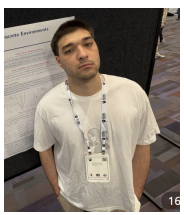
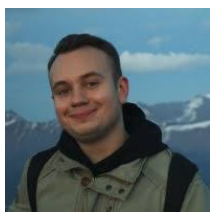
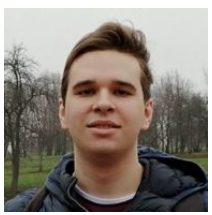
On some methodological aspects of generative flow networks

Dmitry Vetrov [4], Timofei Gritsaev [4], Ian Maksimov [1], Nikita Morozov[1], Alexey Naumov[1,2], **Sergey Samsonov** [1], Daniil Tiapkin[1,3]

[1] HSE University [2] Steklov mathematical institute RAS [3] Ecole Polytechnique [4] Constructor University



GFlowNet Problem

Suppose we have a finite discrete space of objects $\mathcal{X}$ with a non-negative reward function $\mathcal{R}(x)$ given for every $x \in \mathcal{X}$.

Goal: Construct and train a model that will sample objects from $\mathcal{X}$ from the probability distribution $\mathcal{R}(x)/\mathrm{Z}$ proportional to the reward, where Z is an unknown normalizing constant.

Standard approach: MCMC methods usually suffer from the mixing problem.

GFlowNet as an alternative: it learns a stochastic policy that starts with "empty" objects and modifies them until completion. GFlowNet generates each sample independently and, like RL methods, relies on exploration and the generalization power of ML to guess and discover new modes of the reward function (or regions of low energy).

References: [Bengio et al., 2021] (original paper), [Bengio et al., 2023] (mathematical foundations)

Flows Over Directed Acyclic Graphs

DAG: $\mathcal{G} = (\mathcal{S}, \mathcal{E})$: $\mathcal{S}$ is a state space, $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$ is a set of edges. For $s \in \mathcal{S}$, $\mathcal{A}_s$ is the set of actions, i.e. possible next states: $s' \in \mathcal{A}_s \Leftrightarrow s \to s' \in \mathcal{E}$; $s_0 \in \mathcal{S}$ is an initial state ("empty" object) and the set of states with outgoing edge to the final vertex s_f, which directly corresponds to $\mathcal{X}$

Denote $\mathcal{T}$ as a set of all complete trajectories in the graph $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau} \to s_{n_\tau+1} = s_f)$, where n_τ is the trajectory length, $s_i \to s_{i+1} \in \mathcal{E}$, s_0 is always the initial state, and s_{n_τ} is always a terminal object, thus $s_f \in \mathcal{X}$

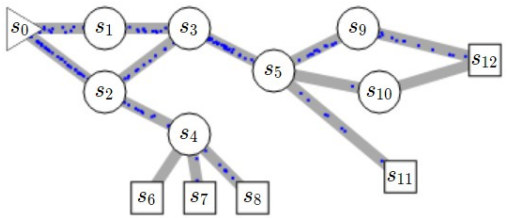


Figure: Directed Acyclic Graphs

A **trajectory flow** is any nonnegative function $\mathcal{F}: \mathcal{T} \to \mathbb{R}_{\geq 0}$.

$$\mathcal{F}(s) \triangleq \sum_{\tau \ni s} \mathcal{F}(\tau),$$

$$\mathcal{F}(s \to s') \triangleq \sum_{\tau \ni (s \to s')} \mathcal{F}(\tau).$$

Figure: Molecule synthesis experiment

Reward Matching Condition

If $\mathcal{F}$ is not identically zero, it can be used to introduce a probability distribution over complete trajectories:

$$\mathcal{P}(\tau) \triangleq \frac{1}{\mathrm{Z}_{\mathcal{F}}}\mathcal{F}(\tau), \quad \mathrm{Z}_{\mathcal{F}} \triangleq \sum_{\tau \in \mathcal{T}} \mathcal{F}(\tau).$$

Then, $\mathcal{F}(s)/\mathrm{Z}_{\mathcal{F}}$ corresponds to the probability that a random trajectory contains the state s, and $\mathcal{F}(s \to s')/\mathrm{Z}_{\mathcal{F}}$ corresponds to the probability that a random trajectory contains the edge $s \to s'$. These definitions also imply $\mathrm{Z}_{\mathcal{F}} = \mathcal{F}(s_0) = \sum_{x \in \mathcal{X}} \mathcal{F}(x)$.

Denote $\mathcal{P}(x)$ as the probability that x is the terminal state of a trajectory sampled from $\mathcal{P}(\tau)$. If *the reward matching constraint* is satisfied for all terminal states $x \in \mathcal{X}$:

$$\mathcal{F}(x) = \mathcal{R}(x), \tag{1}$$

$\mathcal{P}(x)$ will be equal to $\mathcal{R}(x)/\mathrm{Z}$. **Thus by sampling trajectories we can sample the objects from the distribution of interest**

Markovian flow

The trajectory flow $\mathcal{F}$ is said to be Markovian if $\forall s \in \mathcal{S} \setminus \mathcal{X}$ there exist distributions $\mathcal{P}_{\mathrm{F}}(-|s)$ over the children of s such that $\mathcal{P}$ can be factorized as

$$\mathcal{P}(\tau = (s_0 \to \ldots \to s_{n_\tau})) = \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}).$$

