Bayesian Structure Learning with Generative Flow Networks (Supplementary material)

Tristan Deleu[1] **António Góis**[1] **Chris Emezue**[2] **Mansi Rankawat**[1]
Simon Lacoste-Julien[1,4] **Stefan Bauer**[3,5] **Yoshua Bengio**[1,4,6]

[1]Mila, Université de Montréal [2]Technical University of Munich [3]KTH Stockholm
[4]CIFAR AI Chair [5]CIFAR Azrieli Global Scholar [6]CIFAR Senior Fellow

A LIMITATIONS OF DAG-GFLOWNET

Although we have shown in the main paper that DAG-GFlowNet is capable of learning an accurate approximation of the posterior distribution $P(G \mid \mathcal{D})$ when the size of the dataset $\mathcal{D}$ is moderate (a situation where the benefits of a Bayesian treatment of structure learning are larger), we observed that as the size of the dataset increases, fitting the detailed-balance loss in (10) was more challenging. This can be explained by the fact that with a larger amount of data, the posterior distribution becomes very peaky (Koller and Friedman, 2009). As a consequence, in this situation, the delta-score in (9), which is required to calculate the loss, can take a wide range of values: adding an edge to a graph can drastically increase or decrease its score. In turn, the neural network parametrizing $P_\theta(G_{t+1} \mid G_t)$ needs to compensate for these large fluctuations, making it harder to train.

Unfortunately, some of the standard techniques used in Machine Learning to tackle this issue, such as normalization of the inputs, cannot be applied here. Normalizing the delta-score is equivalent to normalizing the rewards $R(G)$ and $R(G')$ themselves, and as a consequence it would change the distribution that is being approximated: instead of approximating the posterior distribution $P(G \mid \mathcal{D})$, we would approximate a distribution $P(G \mid \mathcal{D})^\tau$ under some temperature τ. Solutions to this problem include a schedule of temperature, similar to simulated annealing, or a reparametrization of $P_\theta(G_{t+1} \mid G_t)$ to better handle large fluctuations of delta-scores; this exploration is left as future work.

B DETAILED-BALANCE CONDITION WITH ALL COMPLETE STATES

In this section, we will prove a special case of the *detailed-balance condition* introduced by Bengio et al. (2021) applied to the case where all the states of the GFlowNet are complete (except the terminal state s_f). To simplify the presentation, we will follow the notations of Bengio et al. (2021), and denote the forward transition probability by $P_F(s_{t+1} \mid s_t)$—instead of $P_\theta(s_{t+1} \mid s_t)$ in the main paper. Recall that the detailed-balance condition (Bengio et al., 2021, Def. 17) is given by

$$F(s_t)P_F(s_{t+1} \mid s_t) = F(s_{t+1})P_B(s_t \mid s_{t+1}). \tag{B.1}$$

In the case where all the states are complete, we also know that (Bengio et al., 2021, Def. 16)

$$P_F(s_f \mid s_t) := \frac{F(s_t \to s_f)}{\sum_{s' \in \mathrm{Ch}(s_t)} F(s_t \to s')} = \frac{R(s_t)}{F(s_t)} \quad \Leftrightarrow \quad F(s_t) = \frac{R(s_t)}{P_F(s_f \mid s_t)},$$

where $F(s \to s')$ represents the flow from state s to s', as described in Section 3.1, $F(s)$ is the total flow through state s, and we used Proposition 4 & Equation 34 of Bengio et al. (2021) to introduce $F(s_t)$ and $R(s_t)$ respectively. Replacing $F(\cdot)$ in (B.1) yields the expected condition:

$$R(s_t)P_F(s_{t+1} \mid s_t)P_F(s_f \mid s_{t+1}) = R(s_{t+1})P_B(s_t \mid s_{t+1})P_F(s_f \mid s_t). \tag{B.2}$$

The original formulation in (B.1) would require us to parametrize both $P_F(s_{t+1} \mid s_t)$ and $F(s)$. On the other hand, using this alternative condition, we only have to parametrize $P_F(s_{t+1} \mid s_t)$ (including when $s_{t+1} = s_f$ is the terminal state).

