)$ to $\pi_Q(\cdot|s)$ for all sampled states or applying policy gradients for sampled trajectories. In Appendix A.7, we further derive the soft actor–critic algorithms based on basic minimization operations to better illustrate how the Sub-EB objective and policy gradients operate within the actor–critic framework.

A.7 Soft actor-critic for GFlowNet

The soft actor-critic algorithm tailored to GFlowNet training is presented in Alg. 3 and 4. When all states are visited through sampling in Alg. 3 and the online Bellman objective of V reaches zero, meaning $\forall s \in \mathcal{S} : D_{KL}(\pi_F(\cdot|s), \pi_Q(\cdot|s)) = 0$, and $V = V^\dagger$, we below show that policy π_F and V will converge to optimal quantities π_F^* and V^*. Starting at terminating states, we have:

$$\begin{aligned} V(x) &= -D_{KL}(\pi_F(\cdot|x) \| \pi_Q(\cdot|x)) + Q(x, s_f) \\ &= Q(x, s_f) := \log R(x) \qquad (59) \\ Q(s_{H-1}, x) &:= \log \pi_B(s_{H-1}|x) + V(x) \\ &= \log \frac{F^*(s_{H-1} \to x)}{F^*(x)} + \log R(x) = \log F^*(s_{H-1} \to x), \qquad (60) \end{aligned}$$

where we use the definition of π_B. Next, we have:

$$\begin{aligned} V(s_{H-1}) &:= -D_{KL}\left(\pi_F(\cdot|s_{H-1}) \| \pi_Q(\cdot|s_{H-1})\right) + \log \sum_x \exp Q(s_{H-1}, x) \\ &= \log \sum_x F^*(s_{H-1}, x) = \log F^*(s_{H-1}), \qquad (61) \\ \pi_F(x|s_{H-1}) = \pi_Q(x|s_{H-1}) &:= \frac{\exp Q(s_{H-1}, x)}{\sum_x \exp Q(s_{H-1}, x)} \\ &= \frac{F^*(s_{H-1} \to x)}{F^*(s_{H-1})} := \pi_F^*(x|s_{H-1}). \qquad (62) \end{aligned}$$

Continuing this recursion, we will arrive at $V^\dagger(s) = \log F^*(s)$, $Q^\dagger(s \to s') = \log F^*(s \to s')$, and $\pi_F(s'|s) = \pi_F^*(s'|s)$ for all s and $(s \to s')$.

The online expectation over π_F make algorithm 3 computational expensive, and also eliminating the possible for ehanced the edge-wise formulation of δ_S. To address this, one may modify it into Algorithm 4, where the key difference is online trajectories sampled from $P_F(\tau)$. While Niu et al. (2024) improve the basic policy-optimization operator using policy-gradient theory for practical implementation, our Sub-EB objective further enables subtrajectory-level formulations of δ_S and parameterized $\pi_B(\cdot \mid \cdot\ \phi)$. Since traditional policy-gradient methods operate strictly in an on-policy manner, our backward Sub-EB objective derived from Theorem 3.2 is introduced to enable the use of an offline sampler $P_{\mathcal{D}}$.

B Experimental settings and results

Hyperparameters For both the original policy-based method and the proposed one with the Sub-TB objective (RL and Sub-EB), we set the hyperparameter γ to 0.99 based on the ablation study

Algorithm 3 Soft Actor-Critic Workflow

```
Require: $\pi_F(\cdot\,|\cdot\,;\theta)$, $\pi_B(\cdot\,|\cdot)$, $V(\cdot\,;\phi)$, batch size $K$, number of total iterations $N$
  for $n = 1, \ldots, N$ do
    $\mathcal{D} \leftarrow \{\tau^k | \tau^k \sim P_{\mathcal{D}}(\tau)\}_{k=1}^K$
    Based on $\mathcal{D}$, update $\phi$ by its gradients w.r.t. $\frac{1}{K}\sum_{s\in\tau^k}\mathbb{E}_{\pi_F(s'|s)}[\delta_S(s\rightarrow s';\phi)^2]$.
    Based on $\mathcal{D}$ and $V$, setting $\pi_F(\cdot\,|s)$ to $\pi_Q(\cdot\,|s)$ for any $s(\neq s_f)\in\mathcal{D}$.
  end for
  return $\pi_F(\cdot\,|\cdot\,;\theta)$, $V(\cdot\,;\phi)$
```

Algorithm 4 Modified Actor-Critic Workflow

```
Require: $\pi_F(\cdot\,|\cdot\,;\theta)$, $\pi_B(\cdot\,|\cdot)$, $V(\cdot\,;\phi)$, batch size $K$, number of total iterations $N$
  for $n = 1, \ldots, N$ do
    $\mathcal{D} \leftarrow \{\tau^k | \tau^k \sim P_F(\tau)\}_{k=1}^K$
    Based on $\mathcal{D}$, update $\phi$ by its gradients w.r.t. $\frac{1}{K}\sum_{(s\rightarrow s')\in\tau^k}\delta_S(s\rightarrow s';\phi)^2$.
    Based on $\mathcal{D}$ and $V$, setting $\pi_F(\cdot\,|s)$ to $\pi_Q(\cdot\,|s)$ for any $s(\neq s_f)\in\mathcal{D}$.
  end for
  return $\pi_F(\cdot\,|\cdot\,;\theta)$, $V(\cdot\,;\phi)$
```

results reported in Niu et al. (2024). For the data collection policy $\pi_{\mathcal{D}}$ of Sub-TB, the hyperparameter α starts at 1.0 and decays exponentially at a rate of 0.99, where the decay rate is also determined based on the results of the ablation study in Niu et al. (2024). In the Sub-TB objective, the hyperparameter λ is set to 0.9 following the ablation study by Madan et al. (2023). For the Sub-EB objective, λ is set to be 0.9 selected from $\{0.1, 0.2, \ldots, 0.9, 0.99\}$ based on the ablation study results shown in Fig. 6.

