Optimizing Backward Policies in GFlowNets via Trajectory Likelihood Maximization

Timofei Gritsaev[1,2], Nikita Morozov[1], Sergey Samsonov[1], Daniil Tiapkin[3,4]

[1]HSE University [2]Constructor University, Bremen [3]CMAP, École Polytechnique [4]LMO, Université Paris-Saclay

GFlowNets

- **GFlowNets** learn to sample diverse objects from a complex discrete space $\mathcal{X}$ according to au unnormalized probability mass function $\mathcal{R}(x)$ (*GFlowNet reward*) given up to an unknown normalizing constant $\mathrm{Z} = \sum_{x\in\mathcal{X}} \mathcal{R}(x)$.
- Introduce a directed acyclic graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$. Non-terminal states describe "*incomplete*" objects, with an empty object denoted as s_0, and edges — adding new components to them. Terminal states are "*complete*" objects and coincide with $\mathcal{X}$.
- Instead of sampling only objects $x \in \mathcal{X}$, we will sample trajectories in $\mathcal{G}$ that lead to these objects, following a forward policy $\mathcal{P}_{\mathrm{F}}(s_t|s_{t-1})$. Also, we can *destroy* the object following the backward policy $\mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t)$.
- Thus, it is enough to *match the following trajectory distributions*:

$$\mathcal{P}_{\mathrm{B}}(\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau})) = \frac{\mathcal{R}(s_{n_\tau})}{\mathrm{Z}} \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t), \quad \mathcal{P}_{\mathrm{F}}(\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau})) = \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}).$$

- Parameterize policies by neural networks and introduce a training objective that will force this constraint. For example, one can optimize the *sub-trajectory balance* objective:

$$\mathcal{L}_{\mathrm{SubTB}}(\theta, \psi; \tau) = \sum_{0\le j<k\le n_\tau} w_{jk} \left(\log \frac{F_\theta(s_j) \prod_{t=j+1}^{k} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}, \theta)}{F_\theta(s_k) \prod_{t=j+1}^{k} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t, \theta)} \right)^2,$$

where w_{jk} are some non-negative weights, $F_\theta(s)$ is the so-called flow function with a convention $F_\theta(x) = \mathcal{R}(x)$ for all terminal $\forall x \in \mathcal{X}$.
- Importantly, $\mathcal{P}_{\mathrm{B}}(s|s', \theta)$ *can be either trained or fixed*, e.g. to be uniform over parents of each state.

Trajectory Likelihood Maximization

- Typical approaches for backward policy $\mathcal{P}_{\mathrm{B}}(s|s', \theta)$: (1) fixed as a uniform, (2) trained using the same forward loss.
- (1) can be sub-optimal, (2) can lead to instabilities and even slow down convergence in purely soft RL approaches. **Question:** can we do better?
- *Theoretical insight from equivalence with Soft RL*: all existing GFlowNet training methods (implicitly) minimize this KL-divergence $\mathrm{KL}(\mathcal{P}_{\mathrm{F}}(\tau)\|\mathcal{P}_{\mathrm{B}}(\tau))$ when $\mathcal{P}_{\mathrm{B}}$ is fixed.
- **Main idea**: just minimize the same objective, but over parameters of $\mathcal{P}_{\mathrm{B}}$:

$$\min_{\mathcal{P}_{\mathrm{B}}} \mathrm{KL}(\mathcal{P}_{\mathrm{F}}(\tau)\|\mathcal{P}_{\mathrm{B}}(\tau)) \iff \min_{\theta} \mathcal{L}_{\mathtt{TLM}}(\theta; \tau) := \mathbb{E}_{\tau\sim\mathcal{P}_{\mathrm{F}}}\left[-\sum_{i=1}^{n_\tau} \log \mathcal{P}_{\mathrm{B}}(s_{i-1}|s_i, \theta) \right].$$

Stabilisation Techniques

Important issue: learnable backward policy destabilizes training.

- Smaller learning rate for backward policy loss;
- Target networks in the loss computation:

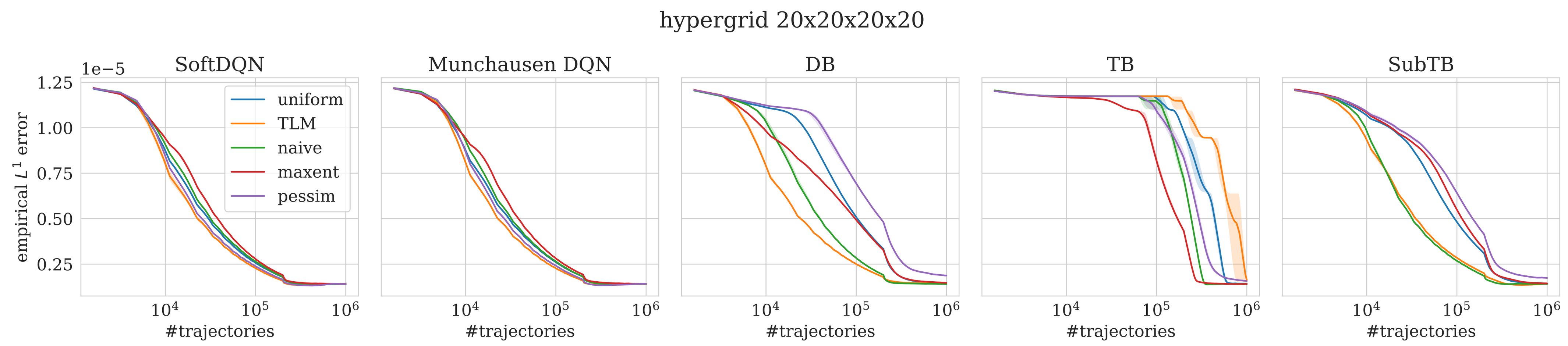
$$\mathcal{L}_{\mathrm{SubTB}}(\theta; \tau) = \sum_{0\le j<k\le n_\tau} w_{jk} \left(\log \frac{F_\theta(s_j) \prod_{t=j+1}^{k} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}, \theta)}{F_\theta(s_k) \prod_{t=j+1}^{k} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t, \bar{\theta})} \right)^2,$$