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

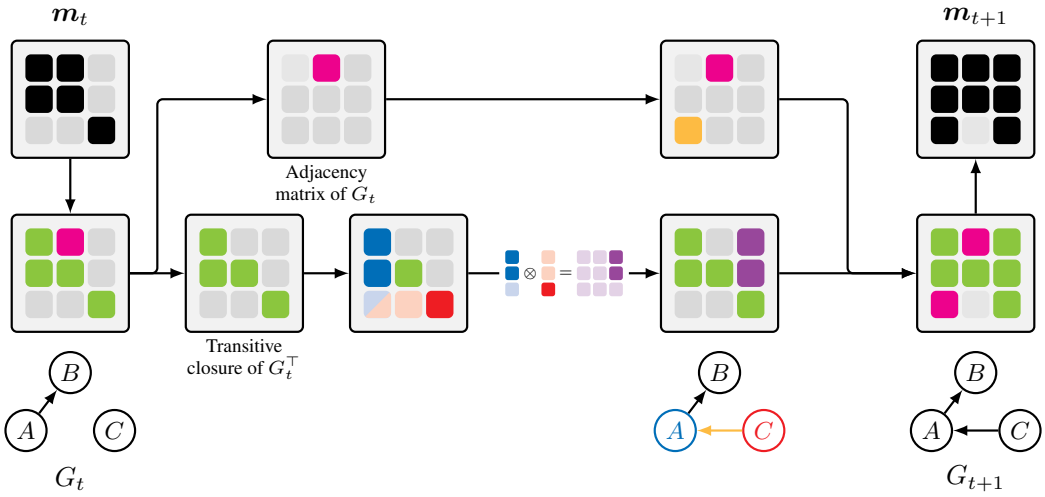

Figure 6: Online update of the mask $\boldsymbol{m}$. The mask $\boldsymbol{m}_t$ associated with G_t represents (in black) the edges that cannot be added to G_t to obtain a valid DAG. $\boldsymbol{m}_t$ is decomposed in two parts: the adjacency matrix of G_t (top), and the transitive closure of $G_t^\top$ (bottom). To update the mask and obtain $\boldsymbol{m}_{t+1}$ associated with G_{t+1}, the result of adding the edge $C \to A$ to G_t, each component must be updated separately, and then recombined. The diagonal elements of $\boldsymbol{m}_t$, corresponding to self-loops (which are always invalid actions to take) are integrated into the transitive closure of $G_t^\top$ by convention.

C DEFINITION AND UPDATE OF THE MASK OVER ACTIONS

In Section 4.1, we introduced a mask $\boldsymbol{m}$ associated with a DAG G to indicate which edges could be legally added to G to obtain a new valid DAG G'. This mask must ignore (1) the edges already present in G (which cannot be added further), and (2) any edge whose addition leads to the introduction of a cycle. The mask $\boldsymbol{m}$ is constructed using (1) the adjacency matrix of G, and (2) the adjacency matrix of the transitive closure of $G^\top$, the transpose graph of G; recall that $G^\top$ is obtained from G by inverting the direction of its edges.

Giudici and Castelo (2003) use a similar construction to efficiently obtain the legal actions their MCMC sampler may take. In particular, they show that this mask $\boldsymbol{m}$ can be updated very efficiently online as edges are added one by one. In practice, this allows us to circumvent an expensive check for cycles at every stage of the construction of a sample DAG in the GFlowNet. Since the mask can be composed in 2 parts (as explained above), we can simply update each part anytime a new edge is added to a DAG G.

In Figure 6, we show how the mask $\boldsymbol{m}_t$ associated with a graph G_t can be updated after adding a new edge $C \to A$ to obtain the mask $\boldsymbol{m}_{t+1}$. The mask is decomposed in 2 parts: the adjacency matrix of G_t, and the transitive closure of $G_t^\top$. After adding $C \to A$, each component is updated separately:

1. **Adjacency matrix:** To update the adjacency matrix, the entry in the adjacency matrix must be set (here, the entry corresponding to the edge $C \to A$).
2. **Transitive closure:** To update the transitive closure of the transpose, we need to compute the outer product of the column corresponding to the target of the edge (here A, in blue) with the row corresponding to the source of the edge (here C, in red). The outer product is added (more precisely, this is a binary OR) to the initial transitive closure.

These two operations can be done very efficiently in $O(d^2)$, where d is the number of nodes in the DAG.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 DETAILS ABOUT THE METRICS

Throughout this paper, we used mainly two metrics to compare the performance of DAG-GFlowNet over alternative Bayesian structure learning algorithms: the *expected SHD* ($\mathbb{E}-$SHD), and the *area under the ROC curve* (AUROC). Let $\{G_1, \ldots, G_n\}$ be samples from the posterior approximation to be evaluated, and $G^\star$ be the ground truth graph. The $\mathbb{E}$-SHD

