Better Training of GFlowNets with Local Credit and Incomplete Trajectories

Ling Pan [1] **Nikolay Malkin** [1] **Dinghuai Zhang** [1] **Yoshua Bengio** [1 2]

Abstract

Generative Flow Networks or GFlowNets are related to Monte-Carlo Markov chain methods (as they sample from a distribution specified by an energy function), reinforcement learning (as they learn a policy to sample composed objects through a sequence of steps), generative models (as they learn to represent and sample from a distribution) and amortized variational methods (as they can be used to learn to approximate and sample from an otherwise intractable posterior, given a prior and a likelihood). They are trained to generate an object x through a sequence of steps with probability proportional to some reward function $R(x)$ (or $\exp(-\mathcal{E}(x))$ with $\mathcal{E}(x)$ denoting the energy function), given at the end of the generative trajectory. Like for other RL settings where the reward is only given at the end, the efficiency of training and credit assignment may suffer when those trajectories are longer. With previous GFlowNet work, no learning was possible from incomplete trajectories (lacking a terminal state and the computation of the associated reward). In this paper, we consider the case where the energy function can be applied not just to terminal states but also to intermediate states. This is for example achieved when the energy function is additive, with terms available along the trajectory. We show how to reparameterize the GFlowNet state flow function to take advantage of the partial reward already accrued at each state. This enables a training objective that can be applied to update parameters even with incomplete trajectories. Even when complete trajectories are available, being able to obtain more localized credit and gradients is found to speed up training convergence, as demonstrated across many simulations.

[1]Mila – Québec AI Institute and Université de Montréal [2]CIFAR. Correspondence to: Ling Pan <penny.ling.pan@gmail.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Generative Flow Networks (GFlowNets) (Bengio et al., 2021a;b) are variants of reinforcement learning (RL) methods (Sutton & Barto, 2018) trained to generate an object $x \in \mathcal{X}$ through a sequence of steps, by learning a stochastic policy P_F with a training objective that would make it sample x with probability proportional to a reward function $R(x)$, with sufficient capacity and training iterations.

GFlowNets are related to MCMC methods (Metropolis et al., 1953; Hastings, 1970; Andrieu et al., 2003) that approximately sample from a distribution associated with a given energy function or unnormalized probability function, but GFlowNets exploit the ability of machine learning (ML) to generalize and to amortize the cost of sampling (which becomes quick), at the expense of training time. Unlike MCMC methods, they do not suffer from the mixing problem (Salakhutdinov, 2009; Bengio et al., 2013; 2021a), because they do not rely on a Markov chain that makes small local steps[1], and instead GFlowNets generate each sample independently. On the other hand, like RL methods, they rely on exploration and the generalization power of ML to guess and discover new modes of the reward function (or regions of low energy), thanks to underlying regularities in the given energy function.

They are also related to generative models (Kingma & Welling, 2013; Goodfellow et al., 2014; 2016; Ho et al., 2020) as they learn to represent and sample from a distribution, although they can either learn from a dataset (Zhang et al., 2022c;a), like typical deep generative models, or from an energy or reward function, like amortized variational methods (Kingma & Welling, 2014; Rezende et al., 2014) as they can be used to learn to approximate and sample from an otherwise intractable posterior, given a prior and a likelihood. As shown by Malkin et al. (2022b), GFlowNets can be seen as variants of amortized variational inference methods, using training objectives that are different from the usual KL objectives and enable off-policy learning and exploration methods that can avoid the factorization assumptions, importance reweighting, mode-seeking behavior and

[1]To mix between two modes, a Markov chain has to make the right sequences of moves to encounter a new far away and concentrated mode, and this can be very unlikely since it would require many low-probability moves.

other issues with standard variational inference methods.

The sequential process for generating the composite object is related to reinforcement learning (RL) methods (Sutton & Barto, 2018; Mnih et al., 2015; Lillicrap et al., 2015; Fujimoto et al., 2018). However, because they are trained to sample proportionally to the reward rather than to maximize it, GFlowNets tend to generate a greater diversity of solutions, which can be very appealing for tasks where such diversity is desirable (Jain et al., 2022a;b). Note that they are related but different from entropy-regularized RL (Haarnoja et al., 2017; 2018), as we will discuss in Section 3.

Yet, *previous learning objectives of GFlowNets* (Bengio et al., 2021a;b; Malkin et al., 2022a; Madan et al., 2022) *only learn from the reward of the terminal state, which is given only at the end of each trajectory*. Due to the delayed and possibly sparse feedback from the environment (especially for the more interesting cases where the target distribution concentrates probability mass), GFlowNets may suffer from the problem of inefficient credit assignment, due to long trajectories and a highly delayed reward. This paper proposes an approach to take advantage of incomplete reward signals that are available before the terminal state is reached, even allowing to benefit from incomplete trajectories, that do not reach a terminal state, i.e., without knowledge of the terminal reward.

As a motivating example, consider the possibility of using GFlowNets to sample abstract compositional objects akin to thoughts, as previously suggested (Bengio et al., 2021b; 2022). The human brain's working memory can only hold around a handful of symbols at a time, and although a sequence of such memory contents can correspond to a probabilistic inference (Baddeley, 1992; Cowan, 1999) (e.g., an interpretation for an image or a video in terms of latent variables that can explain it), this working memory bottleneck – see the work on the global workspace theory (Baars, 1993; Dehaene et al., 1998; Shanahan & Baars, 2005; Shanahan, 2006; 2010; 2012; Dehaene et al., 2017) – generally prevents us from forming a complete description of the full explanation of the input. This suggests that the brain can learn from partial inferences that do not contain a full specification of the latent variables that explain the given input.

However, variational methods and current GFlowNet training objectives (Bengio et al., 2021a; Malkin et al., 2022a; Madan et al., 2022; Bengio et al., 2021b) require complete specification of the latent explanation (i.e., a complete trajectory in the case of GFlowNets) before receiving a reward (such as how well the generated explanation fits the observed data and the prior). To perform the kind of computation associated with working memory and forming sequences of thoughts, GFlowNets would need to be modified to accommodate training trajectories that do not reach a terminal state. How could that even be possible if no reward is available before a terminal state is reached, i.e., before a full specification of the latent explanation is constructed?

