Order-Preserving GFlowNets

Yihang Chen[1] Lukas Mauch[2]
[1]EPFL [2]Sony Europe B.V.

Overview

Generative Flow Networks: GFlowNets have been introduced to sample a diverse set of candidates with probabilities proportional to a given reward.

Weakness of Previous GFlowNets:

- GFlowNets require a scalar reward $R(x)$, and cannot be directly applied to the multi-objective optimization $\vec{u}(x)$.
- GFlowNets typically operate on the exponentially scaled reward $R(x) = (u(x))^\beta$ to prioritize high scalar $u(x)$ values. However, the optimal β balancing the exploration-exploitation is generally unknown.
- The exact computation of $u(x)$ might be costly, but the comparison of the ordering of $u(x)$ and $u(x')$ may be more efficient.

Contributions:

- We propose the OP-GFNs for both the single-objective maximization and multi-objective Pareto approximation, which require **only** the (partial-)ordering relations among candidates.
- We empirically evaluate our method on synthesis environment HyperGrid, and two real-world applications: NATS-Bench, and molecular designs to demonstrate its advantages in the diversity and the top reward (or the closeness to the Pareto front) of the generated candidates.
- We show that the learned order-preserving reward will balance the exploration in the early stages and the exploitation in the later stages of the training, by gradually sparsifying the reward function during the training.

Related Works:

- Bengio, Yoshua, et al. "Gflownet foundations." Journal of Machine Learning Research 24.210 (2023): 1-55.
- Jain, Moksh, et al. "Multi-objective gflownets." International conference on machine learning. PMLR, 2023.

Method

Definition: We want to maximize a set of D objectives over $\mathcal{X}$, $\vec{u}(x) \in \mathbb{R}^D$. We define the the *Pareto dominance* on vectors $\vec{u}, \vec{u'} \in \mathbb{R}^D$, such that $\vec{u} \preceq \vec{u'} \Leftrightarrow \forall k, u_k \le u'_k$. We remark that $\preceq$ induces a total order on $\mathcal{X}$ for $D = 1$, and a partial order for $D > 1$. **Notations:**

- A directed acyclic graph $G = (\mathcal{S}, \mathcal{A})$ with state space $\mathcal{S}$ and action space $\mathcal{A}$. Let $s_0 \in \mathcal{S}$ be the *initial state*, the only state with no incoming edges; and *terminal states* set $\mathcal{X}$ be the states with no outgoing edges.
- A sequence of transitions $\tau = (s_0 \to s_1 \to \cdots \to s_n)$ going from the initial state s_0 to a terminal state $s_n = x$.
- A *trajectory flow* is a nonnegative function $F : \mathcal{T} \to \mathbb{R}_{\ge 0}$.
- For any state s, define the state flow $F(s) = \sum_{s \in \tau} F(\tau)$, and, for any edge $s \to s'$, the edge flow $F(s \to s') = \sum_{\tau = (\cdots \to s \to s' \to \dots)} F(\tau)$.
- The forward transition P_F and backward transition probability are defined as $P_F(s'|s) := F(s \to s')/F(s), P_B(s|s') = F(s \to s')/F(s')$ for the consecutive state s, s'.
- To approximate a Markovian flow F on the graph G such that $F(x) = R(x) \quad \forall x \in \mathcal{X}.$.

Idea:

- We want to learn an order-preserving reward $\widehat{R}(x)$, such that $\widehat{R}(x) \le \widehat{R}(x') \leftrightarrow \vec{u}(x) \preceq \vec{u}(x')$.
- We also want $\widehat{R}(x)$ to be almost uniform in the early training stages, and to concentrate on non-dominated candidates in the later training stages.
- We use relative rather explicit boundary conditions on the terminal states to train GFNs.

Algorithm:

- Consider the terminal state set $X \subset \mathcal{X}$. The labeling distribution $\mathbb{P}_y$, indicator function of the Pareto front of X.

$$\mathbb{P}_y(x|X) := \frac{\mathbf{1}[x \in \text{Pareto}(X)]}{|\text{Pareto}(X)|}.$$

- The reward $\widehat{R}(\cdot)$ also induces a conditional distribution on the set X,

$$\mathbb{P}(x|X, \widehat{R}) := \frac{\widehat{R}(x)}{\sum_{x' \in X} \widehat{R}(x')}, \forall x \in X.$$

- Minimizing the KL divergence

$$\mathcal{L}_{\text{OP}}(X; \widehat{R}) := \text{KL}(\mathbb{P}_y(\cdot|X) \| \mathbb{P}(\cdot|X, \widehat{R})).$$