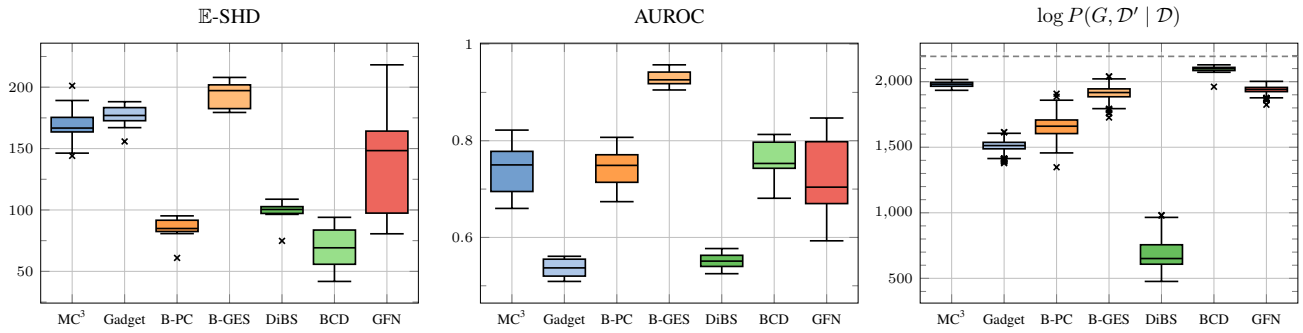

Figure 7: Bayesian structure learning of linear-Gaussian Bayesian networks with $d = 50$ nodes. Results for $\mathbb{E}$-SHD & AUROC are aggregated over 10 randomly generated datasets $\mathcal{D}$, sampled from different (ground-truth) Bayesian networks. Results for $\log P(G, \mathcal{D}' \mid \mathcal{D})$ are given for a single dataset $\mathcal{D}$; the dashed line corresponds to the log-likelihood of the ground truth graph. Labels: B-PC = Bootstrap-PC, B-GES = Bootstrap-GES, BCD = BCD Nets, GFN = DAG-GFlowNet.

to $G^\star$ can be estimated as

$$\mathbb{E}-\text{SHD} \approx \frac{1}{n} \sum_{k=1}^{n} \text{SHD}(G_k, G^\star), \tag{D.3}$$

where $\text{SHD}(G, G^\star)$ counts the number of edges changes (adding, removing, reversing an edge) necessary to move from G to $G^\star$.

D.2 SIMULATED DATA

In addition to the experiments on simulated data with graphs over $d = 20$ nodes, we also compared DAG-GFlowNet with other methods on graphs with $d = 50$ nodes. The experimental setup described in Section 6.2 remains unchanged, and the data generation process is detailed below. We show this comparison in Figure 7, in terms of $\mathbb{E}$-SHD, AUROC, and the joint log-likelihood $P(\mathcal{D}', G \mid \mathcal{D})$ on some held-out dataset $\mathcal{D}'$. We observe that DAG-GFlowNet is still competitive compared to the other algorithms, even though it suffers from a higher variance. This can be partly explained by the neural network parametrizing the forward transition probability $P_\theta(G_{t+1} \mid G_t)$ (see Sections 4.2 and 4.3) underfitting the data, and therefore not accurately matching the detailed-balance conditions, necessary for a close approximation of the posterior distribution $P(G \mid \mathcal{D})$. Similar to our observations in Section 6.3, we also noticed that algorithms that tend to perform better in terms of $\mathbb{E}$-SHD (e.g. BCD Nets, Bootstrap-PC) tend to have an order of magnitude fewer edges in the sampled DAGs.

Data generation For our experiments on simulated data, we followed the generation process described in (Lorch et al., 2021). The data was generated in the following way:

1. We sampled a DAG using an Erdős-Rényi model (Erdős and Rényi, 1960), with $2d$ edges on average; the value of the probability of creating an edge between two nodes was scaled accordingly.
2. Once the structure of the graph is known, we sampled the parameters of the linear-Gaussian model randomly from a standard Normal distribution $\mathcal{N}(0, 1)$. The linear-Gaussian model is therefore defined as, $\forall j \in [1, d]$

$$X_j = \sum_{X_i \in \text{Pa}_G(X_j)} \beta_{ij} X_i + \varepsilon,$$

   where $\beta_{ij} \sim \mathcal{N}(0, 1)$, and $\varepsilon \sim \mathcal{N}(0, 0.01)$. The defines all the conditional probability distribution of the generative model.
3. Once the full Bayesian Network is known, we used ancestral sampling to generate $N = 100$ datapoints to fill our dataset $\mathcal{D}$.

