BAYESIAN STRUCTURE LEARNING WITH GENERATIVE FLOW NETWORKS

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]

Mila – Quebec Artificial Intelligence Institute
deleutri@mila.quebec

ABSTRACT

In Bayesian structure learning, we are interested in inferring a distribution over the directed acyclic graph (DAG) structure of Bayesian networks, from data. Defining such a distribution is very challenging, due to the combinatorially large sample space, and approximations based on MCMC are often required. Recently, a novel class of probabilistic models, called Generative Flow Networks (GFlowNets), have been introduced as a general framework for generative modeling of discrete and composite objects, such as graphs. In this work, we propose to use a GFlowNet as an alternative to MCMC for approximating the posterior distribution over the structure of Bayesian networks, given a dataset of observations. Generating a sample DAG from this approximate distribution is viewed as a sequential decision problem, where the graph is constructed one edge at a time, based on learned transition probabilities. Through evaluation on both simulated and real data, we show that our approach, called DAG-GFlowNet, provides an accurate approximation of the posterior over DAGs, and it compares favorably against other methods based on MCMC or variational inference.

1 INTRODUCTION

Given a dataset of observations, most of the existing algorithms for structure learning of Bayesian networks (Pearl, 1988) return a single DAG (or a single equivalence class; Chickering, 2002), and in practice those may lead to poorly calibrated predictions (Madigan et al., 1994). Instead of learning a single graph candidate, we can view the problem of structure learning from a Bayesian perspective and infer the posterior over graphs $P(G \mid \mathcal{D})$, given a dataset of observations $\mathcal{D}$. Except in limited settings (Koivisto, 2006; Meilă & Jaakkola, 2006), characterizing a whole distribution over DAGs remains intractable because of its combinatorially large sample space and the complex acyclicity constraint. Therefore, we must often resort to approximations of this posterior distribution, e.g., based on MCMC (Madigan et al., 1995; Friedman & Koller, 2003; Giudici & Castelo, 2003; Niinimäki et al., 2016; Kuipers & Moffa, 2017; Viinikka et al., 2020) or, more recently, variational inference (Annadani et al., 2021; Cundy et al., 2021; Lorch et al., 2021).

In this paper, we propose to use a novel class of probabilistic models called *Generative Flow Networks* (GFlowNets; Bengio et al., 2021a;b) to approximate this posterior distribution over DAGs. A GFlowNet is a generative model over discrete and composite objects that treats the generation of a sample as a sequential decision problem. This makes it particularly appealing for modeling a distribution over graphs, where sample graphs are constructed sequentially, starting from the empty graph, by adding one edge at a time. In the context of Bayesian structure learning, we also introduce improvements over the original GFlowNet framework, including a novel flow-matching condition and corresponding loss function, and a hierarchical probabilistic model for forward transitions. We call our method *DAG-GFlowNet*, to emphasize that the support of the distribution induced by the GFlowNet is exactly the space of DAGs, unlike some variational approaches that may sample cyclic

[1]Université de Montréal, [2]Technical University of Munich, [3]KTH Stockholm, [4]CIFAR AI Chair, [5]CIFAR Azrieli Global Scholar, [6]CIFAR Senior Fellow. [*]Work done during an internship at Mila.
Code is available at: https://github.com/tristandeleu/jax-dag-gflownet/

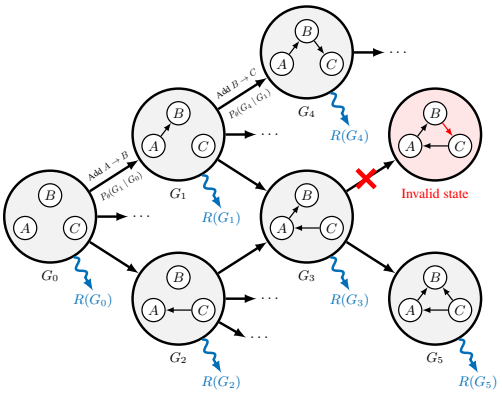


Figure 1: Structure of a GFlowNet over DAGs. The states of the GFlowNet correspond to DAGs, with the initial state G_0 being the completely disconnected graph. Each state G is complete (i.e., connected to the terminal state s_f, represented by blue arrows for brevity) and associated to a reward $R(G)$. Transitioning from one state to another corresponds to adding an edge to the graph. The state in red is invalid since the graph includes a cycle.

graphs (Annadani et al., 2021; Lorch et al., 2021). We evaluate DAG-GFlowNet on various problems with simulated and real data, on both discrete and linear-Gaussian Bayesian networks. Furthermore, we show that DAG-GFlowNet can be applied on both observational and interventional data, by modifying standard Bayesian scores (Cooper & Yoo, 1999). On smaller graphs, we also show that it is capable of learning an accurate approximation of the exact posterior distribution.