Equivalently, $\mathcal{F}$ is Markovian if $\forall s \in \mathcal{S} \setminus \{s_0\}$ there exist distributions $\mathcal{P}_{\mathrm{B}}(-|s)$ over the parents of s such that $\forall x \in \mathcal{X}$

$$\mathcal{P}(\tau = (s_0 \to \ldots \to s_{n_\tau})|s_{n_\tau} = x) = \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t),$$

If $\mathcal{F}$ is Markovian, then $\mathcal{P}_{\mathrm{F}}$ and $\mathcal{P}_{\mathrm{B}}$ can be uniquely defined as

$$\mathcal{P}_{\mathrm{F}}(s'|s) = \frac{\mathcal{F}(s \to s')}{\mathcal{F}(s)}, \quad \mathcal{P}_{\mathrm{B}}(s|s') = \frac{\mathcal{F}(s \to s')}{\mathcal{F}(s')}. \tag{2}$$

We call $\mathcal{P}_{\mathrm{F}}$ and $\mathcal{P}_{\mathrm{B}}$ the **forward policy** and the **backward policy**

Trajectory and detailed balance

The detailed balance constraint is a relation between them:

$$\mathcal{F}(s)\mathcal{P}_{\mathrm{F}}(s'|s) = \mathcal{F}(s')\mathcal{P}_{\mathrm{B}}(s|s'). \tag{3}$$

The trajectory balance constraint states that for any complete trajectory

$$\prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}) = \frac{\mathcal{F}(s_{n_\tau})}{Z_{\mathcal{F}}} \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t). \tag{4}$$

Thus any Markovian flow can be uniquely determined from $Z_{\mathcal{F}}$ and $\mathcal{P}_{\mathrm{F}}(-|s)\ \forall s \in \mathcal{S} \setminus \mathcal{X}$ or from $\mathcal{F}(x)\ \forall x \in \mathcal{X}$ and $\mathcal{P}_{\mathrm{B}}(-|s)\ \forall s \in \mathcal{S} \setminus \{s_0\}$. We refer to the left and right-hand sides of (4) as to the forward and backward trajectory distributions:

$$\mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{F}}}(\tau) := \prod_{i=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_i|s_{i-1}), \qquad \mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{B}}}(\tau) := \frac{\mathcal{R}(s_{n_\tau})}{\mathrm{Z}} \cdot \prod_{i=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{i-1}|s_i), \tag{5}$$

where $\tau = (s_0, s_1, \ldots, s_{n_\tau}) \in \mathcal{T}$.

Training GFlowNet

To train GFlowNet we parameterize a Markovian flow by some model and train it to minimize some objective. If the objective is globally minimized, we should have $\mathcal{F}(x) = \mathcal{R}(x)$.

Trajectory Balance (TB): for each complete trajectory $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau} = x)$:

$$\mathcal{L}_{\mathrm{TB}}(\tau) = \left(\log \frac{\mathrm{Z}_\theta \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1},\theta)}{\mathcal{R}(x)\prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t,\theta)}\right)^2. \tag{6}$$

Detailed Balance (DB): for each edge $s \to s' \in \mathcal{E}$:

$$\mathcal{L}_{\mathrm{DB}}(s,s') = \left(\log \frac{\mathcal{F}_\theta(s)\mathcal{P}_{\mathrm{F}}(s'|s,\theta)}{\mathcal{F}_\theta(s')\mathcal{P}_{\mathrm{B}}(s|s',\theta)}\right)^2. \tag{7}$$

For terminal states $\mathcal{F}_\theta(x)$ is substituted by $\mathcal{R}(x)$ in the objective.

One can also use equivalence between generative flow networks and Entropy-regularized RL, see [Tiapkin et al., 2024].

GFlowNet Training as Soft RL Problem

Theorem

The optimal policy $\pi^\star(s'|s)$ for a regularized MDP with a coefficient $\lambda = 1$ is equal to $\mathcal{P}_\mathrm{F}(s'|s)$ for all $s \in \mathcal{S} \setminus \{s_f\}, s' \in \mathcal{A}_s$;

Moreover, $\forall s \neq s_f, s' \neq s_f$ regularized optimal value $V^\star(s)$ and Q-value $Q^\star(s, s')$ coincide with $\log \mathcal{F}(s)$ and $\log \mathcal{F}(s \to s')$ respectively, where $\mathcal{F}$ is the Markovian flow defined by $\mathcal{P}_\mathrm{B}$ and the GFlowNet reward $\mathcal{R}$.

In addition, for any forward policy $\mathcal{P}_\mathrm{F}$ (not necessarily proper):

$$V_{\lambda=1}^{\mathcal{P}_\mathrm{F}}(s_0; r^{\mathcal{P}_\mathrm{B}}) = \log \mathrm{Z} - \mathrm{KL}(\mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{F}} \| \mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{B}}) \,.$$

- **Main outcome:** any RL algorithm that works with entropy regularization can be utilized to train GFlowNets when $\mathcal{P}_\mathrm{B}$ is fixed.
- Soft RL methods can not address the changing of the reward function, except for reward shaping schemes [Ng et al., 1999].
- The results is due to [Tiapkin et al., 2024].

Our results: Trajectory Likelihood Maximization

Theoretical insight from equivalence with Soft RL

All existing GFlowNet training methods (implicitly) minimize KL-divergence $\mathrm{KL}(\mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{F}}(\tau)} \| \mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{B}}(\tau)})$ when $\mathcal{P}_{\mathrm{B}}$ is fixed.

