Generalization and Distributed Learning of GFlowNets

Tiago da Silva, Amauri Souza, Omar Rivasplata, Vikas Garg, Samuel Kaski, Diego Mesquita

Keywords — GFlowNets, Distributed learning, PAC-Bayes

TL;DR

- we introduce the first non-vacuous generalization bounds for GFlowNets,
- we develop the first Azuma-type PAC-Bayesian bounds for understanding the generalization of GFlowNets under the light of Martingale theory,
- we demonstrate the harmful effect of the trajectory length on the proven learnability of a generalizable policy for GFlowNets,
- we introduce the first distributed algorithm for learning GFlowNets, Subgraph Asynchronous Learning, and show that it drastically accelerates learning convergence and mode discovery when compared against a centralized approach for relevant benchmark tasks

I. Background: GFlowNets

GFlowNets are amortized algorithms for sampling from distributions over discrete and compositional objects (such as graphs).

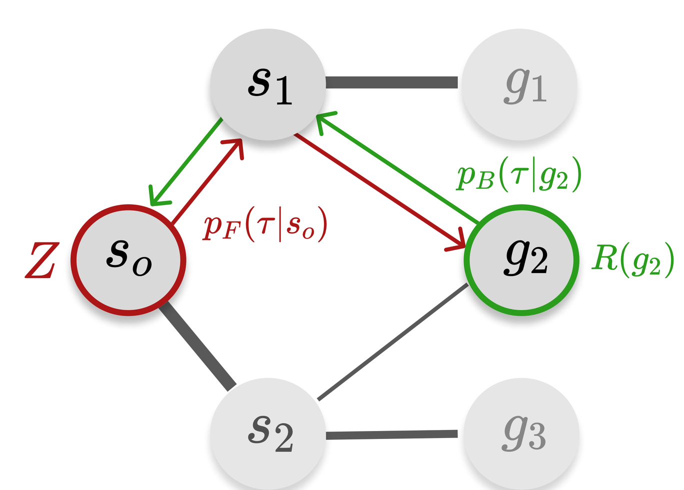

Figure 1: A GFlowNet learns a forward policy on a state graph.

A **flow network** is defined over an extension $\mathcal{S}$ of $\mathcal{G}$, which then represents the sink nodes. To navigate through this network and sample from $\mathcal{G}$ in proportion to a **reward function** $R : \mathcal{G} \to \mathbb{R}_+$, a forward (resp. backward) policy $p_F(\tau)$ ($p_B(\tau|x)$) is used.

$$p_F(\tau) = \prod_{(s,s') \in \tau} p_{F(s' \mid s)} \text{ and } \sum_{\tau \rightsquigarrow g} p_F(\tau) = R(g). \quad (1)$$

To achieve this, we parameterize $p_F(\tau)$ as a neural network trained by minimizing

$$\mathcal{L}_{TB}(p_F) = \mathbb{E}\left[\left(\log \frac{p_F(\tau)Z}{p_B(\tau \mid x)R(x)}\right)^2\right]. \quad (2)$$

for a given $p_B(\tau|x)$. GFlowNets can be trained in an **off-policy** fashion and the above expectation can be under any full-support distribution over trajectories.

II. Background: Probably Approximately Correct Bayesian Bounds

Let $\mathcal{L}$ be a loss function on a parameter space Θ, e.g., the squared loss. Also, let $\hat{\mathcal{L}}(\theta, \boldsymbol{X})$ be its empirical counterpart evaluated on a dataset $\boldsymbol{X}$.