2 Background

A *Bayesian network* is a probabilistic model over d random variables $\{X_1, \dots, X_d\}$, whose joint distribution factorizes according to a DAG G as $P(X_1, \dots, X_d) = \prod_{k=1}^{d} P\big(X_k \mid \mathrm{Pa}_G(X_k)\big)$, where $\mathrm{Pa}_G(X)$ is the set of parents of node X in G. Similarly, we denote by $\mathrm{Ch}_G(X)$ the children of X; when the context is clear, we may drop the explicit dependency on G.

2.1 Generative Flow Networks

Originally introduced to encourage the discovery of diverse modes of an unnormalized distribution (Bengio et al., 2021a), *Generative Flow Networks* (GFlowNets; Bengio et al., 2021b) are a class of generative models over a discrete and structured sample space $\mathcal{X}$. The structure of a GFlowNet is defined by a DAG over some states $s \in \mathcal{S}$; in general, the sample space over which we wish to define a distribution is only a subset of the overall state space of the GFlowNet: $\mathcal{X} \subseteq \mathcal{S}$. Samples $s \in \mathcal{X}$ are constructed sequentially by following the edges of the DAG, starting from a fixed initial state s_0. We also define a special absorbing state s_f, called the terminal state, indicating when the sequential construction terminates; some of the states $s \in \mathcal{X}$ are connected to s_f, and we call them *complete states*.[1] An example of a GFlowNet is given in Figure 1, illustrating the sequential process of constructing a DAG.

In addition to the DAG structure over states, every complete state $s \in \mathcal{X}$ is associated with a *reward* $R(s) \geq 0$, indicating a notion of "preference" for certain states. The goal of a GFlowNet is to find a *flow* that satisfies, for all states $s' \in \mathcal{S}$, the following *flow-matching condition*:

$$\sum_{s \in \mathrm{Pa}(s')} F_\theta(s \to s') - \sum_{s'' \in \mathrm{Ch}(s')} F_\theta(s' \to s'') = R(s'), \tag{1}$$

where $F_\theta(s \to s') \geq 0$ is a scalar representing the flow from state s to s', typically parametrized by a neural network. Putting it in words, the overall flow going into s' is equal to the flow going out of s', plus some residual $R(s')$. To learn the parameters θ of the flow with SGD, we can turn (1) into a regression problem, e.g., using a least squares objective over sampled states.

If the conditions in (1) are satisfied for all states s', a GFlowNet induces a generative process to sample complete states $s \in \mathcal{X}$ with probability $\propto R(s)$. Starting from the initial state s_0, if we

[1] "Complete" here means that the state is a valid sample from the distribution induced by the GFlowNet. This must not be confused with a "complete graph", where all the nodes are connected to one another, when the states are DAGs (see Section 4).

sample a complete trajectory $(s_0, s_1, \dots, s_T, s, s_f)$ using the transition probability defined as

$$P(s_{t+1} \mid s_t) \propto F_\theta(s_t \to s_{t+1}), \tag{2}$$

with the conventions $s_{T+1} = s$ and $s_{T+2} = s_f$, then s is sampled with probability $P(s) \propto R(s)$. Unlike MCMC, each sample $s \in \mathcal{X}$ is constructed from scratch, starting at the initial state s_0, instead of traversing $\mathcal{X}$ from sample to sample. Therefore, the underlying Markov process of the GFlowNet does not have to be irreducible, which is typically necessary in MCMC, but merely requires all the complete states to be reachable from the initial state.

3 Detailed-balance condition

Since the flows are added together, one of the downsides of the flow-matching condition is that flows tend to be orders of magnitude larger the closer we are of the initial state (Bengio et al., 2021a), making it challenging to parametrize F_θ. Bengio et al. (2021b) proposed an alternative characterization of GFlowNets inspired by the detailed-balance equations from the literature on Markov chains (Grimmett & Stirzaker, 2020). Instead of working with flows, this condition uses a parametrization of the forward transition probability $P_\theta(s_{t+1} \mid s_t)$ directly, together with a backward transition probability $P_B(s_t \mid s_{t+1})$, a distribution over the parents of s_{t+1}, to enforce reversibility. If all the states of the GFlowNet are complete (except the terminal state s_f), which will be the case here for generating DAGs, then we show in Appendix E that we can write the *detailed-balance condition* for all transitions $s \to s'$ as follows:

$$R(s')P_B(s \mid s')P_\theta(s_f \mid s) = R(s)P_\theta(s' \mid s)P_\theta(s_f \mid s'). \tag{3}$$