D.3 FLOW CYTOMETRY DATA

In Section 6.3, we described an application of DAG-GFlowNet to real-world flow cytometry data. In particular, we showed in Figure 5 that DAG-GFlowNet was capable of modeling a distribution that was not only capable of capturing the mode of

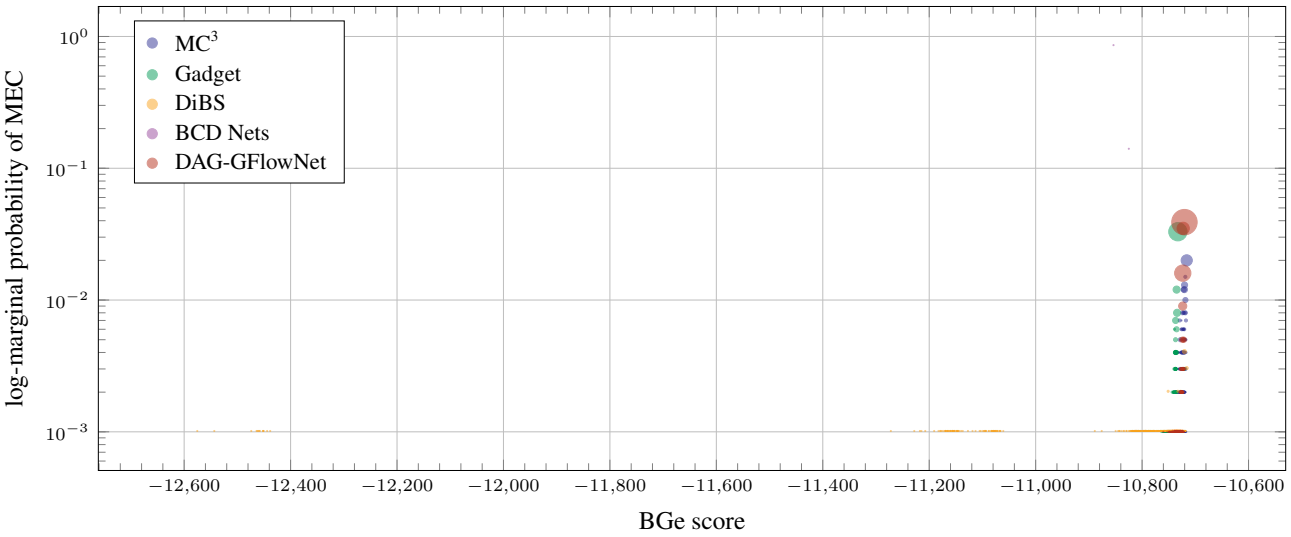

Figure 8: Coverage of the posterior approximations learned on flow cytometry data (Sachs et al., 2005). Each point corresponds to a sampled Markov equivalence class, and its size represents the number of different DAGs (in the equivalence class) sampled from the posterior approximation.

the posterior distribution (i.e., graphs with a high score), but also had diversity in the graphs sampled, both in terms of the different Markov Equivalence Classes (MECs) those graphs belong to, but also multiple unique DAG instances of the same MEC (depicted by the size of each point in Figure 5).

However for clarity, we only compared DAG-GFlowNet to methods based on MCMC in Figure 5. In Figure 8, we also added a comparison to BCD Nets (Cundy et al., 2021) and DiBS (Lorch et al., 2021), two methods based on Variational Inference. Although we saw in Table 1 that those two methods were comparing favorably against other algorithms in terms of $\mathbb{E}-\text{SHD}$ and AUROC, including against DAG-GFlowNet, we can assess more precisely the quality of the posterior approximation returned by BCD Nets and DiBS:

- Out of $1{,}000$ graphs sampled with BCD Nets, those graphs belonged to one of only two MECs (with a BGe score around $-10{,}950$). Furthermore, as shown by the size of each point, those MECs happen to only contain a single unique DAG. Overall, this means that BCD Nets only returned 2 unique DAGs (out of the $1{,}000$ samples), showing the lack of diversity of the posterior approximation learned with BCD Nets.
- DiBS sampled a significant number of very low scoring DAGs, with BGe scores as low as $-12{,}600$ (whereas the best MEC obtained with GES (Chickering, 2002) had a score of $-10{,}716.12$).
- With our choice of the BGe score, the true posterior distribution would assign the same probability to graphs in the same MEC. However, we can see that DiBS only returned graphs belonging to unique MECs, as opposed to having multiple unique DAGs from the same MEC. This shows that while DiBS has a high diversity in terms of MECs (mainly due to covering low-scoring DAGs), DiBS suffers from a lack of diversity with a single MEC, which would be expected from a faithful approximation of the posterior distribution.