- For the trajectory balance objective, let the trajectory $\tau \to x$, we define the parameterization

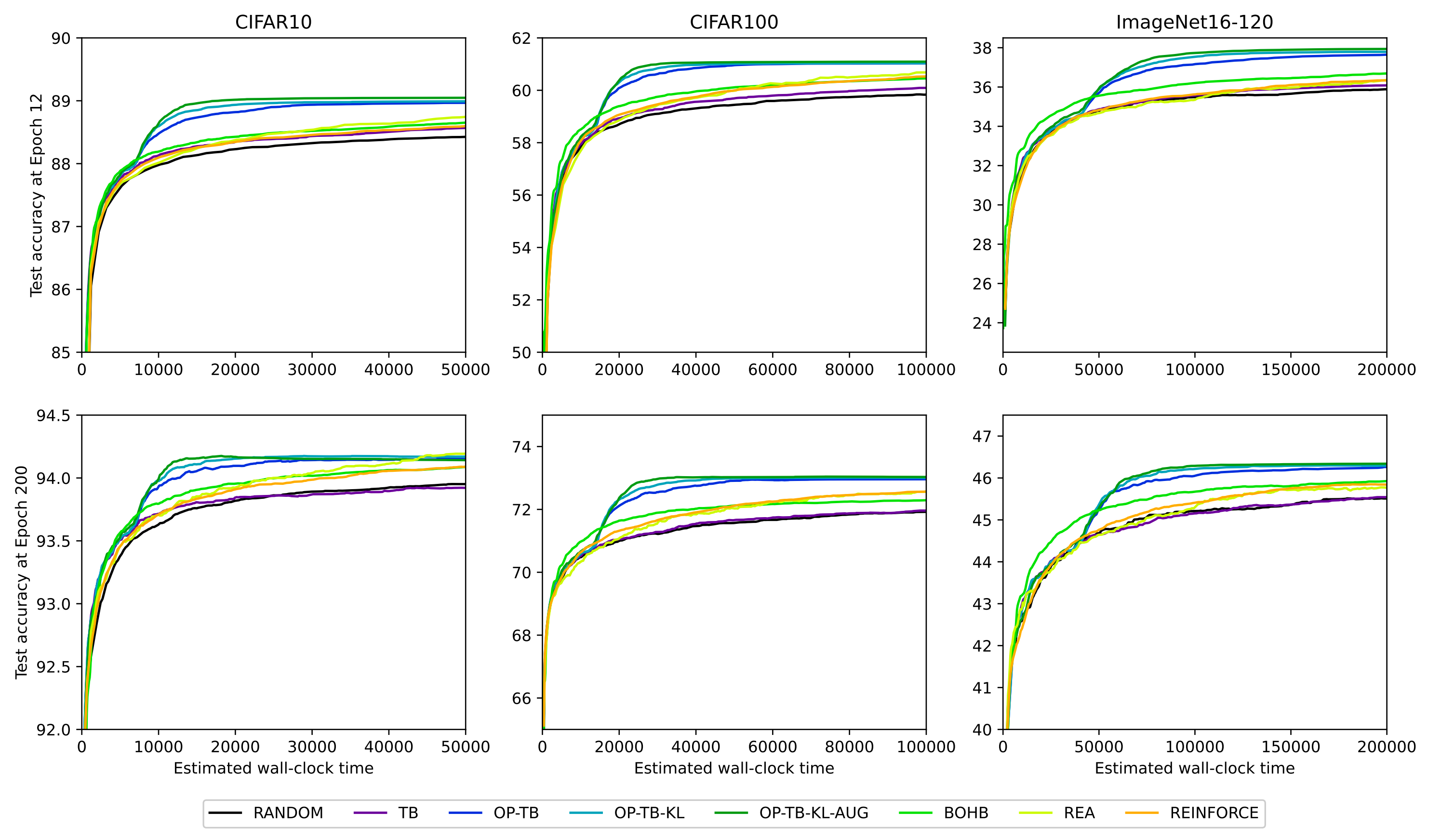
$$\widehat{R}_{\text{TB}}(x; \theta) := Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1}; \theta)/P_B(s_{t-1}|s_t; \theta).$$

Theory: For $\{x_i\}_{i=0}^n \in \mathcal{X}$, assume that $u(x_i) \le u(x_j), 0 \le i < j \le n$. When γ is sufficiently large, there exists α_γ, β_γ, dependent on γ, such that $\widehat{R}(x_{i+1}) = \alpha_\gamma \widehat{R}(x_i)$ if $u(x_{i+1}) > u(x_i)$, and $\widehat{R}(x_{i+1}) = \beta_\gamma \widehat{R}(x_i)$ if $u(x_{i+1}) = u(x_i)$, for $0 \le i \le n-1$. Also, minimize the $\mathcal{L}_{\text{OP-N}}$ qith a variable γ will drive $\gamma \to \infty, \alpha_\gamma \to \infty, \beta_\gamma \to 1$.