Main idea: Why not just minimizing the same objective, but over parameters of $\mathcal{P}_{\mathrm{B}}$?
Details: see [Gritsaev et al., 2024] (ICLR-2025)

Trajectory Likelihood Maximization

Consider the following optimization problem:

$$\min_{\mathcal{P}_\mathrm{F}\in\Pi_\mathrm{F},\mathcal{P}_\mathrm{B}\in\Pi_\mathrm{B}} \mathrm{KL}(\mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{F}} \| \mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{B}}), \tag{8}$$

where Π_F and Π_B are spaces of forward and backward policies, respectively. As stated, for fixed $\mathcal{P}_\mathrm{B}$ the problem (8) is equivalent to

$$V_{\lambda=1}^{\mathcal{P}_\mathrm{F}}(s_0; r^{\mathcal{P}_\mathrm{B}}) = \log \mathrm{Z} - \mathrm{KL}(\mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{F}} \| \mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{B}}) \to \max$$

over $\mathcal{P}_\mathrm{F} \in \Pi_\mathrm{F}$ in an entropy-regularized MDP. A meta-algorithm for solving (8):

$$\begin{aligned} \mathcal{P}_\mathrm{B}^{t+1} &\approx \arg\min_{\mathcal{P}_\mathrm{B}} \mathrm{KL}\left(\mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{F}^t} \middle\| \mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{B}}\right), \\ \mathcal{P}_\mathrm{F}^{t+1} &\approx \arg\min_{\mathcal{P}_\mathrm{F}} \mathrm{KL}\left(\mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{F}} \middle\| \mathrm{P}_\mathcal{T}^{\mathcal{P}_\mathrm{B}^{t+1}}\right). \end{aligned} \tag{9}$$

Trajectory Likelihood Maximization

First Step: Trajectory Likelihood Maximization. Using the connection between forward KL divergence minimization and MLE, we state the *TLM* objective:

$$\theta_{\mathrm{B}}^{t+1} \approx \arg\min_{\theta} \mathbb{E}_{\tau \sim \mathcal{P}_{\mathrm{F}}^{t}} \left[\mathcal{L}_{\mathtt{TLM}}(\theta; \tau) \right],$$

$$\mathcal{L}_{\mathtt{TLM}}(\theta; \tau) := -\sum_{i=1}^{n_\tau} \log \mathcal{P}_{\mathrm{B}}(s_{i-1} | s_i, \theta).$$

- $\tau = (s_0, s_1, \ldots, s_{n_\tau})$ is a trajectory generated by the forward policy $\mathcal{P}_{\mathrm{F}}^{t}$;
- Instead of exact solving, perform one stochastic gradient update:

$$\theta_{\mathrm{B}}^{t+1} = \theta_{\mathrm{B}}^{t} - \gamma \nabla_{\theta} \mathcal{L}_{\mathtt{TLM}}(\theta_{\mathrm{B}}^{t}; \tau).$$

Trajectory Likelihood Maximization

Second Step: Non-Stationary Soft RL Problem. To approximate the second step of (9), we exploit the equivalence between the GFlowNet framework and the entropy-regularized RL problem. This leads to the following expression:

$$\mathcal{P}_{\mathrm{F}}^{t+1} \approx \arg\min_{\mathcal{P}_{\mathrm{F}} \in \Pi_{\mathrm{F}}} \mathrm{KL}\left(\mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{F}}} \middle\| \mathrm{P}_{\mathcal{T}}^{\mathcal{P}_{\mathrm{B}}^{t+1}}\right) \iff \mathcal{P}_{\mathrm{F}}^{t+1} \approx \arg\max_{\mathcal{P}_{\mathrm{F}} \in \Pi_{\mathrm{F}}} V_{\lambda=1}^{\mathcal{P}_{\mathrm{F}}}\left(s_0; r^{\mathcal{P}_{\mathrm{B}}^{t+1}}\right),$$

where $r^{\mathcal{P}_{\mathrm{B}}} = \log \mathcal{P}_{\mathrm{B}}(s|s')$.

- Can be solved with any soft RL method, such as `SoftDQN` [Haarnoja et al., 2018];
- Existing GFlowNet algorithms with a fixed backward policy can be also applied;
- τ denotes a (possibly off-policy) trajectory.

In practice: perform a single stochastic gradient update

$$\theta_{\mathrm{F}}^{t+1} = \theta_{\mathrm{F}}^{t} - \eta \nabla_{\theta} \mathcal{L}_{\texttt{Alg}}(\theta_{\mathrm{F}}^{t}; \tau, \mathcal{P}_{\mathrm{B}}^{t+1}),$$

where $\mathcal{L}_{\texttt{Alg}}$ is a loss function for a GFlowNet or soft RL method (`SubTB`, `SoftDQN`, etc).