Optimization The Adam optimizer is used for optimization. The sample batch size is set to 128 for each optimization iteration following Niu et al. (2024). The learning rates of $\pi_F(\cdot\,|\cdot\,;\theta)$ and $F^{\log}(\cdot\,;\theta)$ are equal to 1×10^{-3}, which is selected from $\{5\times10^{-3}, 1\times10^{-3}, 5\times10^{-4}, 1\times10^{-4}\}$ based on the performance of Sub-TB on the 256×256 grid. The learning rate of $V_\phi(\cdot)$ is set to 5×10^{-3}, which is selected from $\{10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 10^{-4}\}$ based on the performance of RL on the 256×256 grid. In all experiments, each training method is run five times, initialized from five different random seeds.

Model architecture The forward policy $\pi_F(\cdot\,|\,\cdot;\theta)$ and evaluation function $V_\phi(\cdot)$ are both parameterized by a neural network with four hidden layers, each with a hidden dimension of 256. The backward policy $\pi_B(\cdot\,|\,\cdot)$ is a uniform distribution over valid transitions (edges). In hypergrid and sequence design experiments, coordinate tuples and integer sequences are transformed using K-hot encoding before entering the neural networks. In BN structure learning, adjacency matrices are used directly as input into neural networks without encoding.

Peformance metrics The first one is the total variation D_{TV} between $P_F(x)$ and $P^*(x)$, which is defined as: $D_{\text{TV}}(P_F(x), P^*(x)) = \frac{1}{2}\sum_{x\in\mathcal{X}}|P_F(x) - P^*(x)|$. An alternative performance metric adopted in literature is the average l_1-distance, which is defined as $|\mathcal{X}|^{-1}\sum_x|P^*(x) - P_F(x)|$. The reason that we use D_{TV} instead is as follows. The design space $|\mathcal{X}|$ is usually large ($> 10^4$) and $\sum_x|P^*(x) - P_F(x)| \leq 2$, resulting in the average l_1-distance being heavily scaled down by $|\mathcal{X}|$. We also evaluate different methods using the Jensen–Shannon divergence D_{JSD} as the second metric, which can be written as: $D_{\text{JSD}}(P_F(x), P^*(x)) = \frac{1}{2}D_{\text{KL}}(P_F(x), P_M(x)) + \frac{1}{2}D_{\text{KL}}(P^*(x), P_M(x))$, where $P_M := \frac{1}{2}(P_F + P^*)$.

B.1 Hypergrids

The generative process of hypergrid experiments is defined as follows. For a grid with height H and width D, the state space excluding the final state, $\mathcal{S}\backslash\{s_f\}$, consists of all D-dimensional coordinate vectors of the form $\{s = ([s]_1, \ldots, [s]_d, \ldots, [s]_D) \mid [s]_d \in \{0, \ldots, H-1\}\}$. The generating process begins at the initial state $s_0 = (0, \ldots, 0)$ and ends in the final state s_f, which we denote as $(-1, \ldots, -1)$. From any state $s \in \mathcal{S}\backslash\{s_f\}$, there are $D+1$ valid transitions (edges): (1) for each $d \in \{0, \ldots, D\}$, the d-th coordinate can be incremented by one, leading to a new state $s' = ([s]_1, \ldots, [s]_d + 1, \ldots, [s]_D)$; (2) if $s \in \mathcal{X}$, the process can be stopped by taking transition $(s\rightarrow s_f)$, returning s as the terminating coordinate tuple x. By definitions above, the DAGs of Hypergrid experiments are not *graded*, meaning every state (excluding s_f) can be returned as a

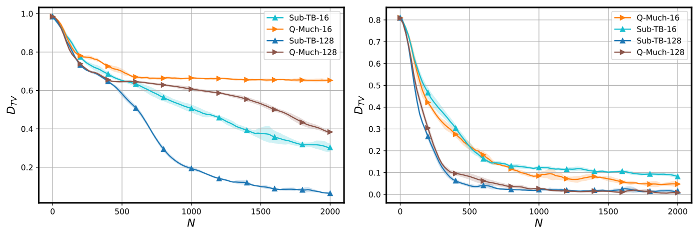

Figure 4: Plots of the means and standard deviations (represented by the shaded area) of D_{TV} (right) for different training methods on the 128×128 (left) 20×20 (right) grids, based on five randomly started runs for Sub-TB-16 Sub-TB-128, Q-much-16 and Q-much 128. Here, “16” and “128” denote the training batch sizes.