In many cases, the reward function can actually be decomposed into a product of factors, where some of these factors can be accrued along the trajectory. A natural case is where the reward $R(x) = \exp(-\mathcal{E}(x))$ corresponds to an energy function $\mathcal{E}(x) = \sum_t \mathcal{E}_t$ that is additive with terms that can be obtained along the trajectory (e.g., at transitions t of the trajectory), as in GFlowNets over sets (Bengio et al., 2021b). In the case of a sparse factor graph that models the joint of latents and observed variables, each time the variables associated to a clique of the graph are specified, we can compute an energy term (Bengio et al., 2021b).

In this paper, we focus on the underlying technical question: how to train GFlowNets with incomplete trajectories? We find that the proposed solution also accelerates training even when complete trajectories are available. We propose Forward-Looking GFlowNets (FL-GFN), a novel formulation that exploits the ability to compute an energy value (even if incomplete) for intermediate states, in order to deliver a local credit signal and gradient as soon as a local reward factor is accrued. The code is publicly available at `https://github.com/ling-pan/FL-GFN`.

The main contributions are summarized as follows:

- We propose a novel GFlowNets formulation, FL-GFN, that can exploit the existence of a per-state or per-transition energy function.
- We show that with FL-GFN, we can still guarantee convergence, to fit and match the distribution corresponding to the given reward function, while credit information is available early. We show how FL-GFN can be trained from incomplete trajectories without access to terminal states.
- We conduct extensive experiments on the set generation and bit sequence generation tasks, which demonstrates the effectiveness of FL-GFN. It is also found to be scalable to the more complex and challenging molecule discovery task.

2. Background

2.1. GFlowNet Preliminaries

Let $G = (\mathcal{S}, \mathbb{A})$ be a directed acyclic graph (DAG), where $\mathcal{S}$ is the set of vertices (states), and $\mathbb{A} \subset \mathcal{S} \times \mathcal{S}$ is the set of edges (called actions). A unique *initial state* $s_0 \in \mathcal{S}$ is defined, without incoming edges. States without outgoing edges are called *terminal*,[2] and the set of terminal states is

[2]The alternative convention of Bengio et al. (2021b) defines terminal states as those with an outgoing edge to a designated sink state s_f. The two conventions are equivalent, as noted by Malkin

denoted by $\mathcal{X} \subset \mathcal{S}$. A sequence $\tau = (s_0 \to s_1 \to \cdots \to s_n)$, with $s_n \in \mathcal{X}$ and $(s_i \to s_{i+1}) \in \mathbb{A}$ for all i, is called a *complete trajectory*.

Let $R : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a nonnegative reward function on the set of terminal states. The goal of GFlowNets is to learn a stochastic policy P_F, specified as a distribution over the children of every nonterminal state in G, such that complete trajectories starting at s_0 and taking actions sampled from P_F terminate at $x \in \mathcal{X}$ with likelihood proportional to the reward $R(x)$. That is, if $P_F^\top(x)$ is the marginal likelihood that a complete trajectory $s_0 \to s_1 \to \cdots \to s_n$ sampled from P_F has $s_n = x$, then we desire $P_F^\top(x) \propto R(x)$. Note that $P_F^\top(x)$ is a sum of likelihoods of all trajectories leading to x:

$$P_F^\top(x) \stackrel{\text{def}}{=} \sum_{\tau=(s_0\to\cdots\to s_n=x)} P_F(\tau) = \sum_\tau \prod_{i=1}^n P_F(s_i|s_{i-1}),$$

which may be an intractably large sum. The GFlowNet forward policy P_F constructs objects sequentially, by modifying the partial object at each timestep, and transitions sampled from P_F should be thought of as incremental construction actions: unlike in standard RL, G has no cycles, which means that the same state cannot be visited twice within a trajectory. This can easily be achieved by including a time stamp in the state (Bengio et al., 2021b).

The action policy P_F – parameterized as a neural network $P_F(s_i|s_{i-1};\theta)$ taking a state s_{i-1} as input and producing a distribution over the children s_i of s_{i-1} – is the main object of GFlowNet training.

2.2. Training Criteria for GFlowNets

There are several auxiliary quantities that can be either introduced in the training process or computed after a policy has been trained, depending on the training objective chosen. We summarize three objectives that are relevant in this paper as follows:

Detailed balance (DB). Two auxiliary quantities are learned: a scalar *state flow function* $F(s;\theta)$ and a *backward policy* $P_B(s|s';\theta)$, which is a distribution over the parents s of every noninitial state s'. We set $F(x) = R(x)$ for terminal states x. The DB constraint (that will only be approximately achieved, thanks to a training objective) enforces that for any action $s \to s'$, we have

$$F(s;\theta)P_F(s'|s;\theta) = F(s';\theta)P_B(s|s';\theta). \tag{1}$$

For a given forward policy P_F, there is a unique backward policy P_B and state flow function F that satisfy Eq. (1) (Bengio et al., 2021b). In practice, the constraint is enforced by taking gradient steps with respect to θ on the square of the log-ratio between the two sides of Eq. (1), where the edge $s \to s'$ is chosen from a training policy. To improve exploration and prevent the agent from getting trapped around a few modes of R (Jain et al., 2022a; Pan et al., 2022), the training policy is usually chosen as a tempered version of the forward policy or its mixture with a uniform policy, i.e., $\pi_\theta = \epsilon U + (1-\epsilon)P_F$ that resembles ϵ-greedy exploration in RL. Bengio et al. (2021a) prove that if the training policy π_θ has full support and the expected loss is minimized globally, then $P_F^\top(x) \propto R(x)$.

Trajectory balance (TB). With this GFlowNet objective, the learned auxiliary quantities are a backward policy P_B, as above, and a scalar Z_θ (typically parametrized in the log-domain as $\log Z_\theta$). For any complete trajectory $\tau = (s_0 \to s_1 \to \cdots \to s_n = x)$, the TB constraint is

$$Z_\theta \prod_{i=1}^n P_F(s_i|s_{i-1};\theta) = R(x) \prod_{i=1}^n P_B(s_{i-1}|s_i;\theta). \tag{2}$$

Satisfaction of this constraint for all complete trajectories also implies that $P_F^\top(x) \propto R(x)$ (Malkin et al., 2022a). The constraint can again be enforced by optimizing the squared log-ratio between the left and right hand sides of Eq. (2).