Experiments

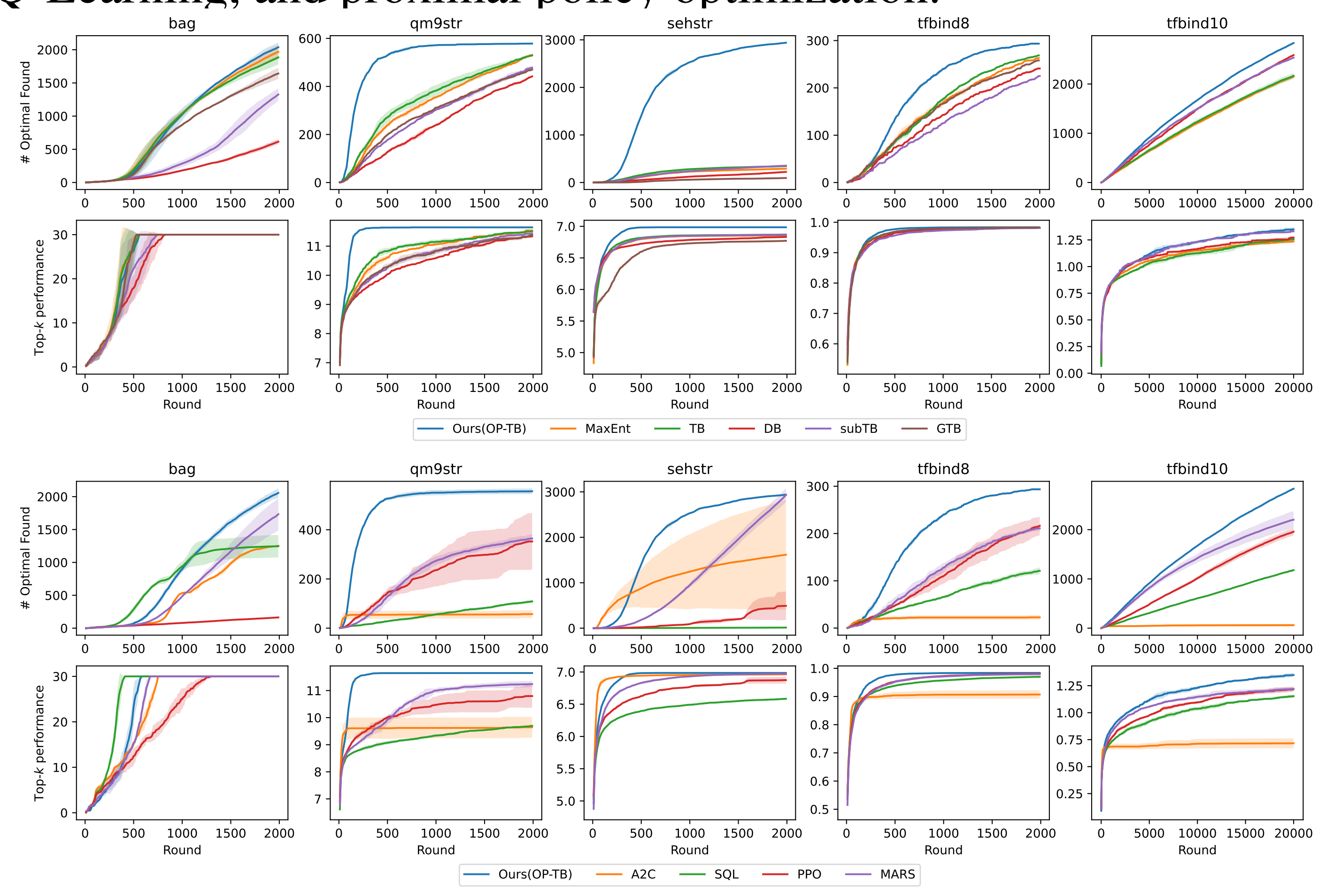
Nerual Architecture Search:

The NAS can be regarded as a sequence generation problem to generate x, where the reward of each sequence of operations is determined by the accuracy of the corresponding architecture. We follow NATS-Bench, to use the 12-th epoch accuracy in the training and the 200-th epoch accuracy in the evaluation.