TLM algorithm

Algorithm 1 Trajectory Likelihood Maximization

Input: Forward and backward parameters θ^1_{F}, θ^1_{B}, any GFlowNet loss function $\mathcal{L}_{\texttt{Alg}}$, *(optional)* experience replay buffer $\mathcal{B}$;
for $t = 1$ **to** N_{iters} **do**
  Sample a batch of trajectories $\{\tau_k^{(t)}\}_{k=1}^K$ from the forward policy $\mathcal{P}_{\mathrm{F}}(\cdot|\cdot, \theta^t_{\mathrm{F}})$;
  (optional) Update $\mathcal{B}$ with $\{\tau_k^{(t)}\}_{k=1}^K$;
  Update $\theta^{t+1}_{\mathrm{B}} = \theta^t_{\mathrm{B}} - \gamma_t \cdot \frac{1}{K} \sum_{k=1}^K \nabla \mathcal{L}_{\texttt{TLM}}(\theta^t_{\mathrm{B}}; \tau_k^{(t)})$, see (7);
  (optional) Resample a batch of trajectories $\{\tau_k^{(t)}\}_{k=1}^K$ from $\mathcal{B}$;
  Update $\theta^{t+1}_{\mathrm{F}} = \theta^t_{\mathrm{F}} - \eta_t \cdot \frac{1}{K} \sum_{k=1}^K \nabla \mathcal{L}_{\texttt{Alg}}(\theta^t_{\mathrm{F}}; \tau_k^{(t)}, \mathcal{P}_{\mathrm{B}}(\cdot|\cdot, \theta^{t+1}_{\mathrm{B}}))$;
end for

Figure: TLM algorithm pseudocode

Hypergrid Environment

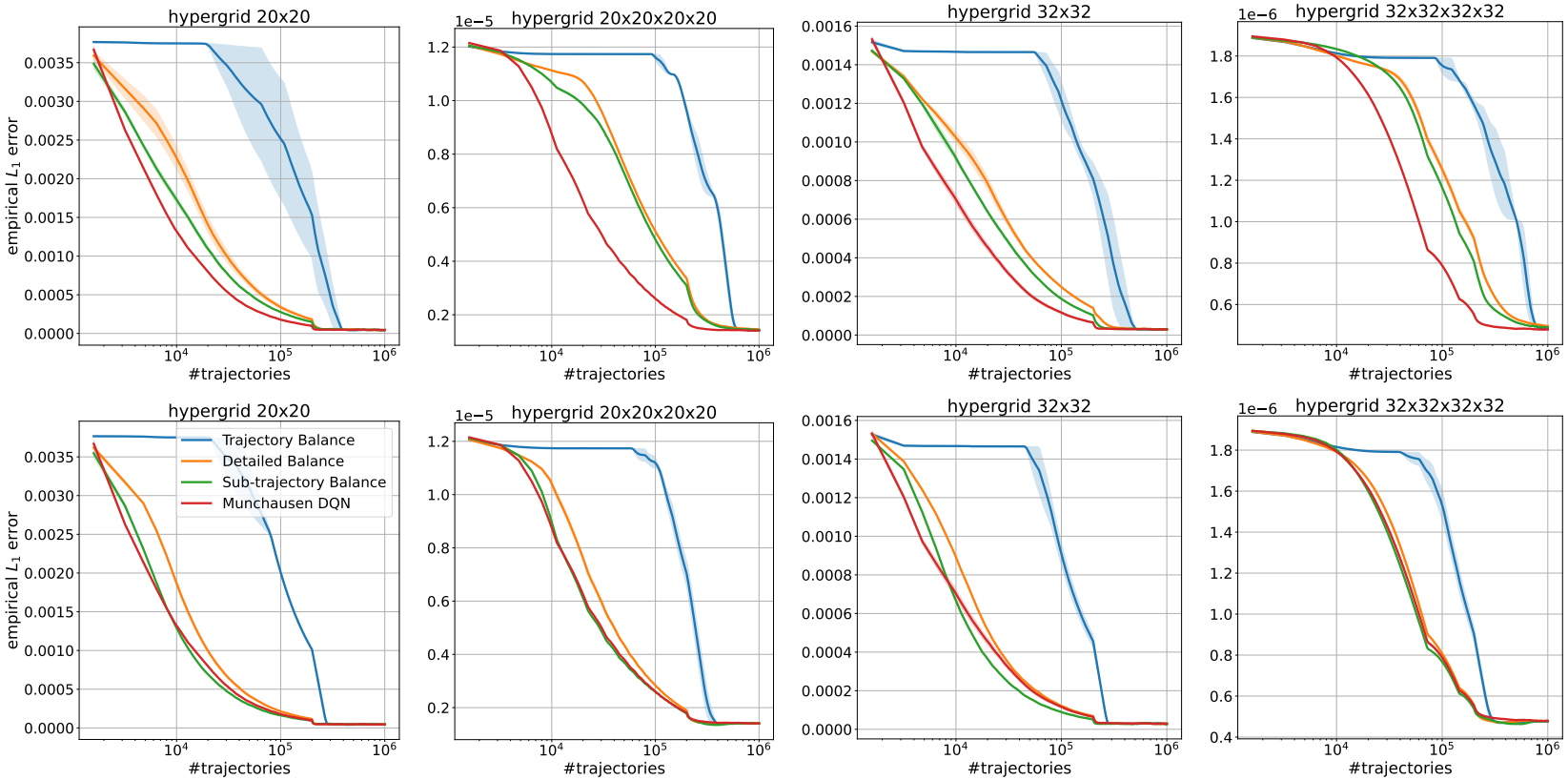


Figure: L_1 distance between target and empirical GFlowNet distributions over the course of training on hypergrid environment. *Top row:* $\mathcal{P}_{\mathrm{B}}$ is fixed to be uniform for all methods. *Bottom row:* $\mathcal{P}_{\mathrm{B}}$ is learnt for the baselines and fixed to be uniform for `M-DQN`. Mean and std values are computed over 3 runs.