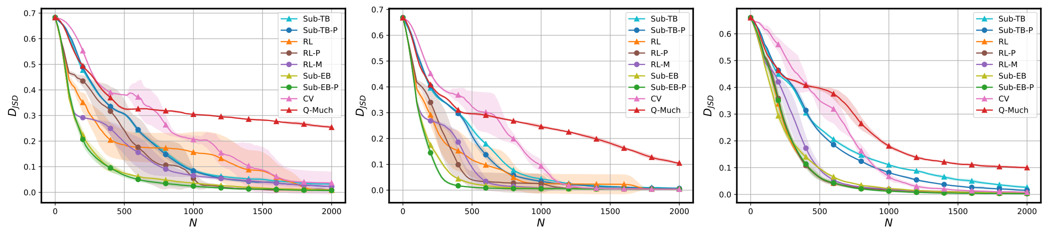

Figure 5: Plots of the means and standard deviations (represented by the shaded area) of D_{JSD} for different training methods on the 256×256 (left) 128×128 (middle) and $64 \times 64 \times 64$ (right) grids, based on five randomly started runs for each method.

terminating state. The reward function associated with terminating states is defined as:

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

where $R_0 = 10^{-2}$, $R_1 = 0.5$ and $R_2 = 2$ in our experiments.

Ablation study on λ For the 128×128 grid, we conduct an ablation study on the hyperparameter λ of the Sub-EB weights $w_{j-i} (= \lambda^{j-i} / \sum_{i<j \in [H+1]} \lambda^{j-i})$ to investigate its effect on policy-based GFlowNet training. We run Sub-EB methods with λ equal to $0.1, 0.2, \ldots, 0.90$ and 0.99. The experimental results are depicted in Fig. 6. It can be observed that the Sub-EB method with $\lambda = 0.9$ achieves the best performance. Although convergence rates and final performances differ, Sub-EB methods under all configurations exhibit good stability, demonstrating that the proposed Sub-EB objective enables reliable learning of the evaluation function V.

B.2 SEQUENCE DESIGN

The generative process of this set of experiments is defined as follows. The state space excluding the final state $\mathcal{S} \backslash \{s_0, s_f\}$ is equal to $\bigcup_{t=1}^{H} \{0, \ldots, N-1\}^t$ where each state is a sequence composed of integers ranging from 0 to $N-1$. The set $\{0, \ldots, N-1\}$ corresponds to the N types of building blocks. The process begins at the initial state $s_0 = (-1, \ldots, -1)$, which represents an empty sequence, and ends at the final state $s_f = (N, \ldots, N)$. For any intermediate state $s_h \in \mathcal{S}_h$, it contains h elements drawn from $\{0, \ldots, N-1\}$. There are $N \times 2$ possible transitions for s_t with $t < D$, corresponding to either appending or prepending one element from $\{0, \ldots, N-1\}$ to the current sequence. For implementation easiness in practice, each state s_t is equivalently represented as a sequence of fixed length D, where the first t elements are integers from $\{0, \ldots, N-1\}$ and the others are equal to -1. This generative process continues until sequences reach length D. By definition, $\mathcal{G}$ is a graded DAG and $\mathcal{S}_D = \mathcal{X} = \{0, \ldots, N-1\}^D$. We use nucleotide sequence datasets (SIX6 and PHO4), and molecular sequence datasets (QM9 and sEH) from Shen et al. (2023). Rewards are defined as the exponents of raw scores from the datasets, $R(x) = \mathrm{score}^{\beta}(x)$.

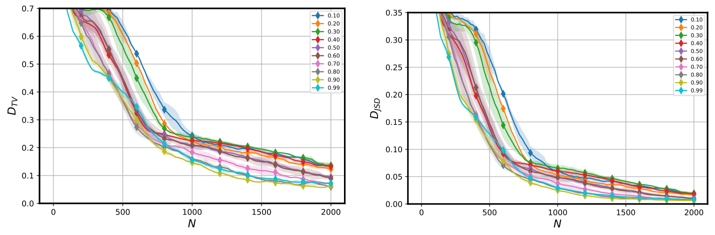
Figure 6: Plots of the means and standard deviations (represented by the shaded area) of D_{TV} (left) and D_{JSD} (right) for Sub-EB runs on the 128×128 grid. The hyperparameter λ for Sub-EB runs ranges from 0.1 to 0.99. The results are based on five Sub-EB runs for each setup.

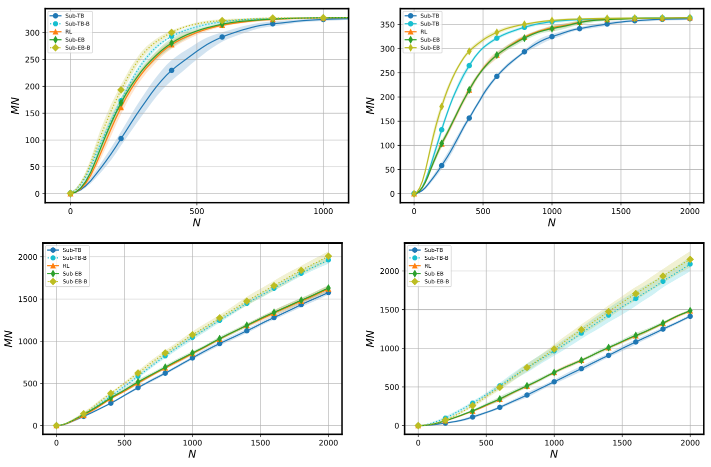
Figure 7: Plots of the mean and standard deviation values (represented by the shaded area) of MN for the SIX6 (top left), QM9 (top right), PHO4 (bottom left), and sEH (bottom right) dataset, based on five randomly started runs for each method.