PAC-Bayesian bounds. Given "prior" Q (independent of $\boldsymbol{X}$) and posterior P distributions over Θ, a PAC-Bayesian bound establishes an upper limit for the expectation of (unobserved) $\mathcal{L}$ based on the (observed) $\hat{\mathcal{L}}$ and a complexity term φ and a confidence level δ,

$$\mathbb{E}_{\theta \sim P}[\mathcal{L}(\theta)] \leq \mathbb{E}_{\theta \sim P}\left[\hat{\mathcal{L}}(\theta, \boldsymbol{X})\right] + \varphi(\delta, P, Q, |\boldsymbol{X}|). \quad (3)$$

When $\mathcal{L}(\theta) \leq B$ a.e., we refer to a bound as *vacuous* if

$$\mathbb{E}_{\theta \sim P}\left[\hat{\mathcal{L}}(\theta, \boldsymbol{X})\right] + \varphi(\delta, P, Q, |\boldsymbol{X}|) \geq B. \quad (4)$$

Otherwise, the bound is *non-vacuous*. Historically, the search for non-vacuous PAC-Bayesian bounds has been associated to the search for provably generalizable learning algorithms. In this regard, recent works have built upon the basic PAC-Bayesian inequalities to obtain theoretical guarantees for GANs, transformers, armed bandits, and variational autoencoders.

Data-dependent priors for PAC-Bayesian bounds. Often, φ involves the KL divergence between P and Q, which dominates the upper bound and commonly results in vacuously true statements. To circumvent this issue, we separate $\boldsymbol{X} = \boldsymbol{X}_\alpha \cup \boldsymbol{X}_{1-\alpha}$ into disjoint and independent subsets. A posterior P is learned on $\boldsymbol{X}_{1-\alpha}$ through conventional methods and a prior Q is subsequently learned on $\boldsymbol{X}_\alpha$ by minimizing the PAC-Bayesian upper bound,

$$Q^\star = \text{argmin}_Q \mathbb{E}_{\theta \sim P}\left[\hat{\mathcal{L}}(\theta, \boldsymbol{X}_\alpha)\right] + \varphi(\delta, P, Q, \alpha\, |\boldsymbol{X}|). \quad (5)$$

III. Non-vacuous Generalization Bounds for GFlowNets

There are four ingredients for a PAC-Bayesian bound: a bounded risk functional $\mathcal{L}$, a prior distribution Q, a posterior distribution P, and a learning algorithm. In alignment with the broader literature, we use a diagonal Gaussian distribution for both P and Q with fixed (small) variance. For learning, we use SGD.

A bounded risk functional for GFlowNets. In a recent work, we demonstrated that Flow Consistency in Subgraphs (FCS) is a sound and tractable learning objective for GFlowNets.

$$\text{FCS}(\pi, p_\top) = \mathbb{E}_{\mathcal{B}}\left[\text{TV}(\pi^{\mathcal{B}}, p_\top^{\mathcal{B}})\right], \quad (6)$$

in which $\pi \propto R$ is the target distribution and

$$p_{\top(x)} = \sum_{\tau \text{ finishing at } x} p_{F(\tau)} \quad (7)$$

is the probability of $x \in \mathcal{G}$ under p_F; the expectation is under a distribution of random independent subsets of $\mathcal{G}$. Intuitively, FCS measures the total variation TV between π and $p_\top$ on random subgraphs of the underlying flow network.

The stochastic and bounded nature of FCS make it a suitable candidate for pursuing a PAC-Bayesian analysis of GFlowNets. We will refer to FCS by L and to its empirical counterpart by $\hat{L}$ to emphasize its use as a risk functional for assessing the generalization of GFlowNets.

Non-vacuous generalization bounds. Let $\mathcal{T}_n$ be a n-sized set of trajectories sampled from a stationary distribution. Also, let P and Q be distributions over Θ. Then,

$$L_{\text{FCS}}(P) \leq \hat{L}_{\text{FCS}}(P, \mathcal{T}_{1-\alpha}) + \min\left(\begin{cases} \eta + \sqrt{\eta\left(\eta + 2\hat{L}_{\text{FCS}}(P, \mathcal{T}_{1-\alpha})\right)} \\ \sqrt{\frac{\eta}{2}} \end{cases}\right) \quad (8)$$

in which the complexity term η is, for chosen $\alpha \in (0, 1)$,

$$\eta := \frac{\text{KL}(P \parallel Q) + \log \frac{2\sqrt{\lfloor(1-\alpha)n\rfloor}}{\delta}}{\lfloor(1-\alpha)n\rfloor}. \quad (9)$$

We optimize Equation 8 to obtain data-dependent priors Q over Θ. Figure 2 shows the resulting bounds are non-vacuous. These are the first positive and rigorous results regarding the genearlization GFlowNets in the literature.