Results: bit sequences generation

- Bit sequences of length $n = 120$;
- $M \subset \{0,1\}^n$ - fixed set, not observable by learner;
- Reward of generated string x computed as

$$R(x) = \exp\Big\{-\min_{y \in M} d_H(x,y)\Big\}\,.$$

Bit sequence generation

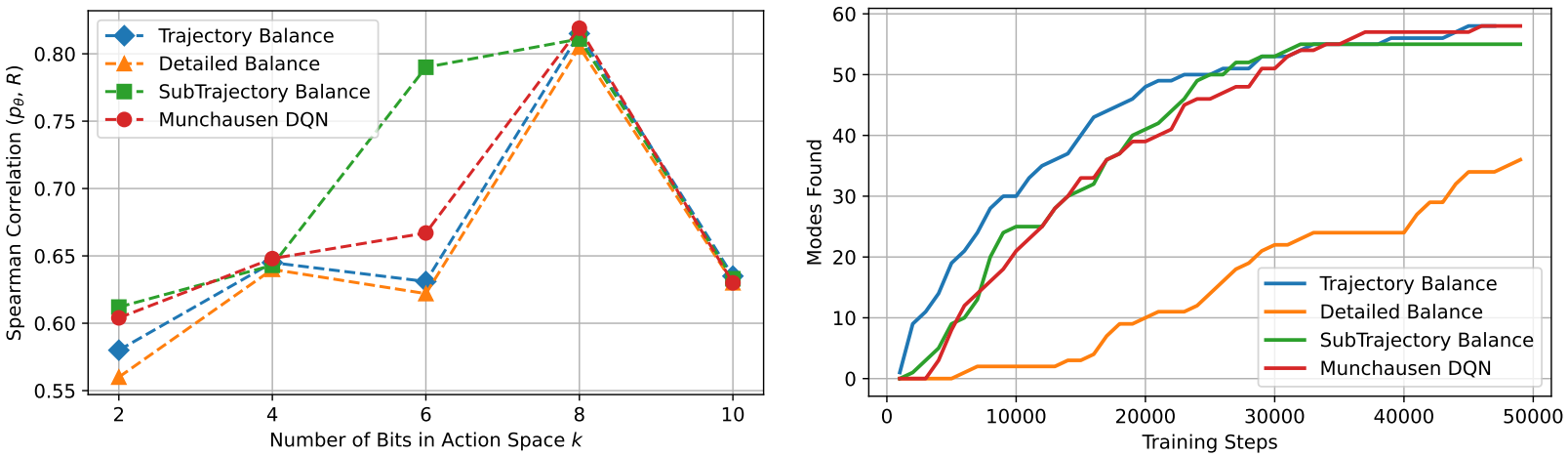


Figure: Bit sequence generation results. *Left:* Spearman correlation between $\mathcal{R}$ and $\mathcal{P}_\theta$ on a test set for each method and varying $k \in \{2, 4, 6, 8, 10\}$. *Right:* The number of modes discovered over the course of training for $k = 8$.

Small Molecule Generation

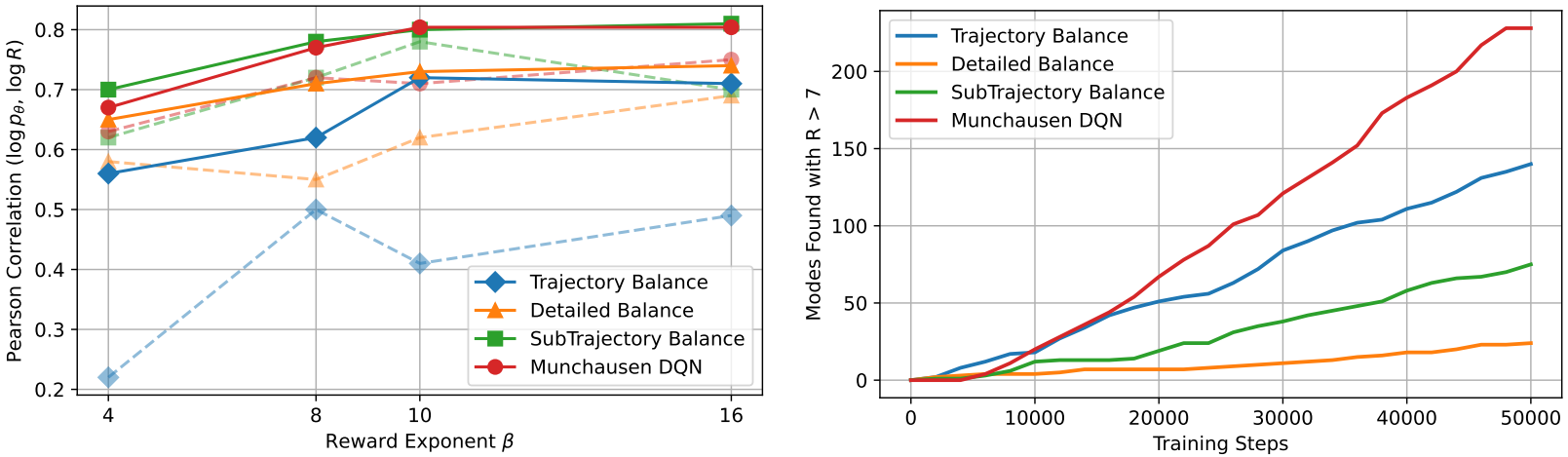


Figure: Small molecule generation results. *Left:* Pearson correlation between $\log \mathcal{R}$ and $\log \mathcal{P}_\theta$ on a test set for each method and varying $\beta \in \{4, 8, 10, 16\}$. Solid lines represent the best results over choices of learning rate, dashed lines — mean results. *Right:* Number of Tanimoto-separated modes with $\tilde{\mathcal{R}} > 7.0$ found over the course of training for $\beta = 10$.