The hyperparameter β is set to 3, 5, 3, 6, and rewards are normalized to $[10^{-3}, 10]$, $[10^{-3}, 10]$, $[0, 10]$ and $[10^{-3}, 10]$ for SIX6, QM9, PHO4 and sEH, respectively. In this experimental setting, we consider an additional metric introduced by Shen et al. (2023). It is Mode Accuracy (MA) of $P_F(x)$ w.r.t. $P^*(x)$, and defined as:

$$\mathrm{MA}(P_F(x), P^*(x)) = \min\left(\frac{\mathbb{E}_{P_F(x)}[R(x)]}{\mathbb{E}_{P^*(x)}[R(x)]}, 1\right). \tag{63}$$

We use dynamic programming (Malkin et al., 2022a) to compute $P_F(x)$ and the exact MA, D_{TV} and D_{JSD} between $P_F(x)$ and $P^*(x)$.

In this set of experiments, we focus not only on distribution modeling but also on mode discovery, where the goal is to uncover high-reward terminating states. In addition to RL, Sub-EB, and Sub-TB methods, we also consider augmenting Sub-TB and the offline Sub-EB (Algorithm 2) with the local search technique (Kim et al., 2023b) for designing $P_{\mathcal{D}}$, to explicitly promote the exploration of high-reward states during trajectory sampling. These variants are denoted Sub-TB-B and Sub-EB-B. As explained in the Related Works section and confirmed by the following experiment results, off-policy techniques that explicitly encourage exploration may not benefit distribution modeling. However, when the focus is on discovering modes of terminating states during training, these techniques can be valuable.

Mode discovery Results Here, we present the experimental results comparing different methods for **mode discovery**. We measure performance using Mode Number (MN), defined as the number of unique high-reward terminating states discovered during training. A terminating state is regarded as highly rewarded if its reward falls within the top 0.5% for the SIX6, QM9, and PHO4 datasets, and the top 0.01% for the sEH dataset. As depicted in Fig. 7, Sub-EB-B and Sub-TB-B can find

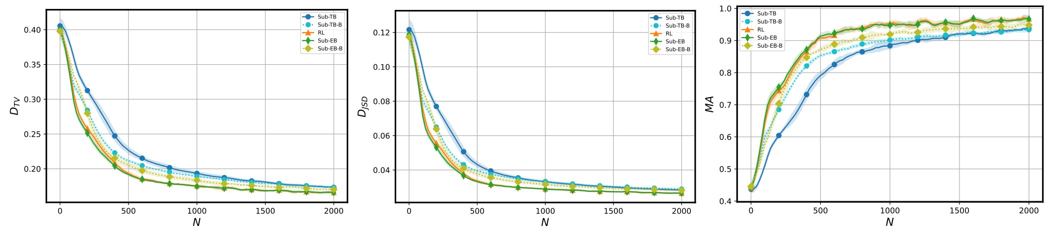

Figure 8: Plots of the mean and standard deviation values (represented by the shaded area) of D_{TV} (left), D_{JSD} (middle), and MA (right) for the SIX6 dataset, based on five randomly started runs for each method.

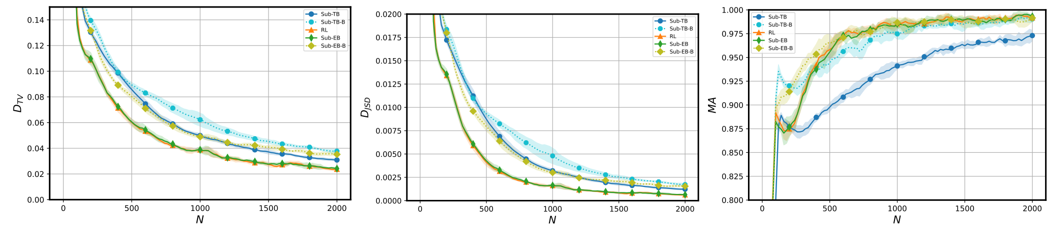

Figure 9: Plots of the mean and standard deviation values (represented by the shaded area) of D_{TV} (left), D_{JSD} (middle), and MA (right) for the QM9 dataset, based on five randomly started runs for each method.

a fixed number of unique modes faster and discover more unique modes within a fixed number of optimization iterations, despite a decline in distribution modeling performance on all datasets except PHO4. These results validate our offline policy-based training workflow and support our claim that the proposed Sub-EB objective enables the integration of offline sampling techniques.

Distribution Modeling Results We compare different training methods by their performance measured by MA, D_{TV} and D_{JSD}, as shown in Figs. 8, 9, 10, and 11 for SIX, QM9, PHO4, and sEH datasets respectively. It can be seen that Sub-EB performs slightly better than RL. This can be ascribed to the sufficient stability of RL in these experiments, rendering the advantages brought by the Sub-TB objective less obvious. Nevertheless, Sub-EB outperforms Sub-TB in terms of both convergence rate and final performance. They both leverage the balance conditions to learn an evaluation function V and a state flow function F, respectively. In principle, the key difference is that π_F and F are learned simultaneously in Sub-TB, whereas Sub-EB first learns V and then uses RL-like techniques to optimize π_F based on V. The results suggest that the balance conditions enable learning both V and F, and incorporating RL-like techniques into the balance-based framework can enhance the performance of traditional value-based training methods such as Sub-TB.