Similar to Section 2.1, finding P_θ and P_B that satisfy this condition for all the transitions $s \to s'$ of the GFlowNet also yields a sampling process of complete states s with probability proportional to $R(s)$, based on the forward transition probability $P_\theta(s_{t+1} \mid s_t)$. Because this system of equations also admits many solutions, similar to (1), we can set the backward transition probability P_B to some fixed distribution (e.g., the uniform distribution over the parent states) to reduce the search space, making P_θ the only quantity to learn. To fit the parameters θ of the forward transition probability, we can minimize the following non-linear least squares objective, called the *detailed-balance loss*:

$$\mathcal{L}(\theta) = \mathbb{E}_\pi \left[\left[\log \frac{R(G')P_B(G \mid G')P_\theta(s_f \mid G)}{R(G)P_\theta(G' \mid G)P_\theta(s_f \mid G')} \right]^2 \right], \tag{4}$$

where π is a distribution over transitions with full support (i.e., $\pi(s \to s') > 0$ for all $s \to s'$).

4 GFlowNet over Directed Acyclic Graphs

Our objective in this paper is to construct a distribution over DAGs. This is a challenging problem in general, as the space of DAGs is discrete and combinatorially large. We propose to use a GFlowNet to model such a distribution; this is particularly appropriate here since graphs are composite objects, and the acyclicity constraint can be obtained by constraining the valid actions (as in Figure 1). The states of the GFlowNet are DAGs over d nodes, and therefore we will use the notation G to denote a state, in favour of s as in Section 2.1, except for the terminal state s_f. A transition $G \to G'$ in this GFlowNet corresponds to adding an edge to G to obtain the graph G'. The initial state is the fully disconnected graph G_0. Since we assume that all the states G of the GFlowNet are valid DAGs, they are all complete (i.e., connected to s_f) with a corresponding reward $R(G)$. We use a hierarchical model to define the forward transition probabilities $P_\theta(G_{t+1} \mid G_t)$, each component being parametrized using a neural network; see Appendix B.2 for details.

In the context of Bayesian structure learning, we consider the task of characterizing the posterior distribution $P(G \mid \mathcal{D})$, given a dataset of observations $\mathcal{D}$. For any DAG G, we define its reward as $R(G) = P(G)P(\mathcal{D} \mid G)$, where $P(G)$ is a prior over DAGs, and $P(\mathcal{D} \mid G)$ is the marginal likelihood. In Sec. 3, we saw that if the detailed-balance conditions are satisfied at all states of the GFlowNet, then this yields a sampling process with probability proportional to $R(G)$. Therefore, by Bayes' theorem, a GFlowNet with a structure described above, and this specific reward function, approximates the posterior $P(G \mid \mathcal{D}) \propto R(G)$. We call our method *DAG-GFlowNet*. Details about the parametrization of $P(G_{t+1} \mid G_t)$ and how to train DAG-GFlowNet are available in App. B & C.

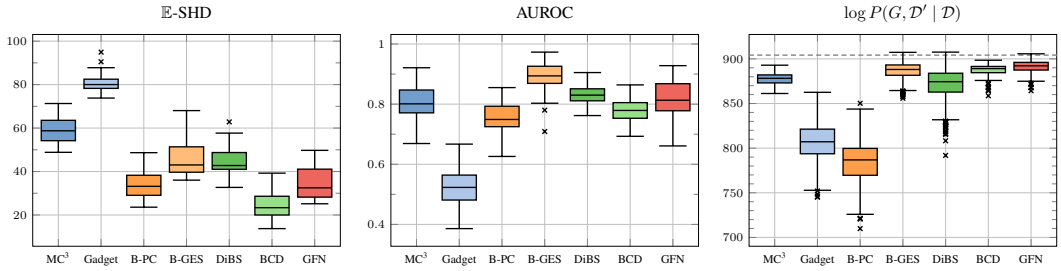


Figure 2: Bayesian structure learning of linear-Gaussian Bayesian networks with $d = 20$ nodes. Results for $\mathbb{E}$-SHD & AUROC are aggregated over 25 randomly generated datasets $\mathcal{D}$, sampled from different (ground-truth) Bayesian networks. Results for $\log P(G, \mathcal{D}' \mid \mathcal{D})$ are given for a single dataset $\mathcal{D}$; the dashed line corresponds to the log-likelihood of $G^\star$. For $\mathbb{E}$-SHD lower is better, and for AUROC & $\log P(G, \mathcal{D}' \mid \mathcal{D})$ higher is better. BCD = BCD Nets, GFN = DAG-GFlowNet.