Our results: non-acyclic GFlownets

Assumptions: we allow graph $\mathcal{G}$ to contain cycles. In addition to finiteness, we make technical assumptions on the structure of $\mathcal{G}$:

- Graph $\mathcal{G}$ is finite;
- There is a special initial state s_0 with no incoming edges and a special sink state s_f with no outgoing edges;
- For any state $s \in \mathcal{S}$ there exists a path from s_0 to s and a path from s to s_f.

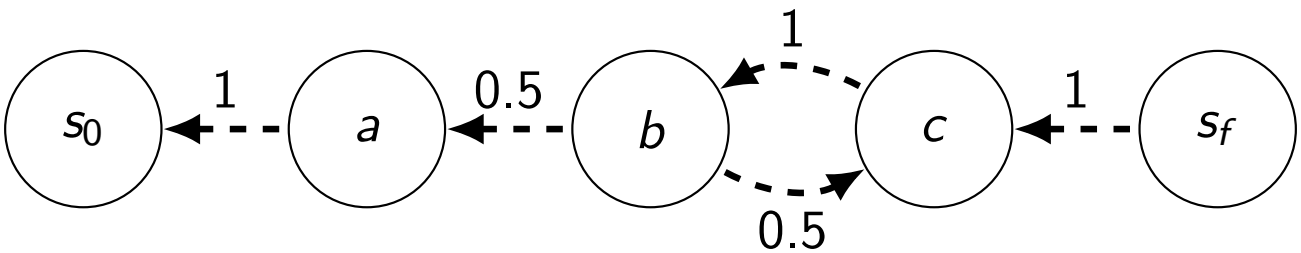


Figure: A graph with cycles, backward probabilities are labeled on the edges

Definition

A sequence $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau} \to s_{n_\tau+1} = s_f)$ is called a trajectory of length $n_\tau \in \mathbb{N}$ if all transitions $s_t \to s_{t+1} \in \mathcal{E}, t \in \{0, \ldots, n_\tau\}$. Then $\mathcal{T}$ is a set of all finite-length trajectories that start in s_0 and finish in s_f.

Trajectory length

Lemma

Let $\mathcal{P}_{\mathrm{B}}(s \mid s')$ be a backward policy, such that $\mathcal{P}_{\mathrm{B}}(s \mid s') > 0$ for any edge $s \to s' \in \mathcal{E}$. Then

- *$\mathcal{P}(\tau)$ is a probability distribution over $\mathcal{T}$, that is, $\sum_{\tau \in \mathcal{T}} \mathcal{P}(\tau) = 1$.*
- *Moreover, its expected trajectory length is always finite $\mathbb{E}_{\tau \sim \mathcal{P}}[n_\tau] = \sum_{\tau \in \mathcal{T}} n_\tau \mathcal{P}(\tau) < \infty$.*

Proof: the condition $\mathcal{P}_{\mathrm{B}}(s \mid s') > 0$ allows us to ensure that s_i is a finite state-space absorbing Markov chain.

- Next aim - define state and edge flows;
- Can we still define edge flows as *visitation probabilities* $\mathcal{F}(s \to s') = \mathcal{P}(s \to s' \in \tau)$?

Flow definition: non-acyclic

Definition

Given a final flow $\mathcal{F}(s_f) > 0$, we define state and edge flows as

$$\mathcal{F}(s \to s') \triangleq \mathcal{F}(s_f) \cdot \mathbb{E}_{\tau \sim \mathcal{P}} \left[\sum_{t=0}^{n_\tau} \mathbb{1}\{s_t = s, s_{t+1} = s'\} \right],$$

$$\mathcal{F}(s) \triangleq \mathcal{F}(s_f) \cdot \mathbb{E}_{\tau \sim \mathcal{P}(\tau)} \left[\sum_{t=0}^{n_\tau + 1} \mathbb{1}\{s_t = s\} \right]. \tag{10}$$

We say that the flows defined above are induced by the backward policy $\mathcal{P}_\mathrm{B}$ and final flow $\mathcal{F}(s_f)$.

Flows $\mathcal{F}$ defined above satisfy:

1. $\mathcal{F}(s) \overset{(a)}{=} \sum_{s' \in \mathrm{out}(s)} \mathcal{F}(s \to s') \overset{(b)}{=} \sum_{s'' \in \mathrm{in}(s)} \mathcal{F}(s'' \to s)$, for each $s \in \mathcal{S} \setminus \{s_0, s_f\}$.
2. $\mathcal{F}(s \to s') = \mathcal{F}(s')\mathcal{P}_\mathrm{B}(s \mid s')$ for any $s \to s' \in \mathcal{E}$.
3. $\mathcal{F}(s_0) = \mathcal{F}(s_f)$.

Backward policy

Observation

When a backward policy $\mathcal{P}_{\mathrm{B}} > 0$ is fixed, any loss from the acyclic GFlowNet literature can be used to learn the corresponding forward policy $\mathcal{P}_{\mathrm{F}}$.