For the offline variants, it can be observed that Sub-EB-B performs slightly better than Sub-TB-B, but performs worse than Sub-EB for all datasets except PHO4. Combined with the mode discovery results in Fig. 7, this indicates that while offline techniques that encourage the high-rewarded terminating states is helpful for mode discovery, they may hinder accurate distribution modeling.

B.3 BN STRUCTURE LEARNING

A Bayesian Network is a probabilistic model, representing the joint distribution of N random variables, whose factorization is determined by the network structure x, which is a DAG graph. Accordingly, the distribution can be written as $P(y_1, \ldots, y_N) = \prod_{n=1}^{N} P(y_n|Pa_x(y_n))$, where $Pa_x(y_n)$ denote the parent nodes of y_n in graph x. Since any graph structure can be encoded as an adjacency matrix, the state space excluding the final state is defined as $\mathcal{S}\backslash\{s_f\} := \{s|\mathcal{C}(s) = 0, s \in \{0,1\}^{N\times N}\}$ where $\mathcal{C}$ corresponds to the acyclic graph constraint introduced by Deleu et al. (2022). It should be noted that each BN DAG $x \in \mathcal{X}$ corresponds to a state $s \in \mathcal{S}$ in the GFlowNet DAG $\mathcal{G}$. The generative process begins from the initial state $s_0 = 0^{N\times N}$, representing a graph without edges, and ends at $s_f := -1^{N\times N}$. For any $s \in \mathcal{S}\backslash s_f$, a transition $(s \to s')$

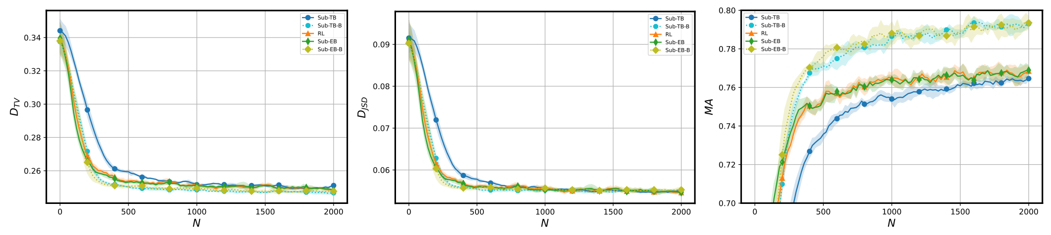

Figure 10: Plots of the mean and standard deviation values (represented by the shaded area) of D_{TV} (left), D_{JSD} (middle), and MA (right) for the PHO4 dataset, based on five randomly started runs for each method.

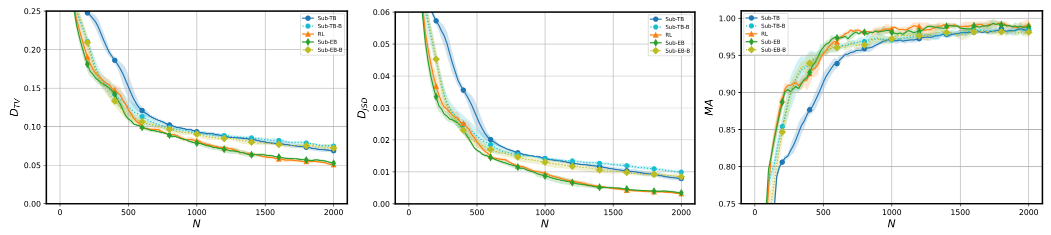

Figure 11: Plots of the mean and standard deviation values (represented by the shaded area) of D_{TV} (left), D_{JSD} (middle), and MA (right) for the sEH dataset, based on five randomly started runs for each method.

corresponds to adding an edge by flipping a zero entry in the adjacency matrix to one, provided that the resulting state s' remains acyclic. Alternatively, the generative process can be stopped by transitioning to s_f and returning s as the terminating graph structure. By definition, the $\mathcal{G}$ is not graded for this experiment set. Given an observed sample set $\mathcal{D}_y$ of $y_{1:N}$, the goal of structure learning is to approximate the posterior distribution $P(x|\mathcal{D}_y) \propto P(\mathcal{D}_y|x)P(x)$. In the absence of prior knowledge about x, the prior distribution $P(x)$ is often assumed to be uniform, reducing the task to maximizing the likelihood, $P(\mathcal{D}_y|x)(\propto P(x|\mathcal{D}_y))$.