Subtrajectory balance (SubTB). The parametrization is the same as in DB. The subtrajectory balance (SubTB) constraint applies to partial trajectories $s_m \to \cdots \to s_n$, where s_m and s_n are not necessarily initial and final:

$$\begin{aligned} &F(s_m;\theta) \prod_{i=m+1}^n P_F(s_i|s_{i-1};\theta) \\ =&F(s_n;\theta) \prod_{i=m+1}^n P_B(s_{i-1}|s_i;\theta). \end{aligned} \tag{3}$$

Special cases for this constraint are DB (when the partial trajectory has length 1) and TB (when the trajectory is complete, noting that $F(x;\theta) = R(x)$ when x is terminal). Satisfaction of the SubTB constraint for all partial trajectories of a given length does not necessarily imply sampling proportionally to the reward, but its satisfaction for partial trajectories of all lengths (thus including complete trajectories and terminal states) implies both the DB and TB constraints hold and is thus sufficient to guarantee $P_F^\top(x) \propto R(x)$. Madan et al. (2022) empirically tested the training objective based on the SubTB constraint, which reduces the gradient variance of the TB objective.

3. Related Work

GFlowNets. There have been many recent efforts in applying GFlowNets in a number of settings, e.g., molecule discovery (Bengio et al., 2021a), sequence design (Jain

et al. (2022a), but ours has the advantage of allowing simpler notation for the expressions needed in this paper.

et al., 2022a), Bayesian structure learning (Deleu et al., 2022; Nishikawa-Toomey et al., 2022), and providing theoretical understandings of GFlowNets (Bengio et al., 2021b; Malkin et al., 2022b; Zimmermann et al., 2022; Zhang et al., 2022a; Lahlou et al., 2023). Given its practical importance, several studies also emerged (Bengio et al., 2021b; Malkin et al., 2022b; Madan et al., 2022) to improve the learning efficiency of GFlowNets since the proposal of the flow matching learning objective initially proposed by Bengio et al. (2021a). It can also be jointly trained with an energy or reward function (Zhang et al., 2022c). Pan et al. (2022) introduce intrinsic exploration rewards into GFlowNets in an additive way to help exploration in sparse reward tasks. There have also been recent efforts in extending GFlowNets to stochastic environments with stochasticity in transition dynamics (Pan et al., 2023) and rewards (Zhang et al., 2023). Yet, GFlowNets could suffer from inefficient credit assignment since they only learn from a trajectory reward provided at terminal states, which poses a critical challenge to the attribution of credit on each of the actions in a trajectory, especially those in the early parts of the trajectory. Previous learning objectives of GFlowNets require access to terminal states and need to be trained with complete trajectories, which can be infeasible when the composite terminal states x have considerably large sizes.

Reinforcement Learning (RL). RL agents typically aim to learn a reward-maximizing policy, instead of learning to sample in proportion to the reward function. Soft Q-learning (Haarnoja et al., 2017) introduces entropy regularization in its objective (Haarnoja et al., 2018; Zhang et al., 2022b) and learns a stochastic energy-based policy. However, it can perform badly in general (non-tree) DAGs as shown in Bengio et al. (2021a), since there can be a potentially large number of trajectories leading to the same terminal state, and some terminal states (corresponding to longer trajectories) may thus have exponentially more trajectories into them, which biases learning in favor of longer trajectories. In episodic RL settings such is the case with GFlowNets, the agent only receives a trajectory feedback at the end of each trajectory, which can hinder learning efficiency in long-horizon problems (Ren et al., 2022).

4. Proposed Method

Previous GFlowNets learning objectives, e.g., detailed balance (DB) (Bengio et al., 2021b), trajectory balance (TB) (Malkin et al., 2022a), and sub-trajectory balance (SubTB) (Madan et al., 2022) therefore require learning with complete trajectories. This is because they only learn from the energy of the terminal state, i.e., $\mathcal{E}(x)$, and therefore need to visit terminal states x to obtain informative learning signals, as illustrated in Figure 1(a). The requirement of learning from complete trajectories is a critical limitation when the trajectory length is long or the final composite object is complex, as motivated from the above working memory bottleneck (Baars, 1993) example and the situation of inferring the latent explanation for a rich input such as a complex image or a video (Bengio et al., 2022). In addition, learning from a highly delayed reward also makes it challenging for agents to properly associate their actions to future rewards, and thus hinders the efficiency of learning and credit assignment (Ren et al., 2022).

In this section, we introduce our approach, Forward-looking GFlowNets or FL-GFN for short to take advantage of the computability of energy at intermediate states, or equivalently, an additive energy decomposition. FL-GFN can take intermediate reward signals into account, and thus obtains faster learning and enables learning from incomplete trajectories. We also theoretically justify the proposed learning objective and discuss its connections to existing approaches. FL-GFN relies on the following assumption:

Assumption 4.1. The terminal state energy function $\mathcal{E} : \mathcal{X} \to \mathbb{R}$ can be extended to the set of all states, i.e., there exists $\mathcal{E} : \mathcal{S} \to \mathbb{R}$.

As a consequence, we can also define an energy function over transitions:

$$\mathcal{E}(s \to s') \stackrel{\text{def}}{=} \mathcal{E}(s') - \mathcal{E}(s). \tag{4}$$

Similarly, we can define the energy differential associated with all trajectories from s to x as

$$\mathcal{E}(s \to x) \stackrel{\text{def}}{=} \mathcal{E}(x) - \mathcal{E}(s). \tag{5}$$

We obtain that the energy of a state s_t (terminal or not) can be written additively over transitions for any trajectory $s_0 \to s_1 \to \cdots \to s_n$ if we define $\mathcal{E}(s_0) \stackrel{\text{def}}{=} 0$:

$$\mathcal{E}(s_t) = \sum_{i=1}^{n} \mathcal{E}(s_{i-1} \to s_i). \tag{6}$$

4.1. Sources of Partial Energies

We now describe two common settings where energies for incomplete states may arise.

Purely additive energies. We can start from a given per-transition energy function $\mathcal{E}(s \to s')$ and apply the above to define a per-state energy function. For example, this applies to the construction of a set whose log-reward is a sum of terms depending on its elements (cf. the Set-GFN construction in Bengio et al. (2021b), where a state corresponds to a set of elements, a transition corresponds to inserting a new element in the current set, and an energy term is associated with each element in s). Another scenario is the sampling of posteriors over latent variables with compositional structure (Hu et al., 2023), where our proposed methodology is applied to parse trees under probabilistic context-free grammars.