- The expected trajectory length $\mathbb{E}_{\tau \sim \mathcal{P}}[n_\tau]$ of a manually chosen $\mathcal{P}_{\mathrm{B}}$ can be large;
- How to circumvent: learnable backward policy!

Practical DB loss

Learning a non-acyclic GFlowNet with the smallest expected trajectory length:

$$\sum_{s \in \mathcal{S} \setminus \{s_0, s_f\}} \mathcal{F}(s) \to \min_{\mathcal{F}, \mathcal{P}_\mathrm{F}, \mathcal{P}_\mathrm{B}} \tag{11}$$

$$\text{subject to } \left(\log \frac{\mathcal{F}(s) \mathcal{P}_\mathrm{F}(s' \mid s)}{\mathcal{F}(s') \mathcal{P}_\mathrm{B}(s \mid s')} \right)^2 = 0\,, \qquad \forall s \to s' \in \mathcal{E}\,,$$

$$\mathcal{F}(s_f) \mathcal{P}_\mathrm{B}(x \mid s_f) = \mathcal{R}(x)\,, \qquad \forall x \to s_f \in \mathcal{E}\,. \tag{12}$$

Approximation to (11): DB with state flow regularization:

$$\left(\log \frac{\mathcal{F}_\theta(s) \mathcal{P}_\mathrm{F}(s' \mid s, \theta)}{\mathcal{F}_\theta(s') \mathcal{P}_\mathrm{B}(s \mid s', \theta)} \right)^2 + \lambda \mathcal{F}_\theta(s) \to \min\,.$$

Here $\lambda > 0$ controls a trade-off between an expected trajectory length and sampling accuracy.

Our results: Loss Stability and Scaling Hypothesis

The scaling hypothesis

The main factor that plays a crucial role in loss stability in practice is the scale in which the error between flows is computed: log-flow scale $\Delta \log \mathcal{F}$ vs flow scale $\Delta \mathcal{F}$. We hypothesize that using log-flow scale losses without regularization can lead to arbitrarily large trajectory length, while flow scale losses are biased towards solutions with smaller flows and thus do not suffer from this issue.

Loss Stability and Scaling Hypothesis

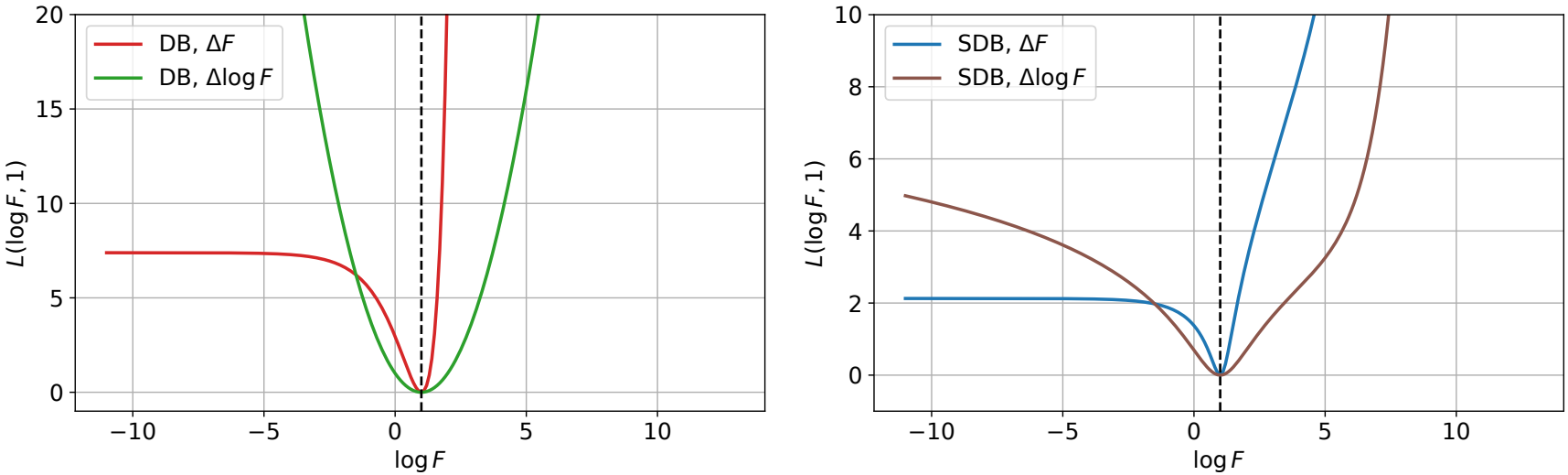


Figure: Plots for DB and SDB losses in $\Delta\mathcal{F}$ and $\Delta \log \mathcal{F}$ scales with fixed predicted log backward flow $= 1$ and varying predicted log forward flow. More specifically, green curve is $y = (x-1)^2$, red curve is $y = \left(e^x - e^1\right)^2$, brown curve is $y = \log\left(1 + (x-1)^2\right) \cdot (1 + 0.001e^x)$, blue curve is $y = \log\left(1 + \left(e^x - e^1\right)^2\right) \cdot (1 + 0.001e^x)$