Following Malkin et al. (2022b), the ground-truth graph structure x^* and the corresponding dataset $\mathcal{D}_y$ of size $|\mathcal{D}_y| = 10^3$ are simulated from Erdős–Rényi model (Deleu et al., 2022). We use BGe score (Kuipers et al., 2014) to assess generated graph structures, and define the reward $R(x) = (\Delta(x;\mathcal{D}_y)/C)^\beta$, where $\Delta(x;\mathcal{D}_y) = \text{BGe}(x;\mathcal{D}_y) - \text{BGe}(s_0;\mathcal{D}_y)$, $\beta = 10$ sharpens the reward function toward high-scoring structures, and $C = \Delta(x^*;\mathcal{D}_y)$ normalizes the reward so that $R(x^*) = 1$. The exact number of DAGs on n nodes, denoted as $a(n)$ satisfies (Robinson, 2006):

$$a(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)}\, a(n-k), \tag{64}$$

with $a(0) = 1$. For the ease of accessing distribution modeling performance, Malkin et al. (2022b) set the number of nodes to 5, resulting in about 2.92×10^4 possible DAGs. In this setup, the ground-truth DAG contains 5 edges. In addition to this small-scale case, we also consider two much larger cases with 10 and 15 nodes, corresponding to about 4.18×10^{18} and 2.38×10^{35} possible DAGs, respectively. We set the ground-truth DAGs to contain 10 and 15 edges in the two respective cases.

Experimental results For the small-scale case, we use dynamic programming (Malkin et al., 2022a) to explicitly compute $P_F(x)$ for learned π_F, and compute the exact D_{TV} and D_{JSD} between $P_F(x)$ and $P^*(x)$. In Fig. 13, the mean and standard deviation values of D_{TV} and D_{JSD} are plotted for five runs of Sub-TB, RL, and Sub-EB, respectively. It can be seen that Sub-EB performs better than both RL and Sub-TB. These results confirm our conclusion that the Sub-EB objective enables more reliable learning of the evaluation function V compared to the λ-TD objective, thereby improving the performance of the RL-like policy-based method. We also include results for Sub-EB-B and Sub-TB-B. These results further support our conclusion that explicitly encouraging exploration of high-reward regions may not be beneficial for distributional modeling performance.

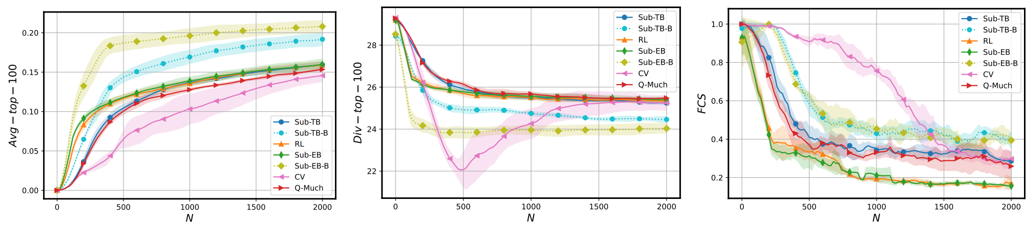

Figure 12: Plots of the mean and standard deviation values (represented by the shaded area) of average reward (left), diversity (middle) and FCS (right) of the top 100 unique candidate graphs over 15 nodes, based on five randomly started runs for each method.

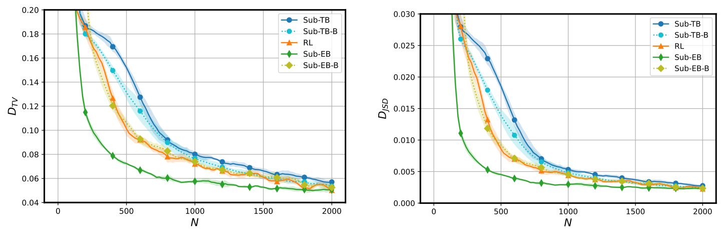

Figure 13: Plots of the mean and standard deviation values (represented by the shaded area) of D_{TV} (left) and D_{JSD} (right) for the BN learning structure learning task over 5 nodes, based on five randomly started runs for each method.

B.4 Molecular graph design

Each molecule (graph) is a composition of at most 8 building blocks, selected from $N = 105$ predefined molecular substructures provided by Bengio et al. (2021). Each block (graph node) has a list that contains at most M available atom indices for forming bonds (graph edges) with other blocks. Each block's list includes contains exactly one target atom and 0 to $M-1$ source atoms. A bond is formed by connecting a source atom in one block to a target atom in another. A graphical illustration of 4 exemplary blocks is shown in Fig. 14.
Following Bengio et al. (2021), we restrict the maximum number of blocks in a molecular graph to be $H = 8$. Each state $s \in \mathcal{S}$ in the generative process is represented as a $H \times 2$ matrix. The generative process begins from the initial state $s_0 = -1^{H\times 2}$, representing an empty graph, and ends at $s_f = 105^{H\times 2}$. For any state $s_h \in \mathcal{S}h$, the first column contains h integers drawn from $\{0, \ldots, N-1\}$, representing the indices of the building blocks present in the molecule. The second column contains $h-1$ integers drawn from $\{0, \ldots, H \times M - 1\}$, encoding the connectivity. For any $s \in \mathcal{S} \setminus (s_f, s_0)$, a transition $(s \to s')$ corresponds to a two-level action:

1. Selecting a building block to add, represented by an integer in $\{0, \ldots, N-1\}$.
2. Selecting a bonding site represented by an integer $k \in \{0, \ldots, H \times M - 1\}$. Supposing $k = (i-1) \times H + j$, this indicates that the target atom of the newly added block will be connected to the j-th source atom of the i-th block.