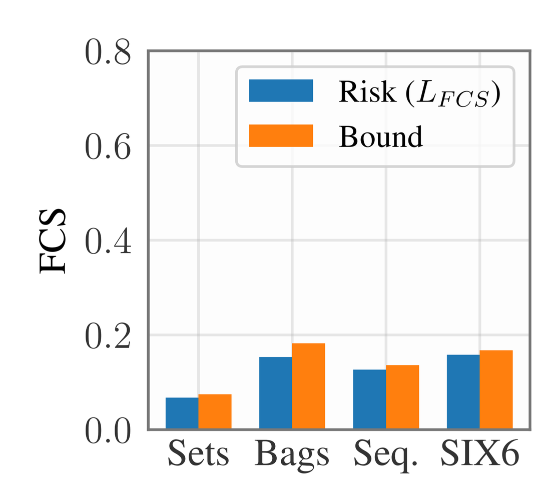

Figure 2: Non-vacuous generalization bounds for GFlowNets.

Oracle generalization bounds for GFlowNets. Let $\mathcal{L}$ be the within-trajectory detailed balance loss function and assume that $\mathcal{L} \leq U$ a.e.. Additionally, define t_m as the maximum trajectory length within the flow network and T as a budget for the number of transitions.

$$\mathbb{E}_{\theta \sim P}[\mathcal{L}(\theta)] \leq \frac{1}{\beta}\mathbb{E}_{\theta \sim P}\left[\hat{\mathcal{L}}(\theta)\right] + \alpha_{T,n}\left(\text{KL}(P \parallel Q) + \log \frac{2}{\delta}\right) + \frac{\log t_m}{\beta T \lambda} + \gamma \frac{\lambda 2 U^2}{\beta T} \quad (10)$$

in which $\beta \in (0, 1)$, $\lambda > 0$, and

$$\alpha_{T,n} = \frac{U}{2\beta(1-\beta)n} + \frac{1}{\beta T \lambda}. \quad (11)$$

IV. Subgraph Asynchronous Learning (SAL)

Equation 10 demonstrates that the larger trajectory length t_m plays a key role in constraining the generalization potential of GFlowNets. To mitigate this effect, we propose a distributed divide-and-conquer learning algorithm that breaks up the state graph into smaller subgraphs and learns a GFlowNet for each subgraph. The resulting GFlowNets are aggregated in a final, efficient step. We refer to this approach as **Subgraph Asynchronous Learning** (SAL).

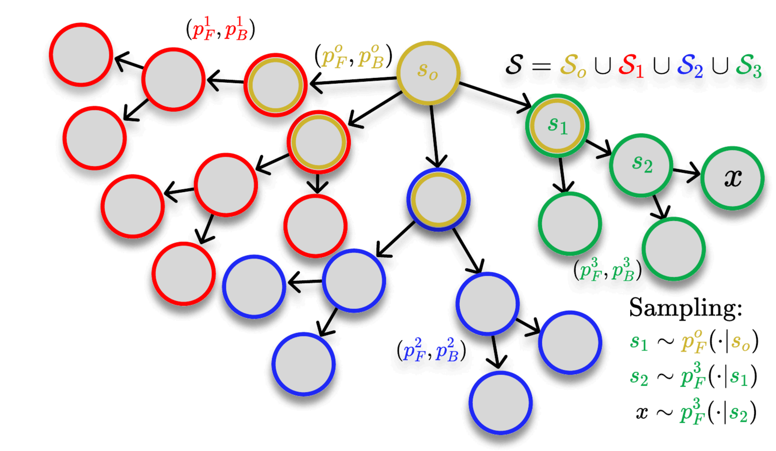

Figure 3: An illustration of SAL.