State spaces extending the target space. A second motivating example is the case where all states belong to the same space (e.g., a vector space or a space of structured objects that can be processed by a neural network) and the energy can be calculated for any state s using the same function that is applied in the computation of the reward for terminal states x. In this case, the energy $\mathcal{E}(s \to s')$ measures the stepwise marginal improvement in the log-reward. Note that this formulation applies even if s and s' are not complete objects from which termination is allowed, as long as an extension of the reward function to nonterminal states can be evaluated (e.g., the molecule generation domain as studied in Section 5.3).

4.2. Forward-looking GFlowNets (FL-GFN)

Consider the flow at s and exploit Assumption 4.1 to rewrite the terminal energy $\mathcal{E}(x)$ as $\mathcal{E}(s) + \mathcal{E}(s \to x)$:

$$F(s) = \sum_{x \geq s} P_B(s|x) e^{-\mathcal{E}(x)} \tag{7}$$

$$= e^{-\mathcal{E}(s)} \sum_{x \geq s} P_B(s|x) e^{-\mathcal{E}(s \to x)}. \tag{8}$$

Here $P_B(s|x)$ denotes the probability of reaching the intermediate state s from the terminal state x via the one-step backward policy $P_B(s|s')$. The idea is to take advantage of the fact that we already know $\mathcal{E}(s)$ when we have reached state s and that we can thus factor it out of the right-hand side, as above. With this in mind, we define the **forward-looking flow** as

$$\widetilde{F}(s) \stackrel{\text{def}}{=} e^{\mathcal{E}(s)} F(s) = \sum_{x \geq s} P_B(s|x) e^{-\mathcal{E}(s \to x)}, \tag{9}$$

i.e., $\widetilde{F}(s)$ only contains a sum over terms with energies of transitions to come (hence the name of forward-looking flow). Using this, we can write a variant of the detailed balance constraint (Eq. (1)), called the FL-DB constraint, expressed in terms of forward-looking flows:

$$\widetilde{F}(s) P_F(s'|s) = \widetilde{F}(s') P_B(s|s') e^{-\mathcal{E}(s \to s')}. \tag{10}$$

Note that this constraint can be extended in the same way that DB was extended to SubTB (as we will study in Section 5.1.3). Based on the resulting dense supervision (where actual energy/reward signals are available at intermediate steps), we can hypothesize that more efficient credit assignment is achieved. In addition, it also makes it possible for GFlowNets to learn based on incomplete trajectories without access to terminal states, so long as the training sequences of intermediate energies over incomplete trajectories contain enough information to generalize over the energies to be expected downstream.

Algorithm 1 Training Step of FL-GFN

1: Initialize forward and backward policies P_F, P_B, and the energy-to-go flow $\tilde{F}$ with parameters θ
2: **for** each transition $s \to s'$ sampled from a training trajectory **do**
3: Measure the transition energy $\mathcal{E}(s \to s')$
4: Update θ by $\theta - \eta \nabla_\theta \mathcal{L}(s_t, s_{t+1})$ as per Eq. (11)
5: **end for**

Implementation. In practice, we train the FL-GFN based on the FL-DB constraint in Eq. (10), following Algorithm 1, to minimize the corresponding loss function over transitions $s \to s'$ in the log-space:

$$\mathcal{L}(s, s') = \Big(\log \widetilde{F}(s;\theta) + \log P_F(s'|s;\theta) - \log \widetilde{F}(s';\theta) - \log P_B(s|s';\theta) + \mathcal{E}(s \to s') \Big)^2. \tag{11}$$

An alternative implementation is to make the following substitution and optimize for the DB constraint Eq. (1):

$$\log F(s) = -\mathcal{E}(s) + \log \tilde{F}(s;\theta), \tag{12}$$

where $\log \tilde{F}(s;\theta)$ is a learned model.

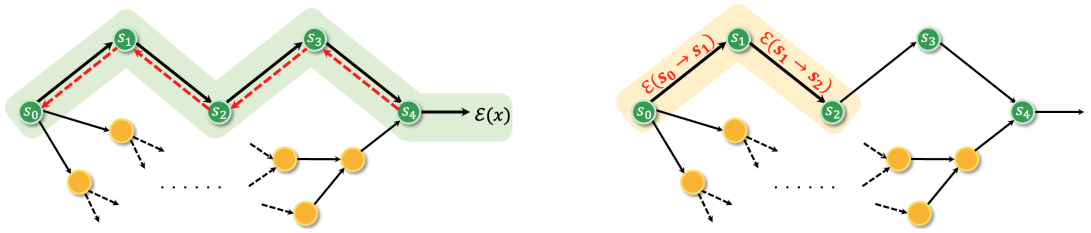
Theoretical justification. We now theoretically justify that if we reach a global minimum of the expected value of the FL-GFN loss (Eq. (11)), then the FL-GFN samples from the target reward distribution correctly. The proof can be found in Appendix A.

Proposition 4.2. *Suppose that Assumption 4.1 is satisfied, which makes it possible to define a forward-looking flow as per Eq. (9). If $\mathcal{L}(s, s') = 0$ for all transitions, then the forward policy $P_F(s'|s;\theta)$ samples proportionally to the reward function.*

Next, we discuss its connection with existing methods in the GFlowNets literature.

Connection with existing methods. The forward-looking flow $\widetilde{F}(s;\theta)$ in Eq. (9) can also be interpreted as a new parametrization of the state flow function as in Eq. (12). The case of $\mathcal{E}(s) \equiv 0$ for non-terminal states s corresponds to regular DB (or SubTB) training, in which learning signals come only from the reward at the end of a trajectory and the flow function is learned directly as a neural network, since $F(s) = \widetilde{F}(s)$. However, in many cases, there is a natural $\mathcal{E}(s)$ available as discussed above and shown in the experiments below.

If termination is permitted from any state s (into terminal state $s^\top$) with reward $e^{-\mathcal{E}(s)}$ and the GFlowNet constraints are satisfied, we have $F(s)P_F(s^\top|s) = R(s^\top) = e^{-\mathcal{E}(s)}$.



(a) (b)

Figure 1. (a) Reward propagation in previous learning objectives of GFlowNets and (b) in FL-GFN, with additive energy terms for each transition. Note that (a) requires learning based on complete trajectories since it only learn from the terminal energy $\mathcal{E}(x)$, while (b) makes it possible to learn from short incomplete trajectories.