where the parameters $\bar{\theta}$ of $\mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t, \bar{\theta})$ are updated via exponential moving average (EMA) of the online parameters θ: $\bar{\theta}_{t+1} = (1-\kappa)\bar{\theta}_t + \kappa\theta_t$.
- Zero-initialization for a final layer of the backward policy to ensure the uniform initial policy;

Experimental Results

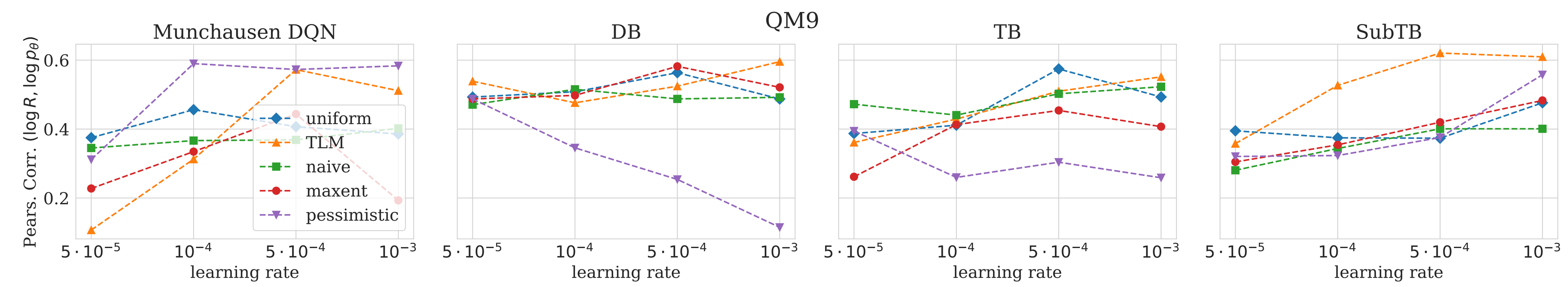
Hypergrid environment

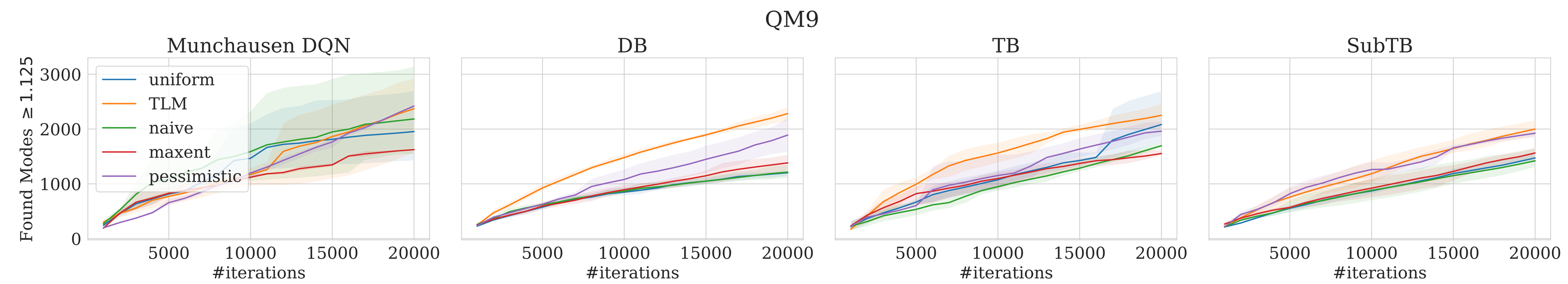
- Toy environment with tractable normalizing constant.
- Metric: L^1 distance between an empirical distribution of samples and $\mathcal{R}/\mathrm{Z}$, speed of convergence is studied.
- L^1 distance between target and empirical sample distributions over the course of training on the standard **(top row)** and hard **(bottom row)** hypergrid environments for each method. *Lower values indicate better performance.*



QM9 molecule generation

- Molecular graphs are constructed atom by atom.
- *Top*: correlation between log sampling likelihood and $\log \mathcal{R}$, *Bottom*: number of Tanimoto-separated modes.





Conclusion

- **TLM** accelerates convergence and improves quality across various benchmarks combined with different forward losses (DQN, DB, TB, and SubTB).
- We also provide results on 2 other environments (bit sequences and sEH) in the paper. Based on our experiments, we put forward a theory that a learnable backward policy is beneficial in more complex and less structured environments.
- We leave developing a rigorous theory when training a backward policy is important as a promising research direction.