Results: Hypergrid

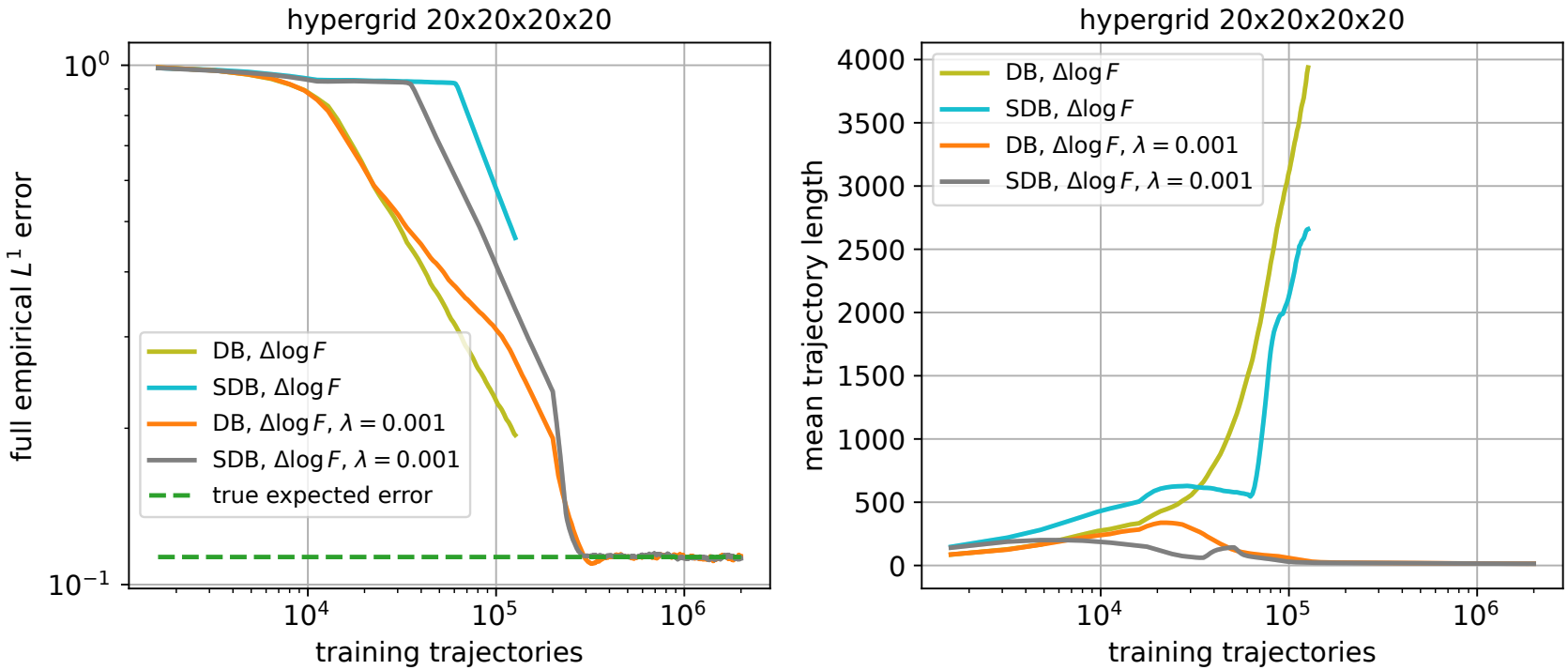


Figure: *Left:* evolution of L^1 distance between empirical distribution of samples and target distribution. *Right:* evolution of mean length of sampled trajectories. Here we note that when $\Delta \log \mathcal{F}$ scale losses are employed without state flow regularization, mean trajectory length tends to infinity. Training is done on-policy.

Further research directions

- Natural idea: extend for finding shortest paths problems, aim to solve combinatorial problems (e.g. Rubik's cube);
- Solving RL problems with hard credit assignment via non-acyclic GFlowNets?
- Amortized inference for large language models (see e.g. [Hu et al., 2023])

References

- D. Tiapkin, N, Morozov, A. Naumov, D. Vetrov. Generative Flow Networks as Entropy-Regularized RL. Spotlight presentation at AISTATS-2024, see https://arxiv.org/pdf/2310.12934.
- T. Gritsaev, N, Morozov, S. Samsonov, D. Tiapkin, Optimizing Backward Policies in GFlowNets via Trajectory Likelihood Maximization. In ICLR-2025, see https://arxiv.org/abs/2410.15474.
- N. Morozov, I. Maksimov, D. Tiapkin, S. Samsonov, Revisiting Non-Acyclic GFlowNets in Discrete Environments. In ICML-2025, https://arxiv.org/abs/2502.07735.
- T. Gritsaev, N. Morozov, K. Tamogashev, D. Tiapkin, S. Samsonov, A. Naumov, D. Vetrov, N. Malkin. Adaptive Destruction Processes for Diffusion Samplers. Submitted to ICLR-2026, https://arxiv.org/abs/2506.01541.

Thank you!

Bibliography I


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


Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023).
Gflownet foundations.
Journal of Machine Learning Research, 24(210):1–55.


Gritsaev, T., Morozov, N., Samsonov, S., and Tiapkin, D. (2024).
Optimizing backward policies in gflownets via trajectory likelihood maximization.
arXiv preprint arXiv:2410.15474.


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

In Dy, J. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 1861–1870. PMLR.


Hu, E. J., Jain, M., Elmoznino, E., Kaddar, Y., Lajoie, G., Bengio, Y., and Malkin, N. (2023).
Amortizing intractable inference in large language models.
arXiv preprint arXiv:2310.04363.


Ng, A. Y., Harada, D., and Russell, S. (1999).
Policy invariance under reward transformations: Theory and application to reward shaping.
In Icml, volume 99, pages 278–287.

Bibliography II

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