Substituting Eq. (12), this simplifies to $\log \widetilde{F}(s;\theta) = -\log P_F(s^\top|s)$. The loss introduced by Deleu et al. (2022) can be considered as the DB loss with the parametrization in Eq. (12) and directly setting $\log \widetilde{F}(s;\theta) = -\log P_F(s^\top|s)$.

5. Experiments

We conducted extensive experiments to investigate the following key questions and understand the effectiveness of the proposed approach: i) How does the forward-looking approach compare against previous GFlowNet learning objectives in terms of learning efficiency and final performance? ii) Can it learn given incomplete trajectories only when it does not have access to terminal states? iii) Can it be applied to different GFlowNet learning objectives? iv) How does it work on more complex and challenging tasks?

5.1. Set Generation

5.1.1. Experimental Setup

We first conducted a series of experiments on a didactic set generation task with set GFlowNets (Bengio et al., 2021b) to understand the effect of the forward-looking approach. The agent generates a set of size $|S|$ from $|U|$ distinct elements sequentially. At each timestep, the agent chooses to add an element from U to the current set s (the GFlowNet state) without repeating elements, and gets an energy term for adding the element (a fixed value for each element). The task terminates when there are exactly $|S|$ elements in the set s, and the total energy for constructing s is $\mathcal{E}(x) = \sum_{t\in\tau} \mathcal{E}(t)$, where τ is the sampled trajectory and $t \in \tau$ is a transition. We consider different scales of the set generation task including small, medium, and large with increasing sizes of $|S|$ and $|U|$. More details of the experimental setup can be found in Appendix B.1.

5.1.2. Performance Comparison

Following Bengio et al. (2021a), we evaluate the methods in terms of the average reward of the unique top-100 sampled candidates (from all the samples generated during training) and the number of modes discovered by each algorithm, so as to measure both performance and diversity. We compare FL-GFNs with previous GFlowNets learning objectives including detailed balance (DB), trajectory balance (TB), and subtrajectory balance (SubTB).

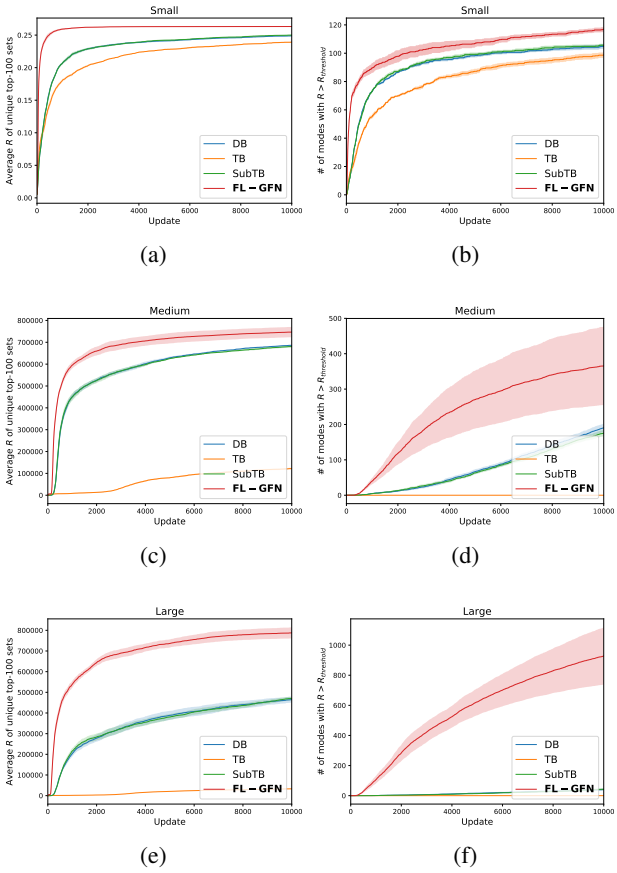
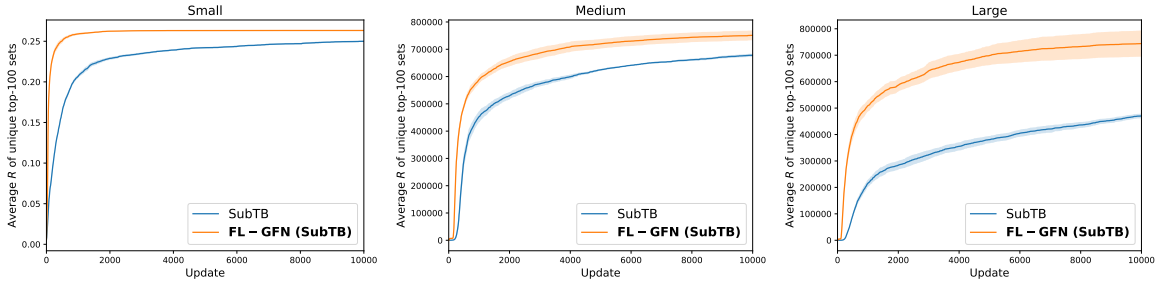
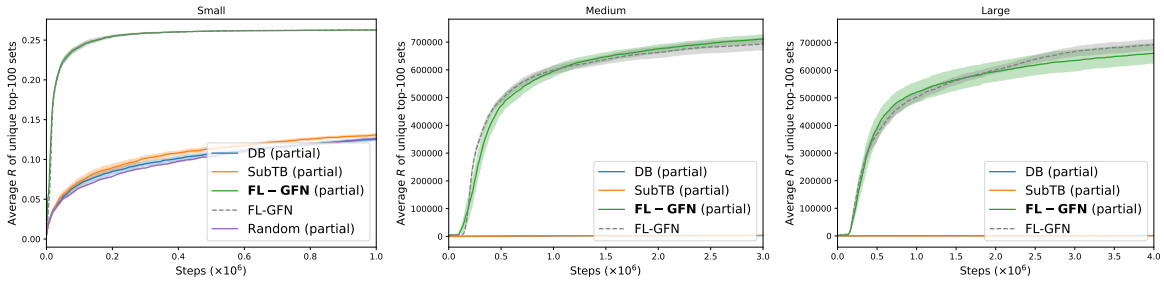


Figure 2. Comparison on the set generation task with (a, b) small, (c, d) medium, and (c, d) large sets. The first column shows the advantage of FL-GFN in terms of the average reward of the unique top-100 sets, the second column in terms of the number of modes discovered by each method, i.e., diversity.