The generative process stops when we make the transition $(x \to s_f)$, when no available source atoms remain for forming bonds, or when the state reaches the limit of H blocks. According to the definition of the generative process, the $\mathcal{G}$ is not graded, meaning any state except s_f can be a terminating state $x \in \mathcal{X}$. Since there may be multiple types of bonds between a given pair of blocks, the size of the terminating state space is greater than $\frac{105!}{(105-8)!} \approx 10^{16}$. The Octanol–Water Partition Coefficient (LogP) and c-Jun N-terminal Kinase 3 (JNK3) scores provided in the *pyTDC* package are directly used as the reward functions $R(x)$. Since actions in this generative process involve two levels, we use separate neural networks to represent the policy for each action component. The log-probability of an action is computed as the sum of the log-probabilities output by the respective

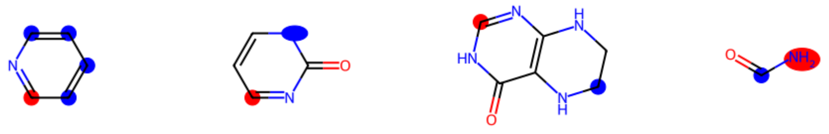

Figure 14: Four exemplary building blocks. Red and blue dots indicate the available atoms for bonding, where each bonding edge originates from a red dot and terminates at a blue dot.

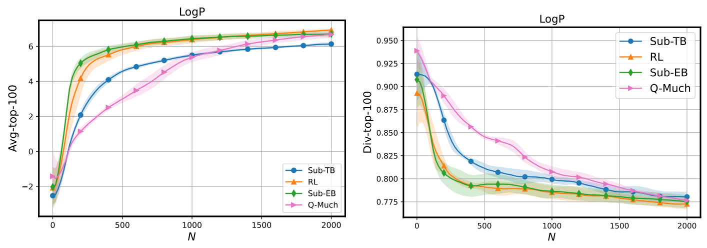

Figure 15: Plots of the mean and standard deviation values of average reward (left) and diversity (right, Tanimoto diversity of molecular Morgan fingerprints) of top 100 unique molecules based on LogP score.

networks for the two action levels. Finally, following Bengio et al. (2021), we use a batch size of 4 to emulate scenarios where querying the real-world molecule oracle is computationally expensive.

The experimental results are presented in Figs. 15 and 16. We report the average reward and diversity of the top 100 unique graphs which are discovered during the training process. The diversity of a set of molecules is computed as the average pairwise dissimilarity based on Tanimoto distance of Morgan fingerprints via the *pyTDC* package. For the LogP task, RL and Sub-EB achieve similar average rewards and convergence rate, both outperforming Sub-TB and Q-Much. All three methods exhibit comparable diversity. For the JNK3 task, Sub-EB achieves the highest average reward and demonstrates the fastest convergence among all methods. Its diversity is comparable to that of Sub-TB and higher than that of RL. While Q-Much achieves the highest diversity, its average reward is almost the lowest. Overall, these results indicate that Sub-EB achieves the best performance in terms of reward and convergence rate, while maintaining reasonable diversity. This confirms its effectiveness for large-scale molecule design tasks.

B.5 Gradient variance study

To measure gradient variance for different policy-based methods (CV, RL, Sub-EB), we follow the procedure of Madan et al. (2023). We first sample a large batch of $2^{10} = 1024$ trajectories $\{\tau_1, \dots, \tau_{1024}\}$ and compute their per-trajectory gradients $g_j^{(0)}$ w.r.t. the forward policy parameters. For each $k \in \{2, \dots, 9\}$, we construct 2^{10-k} disjoint sub-batches $\mathcal{B}_1^{(k)}, \dots, \mathcal{B}_{2^{10-k}}^{(k)}$, each containing exactly 2^k trajectories chosen in order from the large batch. The averaged sub-batch gradient is then defined as $g_i^{(k)} = \frac{1}{2^k} \sum_{j \in \mathcal{B}_i^{(k)}} g_j^{(0)}$. The full-batch reference gradient is $g^{(10)} = \frac{1}{1024} \sum_{j=1}^{1024} g_j^{(0)}$. This hierarchical batching is used solely for gradient-variance analysis; the actual training batch size remains unchanged. For each sub-batch gradient, we compute the cosine similarity $\mathrm{sim}(g_i^{(k)}, g^{(10)}) = \frac{g_i^{(k)} \cdot g^{(10)}}{\|g_i^{(k)}\| \|g^{(10)}\|}$, and estimate the variance at batch size 2^k via the mean similarity $\mathrm{MeanSim}(k) = \frac{1}{2^{10-k}} \sum_{i=1}^{2^{10-k}} \mathrm{sim}(g_i^{(k)}, g^{(10)})$. Higher $\mathrm{MeanSim}(k)$ indicates lower gradient variance. In Fig. 17, we report the mean similarity of CV, RL, and Sub-EB at iterations 500, 1000, and 1500. The results show that Sub-EB consistently achieves the lowest variance.