Algorithm 1 Subgraph Asynchronous Learning

1: $\mathcal{S} = \mathcal{S}_o \cup \bigcup_{j=1}^m \mathcal{S}_j$ ▷ Fixed-horizon partition
2: $\mathcal{I}_j = \mathcal{S}_j \cap \mathcal{S}_o$ for $j \in \{1, \ldots, m\}$
3: ▷ Local training
4: **parfor** $j \in \{1, \ldots, m\}$ **do**
5: ▷ Minimize $\mathcal{L}^j_{\text{ATB}}$ in $\mathcal{S}_j$ with SGD
6: $(p^j_F, F_j) = \arg\min_{p_F, F} \mathcal{L}^j_{\text{ATB}}(p_F, F)$
7: **end parfor**
8: $R^o : x \mapsto \mathbf{1}_{\{x \in \mathcal{X}\}} R(x) + \sum_{j=1}^m \mathbf{1}_{\{x \in \mathcal{I}_j\}} F_j(x)$
9: $(p^o_F, F_o) = \arg\min_{p_F, F} \mathcal{L}_{\text{TB}}(p_F, F, R^o)$
10: **return** $\{(p^o_F, F_o)\} \cup \bigcup_{1 \leq j \leq m} \{(p^j_F, F_j)\}$

SAL. Let $\{(p^1_F, F_1), \ldots, (p^m_F, F_m)\}$ be m GFlowNets defined over each of the m components of the partition defining SAL. Also, let q_j be a distribution over the initial states and p^j_E be an distribution over trajecotires within the jth component S_j. Then, each (p^j_F, F_j) is learned by minimizing the *amortized trajectory balance loss* over S_j

$$\mathcal{L}^j_{\text{ATB}}\left(p^j_F, F_j\right) = \mathbb{E}_{s \sim q_j} \mathbb{E}_{\tau \sim p_E^{j(\cdot \mid s)}}\left[\left(\log \frac{F_j(s) p^j_F(\tau \mid s)}{R(x) p^j_B(\tau \mid x)}\right)^2\right], \quad (12)$$

which replaces Z by the flow function F^j in the conventional $\mathcal{L}_{TB}$.

Our experimental results in Figure 5 and Figure 6 show that SAL often accelerates training convergence and increases the mode-finding capability of the GFlowNet. These observations are in alignment with our theoretical analysis in Equation 10 (Figure 5) and corroborate the intuition that a divide-and-conquer approach enhances the exploration of the learning agent (Figure 6).

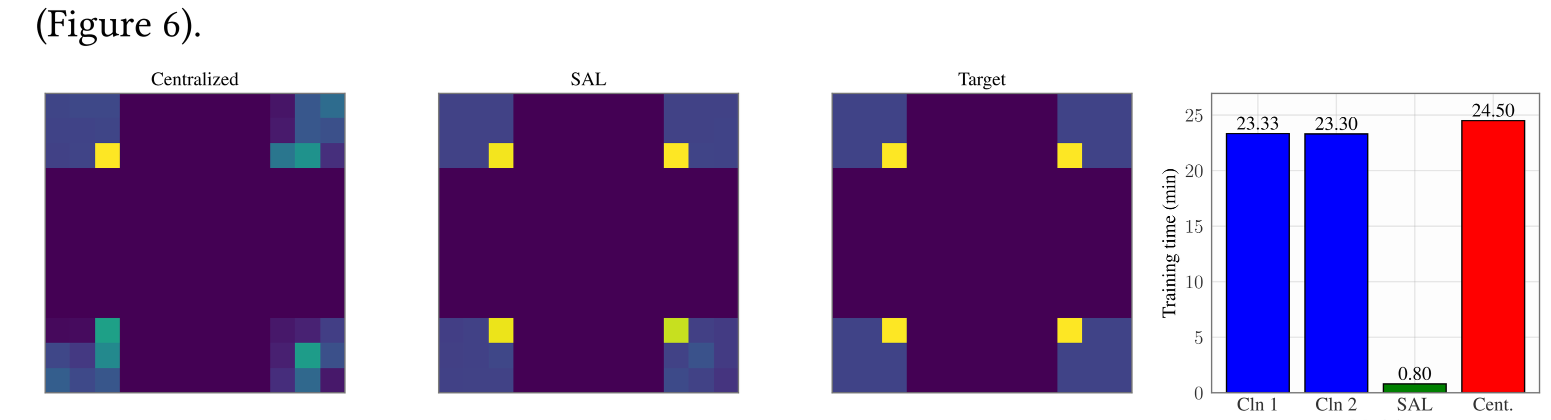

Figure 5: SAL improves learning convergence for the hypergrid environment.