(a) (b) (c)

Figure 3. Performance comparison of FL-GFN (SubTB) and SubTB in the set generation task with different problem sizes including (a) small (b) medium (c) large. FL-GFN (SubTB)converges faster and to better solutions, especially for larger sets.



(a) (b) (c)

Figure 4. Performance comparison of baselines trained with incomplete trajectories in the set generation task with different scales including (a) small (b) medium and (c) large. The advantage of FL-GFN increases with the length of trajectories.

The first column in Figure 2 demonstrates the quality of generated sets in the set generation task with different problem sizes including small, medium, and large in each row. As shown, the forward-looking approach significantly outperforms previous baselines including DB, TB, and SubTB in training efficiency and quality of the solutions, with a more significant gap in larger-scale environments, presumably because of FL-GFN's more efficient credit assignment mechanism. We also observe that the performance of TB degrades with longer trajectories in the larger-scale set generation tasks, presumably due to the larger variance (Madan et al., 2022), and SubTB performs closely to DB. The second column in Figure 2 illustrates the number of modes with rewards above a threshold discovered by each method, diversity of the good solutions, and FL-GFN discovers many more high-reward modes faster.

5.1.3. Applicability to Other Objectives

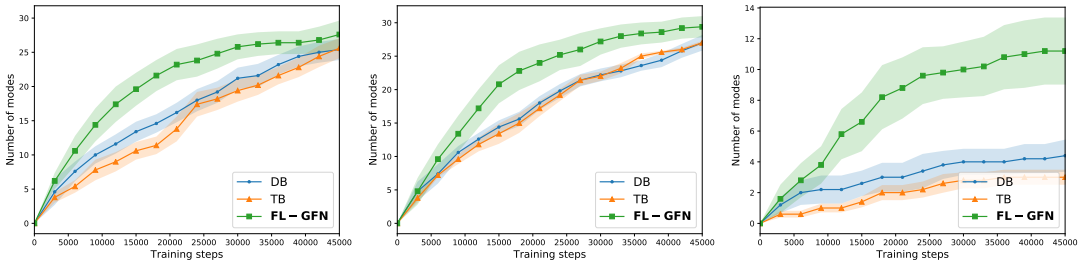
FL-GFN is versatile and can be applied to different learning objectives of GFlowNets except for TB, so long as Assumption 4.1 is satisfied. Here, we investigate the applicability of FL-GFN with the SubTB constraint (Madan et al., 2022), yielding the following FL-SubTB constraint (and the corresponding training objective) as in Eq. (13) following Section 4 and the constraint of SubTB in Eq. (3), where $\widetilde{F}$ denotes the forward-looking flow, and $\mathcal{E}(s \to s')$ denotes the intermediate transition energy.

$$\begin{aligned}&\widetilde{F}(s_i;\theta)\prod_{t=i}^{j-1}P_F(s_{t+1}|s_t;\theta)\\=&\widetilde{F}(s_j;\theta)\prod_{t=i}^{j-1}P_B(s_t|s_{t+1};\theta)\prod_{t=i}^{j-1}e^{-\mathcal{E}(s_t\to s_{t+1})}\end{aligned} \tag{13}$$

As demonstrated in Figure 3, FL-GFN (SubTB) consistently outperforms SubTB with an improved average reward of unique top-100 sets.

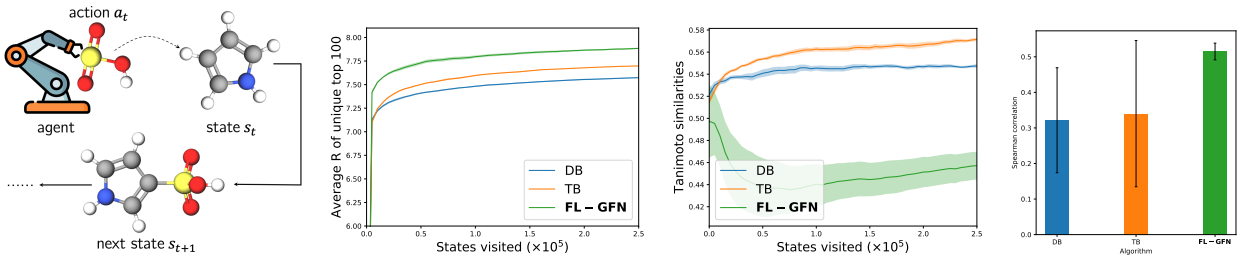
5.1.4. Learning with Incomplete Trajectories

We now investigate the ability of FL-GFN when given only incomplete trajectories, without access to terminal states. Here, the length of incomplete trajectories is uniformly distributed in $[1, |S|-1]$, with $|S|$ the length of complete trajectories. We compare the performance of FL-GFN trained with incomplete trajectories only with its counterpart that is learned with complete trajectories. We include ordinary DB and SubTB approaches in the comparison for completeness



(a) (b) (c)

Figure 5. The number of modes discovered by each method in the bit sequence generation task with increasing lengths of the sequences. Different problem sizes are presented from left to right: (a) normal (b) long (c) very long. The advantage of FL-GFN increases with the size of the problem.



(a) (b) (c) (d)

Figure 6. Results on molecule generation task. (a) Illustration of the GFlowNet generation pipeline. (b) Average reward of the top-100 molecule candidates, showing faster and better training of FL-GFN. (c) The Tanimoto similarity which quantifies diversity (lower is better), showing greater diversity with FL-GFN. (d) The correlation between model log likelihood and the log rewards computed on a held-out set, larger with FL-GFN.

(with the "partial" tag in the figure), although they cannot achieve informative learning signals without access to terminal states and rewards (and the apparent improvements in the small set generation in Figure 4(a) are only due to randomly sampling more trajectories and finding some good ones). Note that TB cannot be implemented with incomplete trajectories, as the learning objective of TB involves the terminal reward.

Figure 4 shows the average reward of unique top-100 sets discovered by each method when learned with incomplete trajectories, where the x-axis corresponds to the number of state transitions (interactions with the environment). Unlike the earlier training methods, FL-GFN (partial) is able to learn well at different scales. More interestingly, it performs closely to its counterpart trained with full trajectories, validating the ability to learn from only incomplete trajectories.