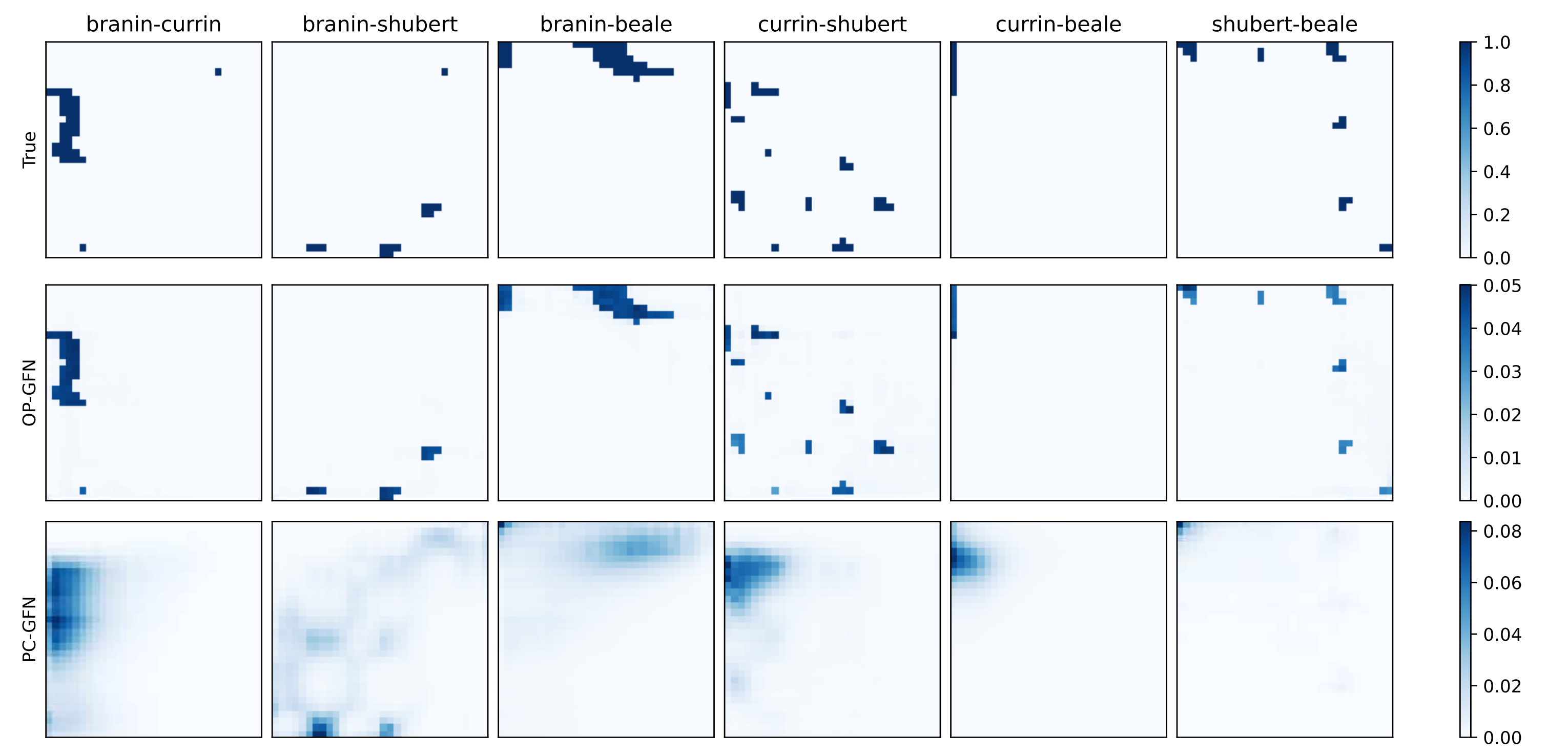
Molecular Generation

We study molecular design environments, including **Bag, TFBind8, TFBind10, QM9, sEH**. We consider GFN baselines including TB, DB, subTB, MaxEnt, and GTB. For reward-maximization methods, we consider Markov Molecular Sampling (MARS), and RL-based methods, including actor-critic, Soft Q-Learning, and proximal policy optimization.



Multi-Objective HyperGrid

We study two-dimensional HyperGrid and consider five objectives. We compare the learned reward function of OP-GFNs and PC(Preference Conditioning)-GFNs. The true Pareto front can be explicitly computed and plotted in the first row.



More experiments can be found in the paper.