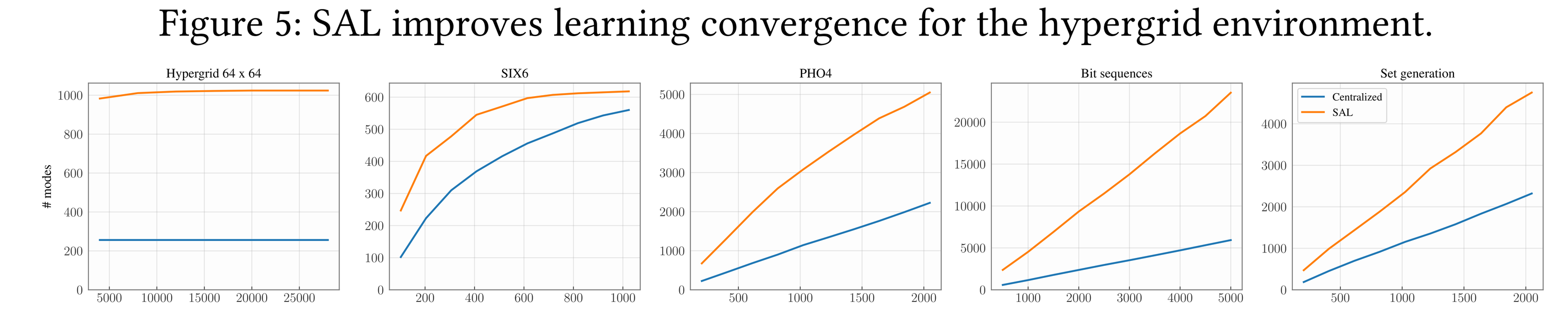

Figure 6: SAL drastically accelerates mode-finding for benchmark tasks.

Recursive SAL. SAL can be hierarchically extended to accommodate nested partitions of the state graph. This is illustrated in Figure [ref], and the resulting method is referred to as Recursive SAL. Notably, the depth of the nested partition characterizes a trade-off between the number of trainable models and the difficult of the problem that each model solves. The balance between these factors should be addressed in a case-by-case basis in future endeavors.

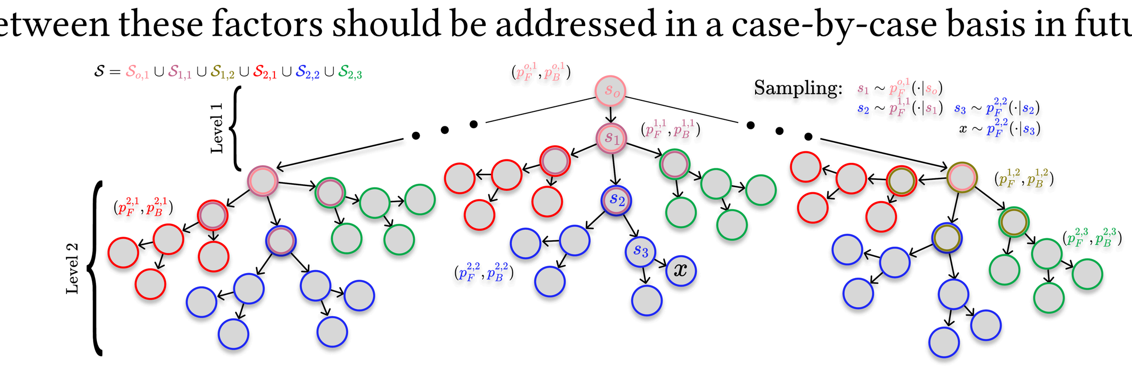

Figure 7: Illustration of Recursive SAL as an hierarchical extension of SAL.

{tiago.henrique, diego.mesquita}@fgv.br, {omar.rivasplata}@manchester.ac.uk, {amauri.souza, sami.kaski}@aalto.fi