5.2. Bit Sequence Generation

We consider the bit sequence generation task from Malkin et al. (2022a). At each time step, the agent chooses to append a k-bit "word" (with $k = 4$). The reward function has modes at a predefined fixed set M of sequences (Malkin et al., 2022a): $\mathcal{E}(x) = \min_{y \in M} d(x, y)$, where d is the edit distance. Intermediate transition energies for FL-GFN are obtained by applying the above energy function on intermediate states as well. We consider increasing sequence lengths, as detailed in Appendix B.2. We find that FL-GFN learns more efficiently in more complex tasks by comparing it with DB and TB, which are the most competitive baselines in this task as shown by Malkin et al. (2022a), and evaluate the methods in terms of the number of modes discovered.

Results. Figure 5 shows the number of modes discovered by each method during training in bit sequence generation with different lengths. FL-GFN outperforms baselines in learning efficiency and also discovers more modes, with a more significant gap for longer sequences, suggesting that this is due to more efficient credit assignment.

5.3. Molecule Generation

We apply FL-GFN to the more challenging and larger-scale molecule generation task (Bengio et al., 2021a; Malkin et al., 2022a) as illustrated in Figure 6(a), with a large state space (about 10^{16}) and action space (from 100 to 2000). A molecule is represented by a graph whose nodes are taken from a vocabulary of molecular building blocks. We want to discover diverse and high-quality molecules with low binding energy to the soluble epoxide hydrolase (sEH) protein, where the binding energy is computed by a pretrained proxy

model, which can be applied to intermediate graphs. Actions consist of adding a molecular block at a selected node in a molecular graph, or going to a terminal state to stop generation. TB and DB baselines are implemented based on existing GFlowNet open-source code[3]. We report the mean and standard variance for each algorithm with three random seeds as in Malkin et al. (2022a). More details of this setup are in Appendix B.3.

Results. Figure 6(b) shows the advantage of FL-GFN in terms of faster and better training, with the average reward of the top-100 unique molecules discovered by each method, which evaluates the quality of generated solutions, while Figure 6(c) highlights diversity (as measured by the Tanimoto similarities, lower is better), following Bengio et al. (2021a). Visualization of the top-3 molecules (and their diversity) discovered by each method in a run is shown in Appendix (Figure 7). Figure 6(d) shows the improvement brought by FL-GFN to the Spearman correlations between the log-reward $\log R(x)$ and the log-sampling probability $\log P_F^{\mathrm{T}}(x)$ (following the methodology from Bengio et al. (2021a)) on a set of test molecules. The results demonstrate consistent and significant performance improvement and faster training of FL-GFN in the more complex and challenging molecule generation task.

6. Conclusion

In this paper, we propose Forward-looking GFlowNets (FL-GFN), a new GFlowNet formulation which exploits the computability of a per-state or per-transition energy function. FL-GFN can be trained to sample proportionally from the target reward distribution, while speeding up training due to its more efficient credit assignment mechanism. It can even be trained given incomplete trajectories when it does not have access to terminal states, which was a requirement for previous GFlowNet learning objectives. We conduct extensive experiments to demonstrate the effectiveness of FL-GFN, which can scale to complex and challenging tasks, such as molecular graph generation.

Acknowledgments

The authors acknowledge funding from CIFAR, Genentech, Samsung, and IBM. We would also like to thank Moksh Jain, Edward Hu, Cristian Dragos Manta, and Hadi Nekoei for valuable discussions.

References

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

Baars, B. J. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

Baddeley, A. Working memory. *Science*, 255(5044):556–559, 1992.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021a.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. Better mixing via deep representations. *International Conference on Machine Learning (ICML)*, 2013.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E., Tiwari, M., and Bengio, E. GFlowNet foundations. *arXiv preprint 2111.09266*, 2021b.

Bengio, Y., Malkin, N., and Jain, M. The GFlowNet Tutorial. https://milayb.notion.site/The-GFlowNet-Tutorial-95434ef0e2d94c24aab90e69b30be9b3, 2022.

Cowan, N. An embedded-processes model of working memory. 1999.

Dehaene, S., Kerszberg, M., and Changeux, J.-P. A neuronal model of a global workspace in effortful cognitive tasks. *Proceedings of the national Academy of Sciences*, 95(24): 14529–14534, 1998.

Dehaene, S., Lau, H., and Kouider, S. What is consciousness, and could machines have it? *Science*, 358(6362): 486–492, 2017.

Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Neural Information Processing Systems (NIPS)*, pp. 2672–2680, 2014.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

[3] https://github.com/GFNOrg/gflownet/tree/trajectory_balance/mols

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. 1970.

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

Hu, E. J., Malkin, N., Jain, M., Everett, K., Graikos, A., and Bengio, Y. GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*, 2023.

Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F., Ekbote, C., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. Biological sequence design with GFlowNets. *International Conference on Machine Learning (ICML)*, 2022a.

Jain, M., Raparthy, S. C., Hernandez-Garcia, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. Multi-objective gflownets. *arXiv preprint arXiv:2210.12765*, 2022b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-García, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A., Bosc, T., Bengio, Y., and Malkin, N. Learning GFlowNets from partial episodes for improved convergence and stability. *arXiv preprint 2209.12782*, 2022.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2022a.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nishikawa-Toomey, M., Deleu, T., Subramanian, J., Bengio, Y., and Charlin, L. Bayesian learning of causal structure and mechanisms with GFlowNets and variational bayes. *arXiv preprint 2211.02763*, 2022.

Pan, L., Zhang, D., Courville, A., Huang, L., and Bengio, Y. Generative augmented flow networks. *arXiv preprint 2210.03308*, 2022.

Pan, L., Zhang, D., Jain, M., Huang, L., and Bengio, Y. Stochastic generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2023.

Ren, Z., Guo, R., Zhou, Y., and Peng, J. Learning long-term reward redistribution via randomized return decomposition. *International Conference on Learning Representations (ICLR)*, 2022.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning (ICML)*, 2014.

Salakhutdinov, R. Learning in markov random fields using tempered transitions. *Neural Information Processing Systems (NIPS)*, 2009.

Shanahan, M. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and cognition*, 15(2):433–449, 2006.

Shanahan, M. *Embodiment and the inner life: Cognition and Consciousness in the Space of Possible Minds*. Oxford University Press, USA, 2010.

Shanahan, M. The brain's connective core and its role in animal cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1603):2704–2714, 2012.