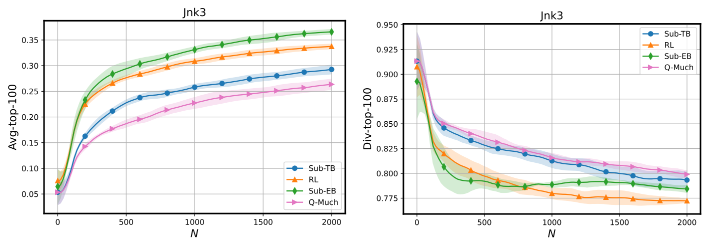

Figure 16: Plots of the mean and standard deviation values of average reward (left) and diversity (right, Tanimoto diversity of molecular Morgan fingerprints) of top 100 unique molecules based on JNK3 score.

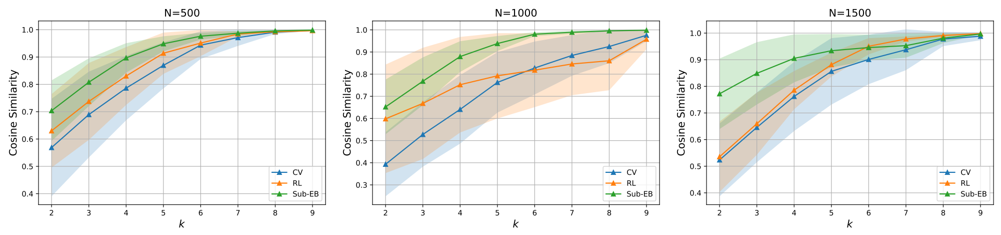

Figure 17: The mean cosine simiarity between small-batch 2^k and large batch (1024) of different methods at $k = \{2, \ldots, 9\}$ and training iteration 500, 1000 and 1500.

Method	Mean	Std
Sub-TB	1h 08m	1.09
Sub-TB-P	1h 14m	10.36
RL	**1h 03m**	0.92
RL-P	1h 27m	0.57
RL-M	1h 09m	1.78
Sub-EB	1h 17m	2.31
Sub-EB-P	1h 17m	1.49

Method	Mean	Std
Sub-TB	0h 52m	0.27
Sub-TB-P	0h 58m	7.92
RL	**0h 44m**	1.43
RL-P	1h 06m	2.21
RL-M	0h 58m	3.27
Sub-EB	0h 54m	2.66
Sub-EB-P	1h 11m	0.82

Method	Mean	Std
Sub-TB	1h 59m	1.53
Sub-TB-P	2h 19m	4.40
RL	**1h 27m**	7.37
RL-P	2h 03m	10.68
RL-M	2h 28m	9.58
Sub-EB	2h 04m	4.11
Sub-EB-P	2h 41m	15.46

Table 1: The mean and standard deviation (std) of the total runtimes for each method on the $64 \times 64 \times 64$ (left), $128 \times 128 \times 128$ (middle), and $256 \times 256 \times 256$ (right) grids. Here, 'h' denotes hours and 'm' denotes minutes, and all std values are reported in minutes.

Method	Mean	Std
Sub-TB	0h 05m	0.21
Sub-TB-B	0h 16m	0.67
RL	**0h 04m**	0.06
Sub-EB	**0h 04m**	0.34
Sub-EB-B	0h 17m	1.63

Method	Mean	Std
Sub-TB	**0h 04m**	1.10
Sub-TB-B	0h 12m	0.57
RL	**0h 04m**	0.12
Sub-EB	**0h 04m**	0.06
Sub-EB-B	0h 13m	0.71

Table 2: The mean and standard deviation (std) of the total runtimes for each method on the SIX6 and QM9 datasets. Here, 'h' denotes hours and 'm' denotes minutes, and all std values are reported in minutes.

Method	Mean Time (h:min)	Std (min)
Sub-TB	0h 19m	2.05
Sub-TB-B	0h 27m	0.42
RL	**0h 18m**	0.10
Sub-EB	**0h 18m**	0.15
Sub-EB-B	0h 30m	1.09

Method	Mean Time (h:min)	Std (min)
Sub-TB	**3h 30m**	10.23
Sub-TB-B	3h 46m	11.59
RL	3h 42m	13.78
Sub-EB	3h 41m	13.97
Sub-EB-B	3h 58m	7.57

Table 3: The mean and standard deviation (std) of the total runtimes for each method on the PHO4 and sEH datasets. Here, 'h' denotes hours and 'm' denotes minutes, and all std values are reported in minutes.

Method	Mean	Std
Sub-TB	0h 57m	1.11
Sub-TB-B	1h 23m	4.27
RL	0h 56m	0.46
Sub-EB	**0h 49m**	0.87
Sub-EB-B	1h 07m	1.43

Method	Mean	Std
Sub-TB	**0h 10m**	0.80
Sub-TB-B	0h 36m	1.82
RL	0h 11m	3.58
Sub-EB	**0h 10m**	0.91
Sub-EB-B	0h 31m	2.27

Method	Mean	Std
Sub-TB	0h 16m	1.28
Sub-TB-B	0h 42m	3.82
RL	**0h 12m**	1.79
Sub-EB	0h 14m	0.28
Sub-EB-B	0h 39m	1.13

Table 4: The mean and standard deviation (std) of the total runtimes for each method on the 5-node, 10-node and 15-node BN tasks. Here, ‘h’ denotes hours and ‘m’ denotes minutes, and all std values are reported in minutes. The runtimes for the 5-node cases are the largest due to the explicit computation of D_{TV} for performance comparison during training.