Shanahan, M. and Baars, B. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, 2005.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Neural Information Processing Systems (NIPS)*, 2017.

Zhang, D., Chen, R. T. Q., Malkin, N., and Bengio, Y. Unifying generative models with GFlowNets. *arXiv preprint 2209.02606*, 2022a.

Zhang, D., Courville, A. C., Bengio, Y., Zheng, Q., Zhang, A., and Chen, R. T. Q. Latent state marginalization as a low-cost approach for improving exploration. *ArXiv*, abs/2210.00999, 2022b.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. *International Conference on Machine Learning (ICML)*, 2022c.

Zhang, D., Pan, L., Chen, R. T., Courville, A., and Bengio, Y. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*, 2023.

Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. A variational perspective on generative flow networks. *arXiv preprint 2210.07992*, 2022.

A. Proof of Proposition 4.2

Proposition 4.2 *Suppose that Assumption 4.1 is satisfied, which makes it possible to define a forward-looking flow as per Eq. (9). If $\mathcal{L}(s, s') = 0$ for all transitions, then the forward policy $P_F(s'|s;\theta)$ samples proportionally to the reward function.*

Proof. We give a simple proof by reduction to the DB training theorem (Bengio et al., 2021b), which states that if for some state flow function F and policies P_F and P_B the DB constraint as in Eq. (1) holds for all transitions $s \to s'$, then P_F samples proportionally to the reward.

Suppose $\widetilde{F}$, P_F, and P_B satisfy Eq. (10). Define a state flow function $\hat{F}$ by $\hat{F}(s) \stackrel{\text{def}}{=} e^{-\mathcal{E}(s)}\widetilde{F}(s)$. By direct algebraic manipulation of Eq. (10), we obtain, for all transitions $s \to s'$,

$$\begin{aligned}\hat{F}(s)P_F(s'|s) &= \frac{e^{-\mathcal{E}(s)}}{e^{-\mathcal{E}(s')}}\hat{F}(s')P_B(s|s')e^{-\mathcal{E}(s\to s')} \\ &= \hat{F}(s')P_B(s|s').\end{aligned}$$

Therefore, the triple $(\hat{F}, P_F, P_B)$ satisfies DB, implying that P_F samples proportionally to the reward. □

B. Experimental Details

All baseline methods are implemented based on the open-source implementation as described in the following sections by following the default hyperparameters and setup. The code will be released upon publication of the paper.

B.1. Set Generation

We study the set generation task with increasing scales including small, medium, and large. The dimension of action space $|U|$ for small, medium, and large is 30, 80, and 100, respectively. The size of the target set to generate $|S|$ is 20, 60, 80, respectively. The predefined energy $\mathcal{E}(t)$ for each element t is randomly generated in the range $[-1, 1]$, and $|S|/10$ of the elements have the same energy (resulting in multiple optimal solutions). We implement DB, TB, SubTB and FL-DB based on the open-source codes[4]. The GFlowNet model is a feedforward network that consists of 2 hidden layers with 256 hidden units per layer which uses the LeakyReLU activation function. We sample a parallel of 16 rollouts from the environment for training all of the models. The GFlowNet model is trained based on the Adam (Kingma & Ba, 2015) optimizer with a learning rate of 0.001 for DB, SubTB, and FL-DB, where we use a larger learning rate of 0.1 for the learnable parameter Z for TB following (Malkin et al., 2022a).

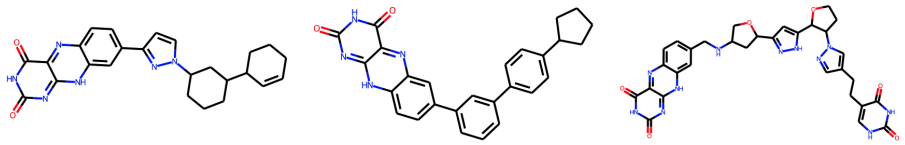
B.2. Bit Sequence Generation

We implement all baselines based on Malkin et al. (2022a) and follow the default hyperparameters and setup as in Malkin et al. (2022a). We study the bit sequence generation tasks with increasing sequence lengths n including normal ($n = 120$), long ($n = 140$), and very long ($n = 160$). The GFlowNet model is a Transformer (Vaswani et al., 2017) consisting of 3 hidden layers with 64 hidden units per layer and has 8 attention heads. The size of the minibatch is 16, and the random action probability is set to 0.0005 for performing ϵ-greedy exploration. The reward exponent is set to 3, and we use a sampling temperature of 1 for the forward policy P_F for the GFlowNet models. The learning rate for the policy parameters is 1×10^{-4} for TB, and the learning rate for the learnable parameter Z is 1×10^{-3}. The learning rate is 5×10^{-3} for the DB and FL-GFN variants.

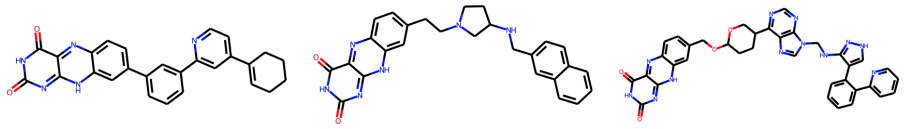
[4] https://github.com/GFNOrg/gflownet

B.3. Molecule Discovery

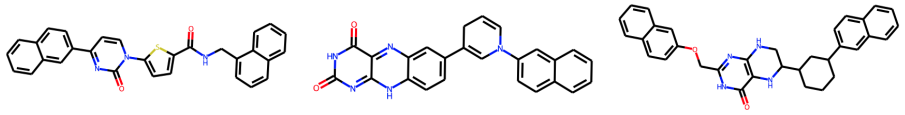
All baselines are implemented based on the open-source codes[5]. We use a reward proxy provided in Bengio et al. (2021a). We use Message Passing Neural Networks (MPNN) for the network architecture for all GFlowNet models, as the molecule is represented as an atom graph. We follow the default hyperparameters and setup as in Malkin et al. (2022a). We fine-tune the reward exponent by grid search following (Malkin et al., 2022a) and set it to be 4. We use a random action probability of 0.1 for performing ϵ-greedy exploration, and the learning rate is 5×10^{-4}.



(a) Detailed balance (DB)



(b) Trajectory balance (TB)



(c) FL-GFN

Figure 7. Visualization of top-3 molecules generated by different methods.

[5] https://github.com/GFNOrg/gflownet/tree/master/mols