5 Experimental results: Simulated data

We selected a representative experiment on simulated data for presentation in this section, and deferred additional results to Appendix F, including an evaluation of the accuracy of the posterior approximation induced by DAG-GFlowNet against the exact posterior, and evaluation on real flow cytometry data (Sachs et al., 2005), based on both observational, and interventional data.

For this experiment, we follow the setup of Zheng et al. (2018), and sample synthetic data from linear-Gaussian Bayesian networks with randomly generated structures. We experiment with Bayesian networks of size $d = 20$ (and $d = 50$, see Appendix F.2). The ground-truth graphs are sampled according to an Erdős-Rényi model (Erdős & Rényi, 1960), with $2d$ edges in expectation. We sampled a dataset $\mathcal{D}$ of $N = 100$ observations, and we used the BGe score (Geiger & Heckerman, 1994; Kuipers et al., 2014) to compute $R(G)$.

Since we have access to the ground-truth graph $G^\star$ that generated $\mathcal{D}$, we evaluate the performance of each algorithm with the *expected structural Hamming distance* ($\mathbb{E}$-SHD) to $G^\star$ over the posterior approximation; a detailed definition is available in Appendix F.2. We also compute the *area under the ROC curve* (AUROC; Husmeier, 2003) for the edge marginals induced by the posterior approximation, compared to the edges of $G^\star$. Finally, we compute the joint log-likelihood $\log P(G, \mathcal{D}' \mid \mathcal{D})$ on a held-out dataset $\mathcal{D}'$; we chose this metric over the log-predictive likelihood $\log P(\mathcal{D}' \mid \mathcal{D})$, as proposed by Eaton & Murphy (2007a), to study the effect of the posterior approximation $P(G \mid \mathcal{D})$.

We compare DAG-GFlowNet against 3 broad classes of Bayesian structure learning algorithms: MCMC (MC^3 (Madigan et al., 1995) & Gadget (Viinikka et al., 2020)), non-parametric DAG Bootstrapping (Friedman et al., 1999) with PC (Spirtes et al., 2000) and GES (Chickering, 2002) as structure learning routines, and variational inference (BCD Nets (Cundy et al., 2021) & DiBS (Lorch et al., 2021)). The results are shown in Figure 2. We observe that both in terms of $\mathbb{E}$-SHD & AUROC, DAG-GFlowNet, is competitive against all other methods, in particular those based on MCMC, and this does not come at a cost in terms of its predictive capacity on held-out data. In particular, we can see that the distribution induced by DAG-GFlowNet yields a predictive log-likelihood concentrated near the log-likelihood of the ground-truth DAG $G^\star$.

6 Conclusion

We have proposed a new method for Bayesian structure learning, based on a novel class of probabilistic models called GFlowNets, where the generation of a sample graph is treated as a sequential decision problem. We introduced a number of enhancements to the standard framework of GFlowNets, specifically designed for approximating a distribution over DAGs. In cases where the data is limited and measuring the epistemic uncertainty is critical, DAG-GFlowNet offers an effective solution to approximate the posterior distribution over DAGs $P(G \mid \mathcal{D})$. However, we also observed that in its current state, DAG-GFlowNet may suffer from some limitations, notably as the size of the dataset $\mathcal{D}$ increases; see Appendix D for a discussion.

While DAG-GFlownet operates on the space of DAGs directly, the structure of the GFlowNet may eventually be adapted to work with alternative representations of statistical dependencies in Bayesian networks, such as essential graphs for MECs (Chickering, 2002). Moreover, although we have already shown that DAG-GFlowNet can approximate the posterior using a mixture of observational and interventional data, we will continue to study in future work its applications to causal discovery, especially in the context of learning the structure of models with latent variables.

REFERENCES

Yashas Annadani, Jonas Rothfuss, Alexandre Lacoste, Nino Scherrer, Anirudh Goyal, Yoshua Bengio, and Stefan Bauer. Variational Causal Networks: Approximate Bayesian Inference over Causal Structures. *arXiv preprint*, 2021.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. *Neural Information Processing Systems*, 2021a.

Yoshua Bengio, Tristan Deleu, Edward J Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. GFlowNet Foundations. *arXiv preprint*, 2021b.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable Causal Discovery from Interventional Data. *Advances in Neural Information Processing Systems*, 2020.

Lars Buesing, Nicolas Heess, and Theophane Weber. Approximate Inference in Discrete Distributions with Monte Carlo Tree Search and Value Functions. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

David Chickering, Dan Geiger, and David Heckerman. Learning Bayesian Networks: Search Methods and Experimental Results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, 1995.

David Maxwell Chickering. Optimal Structure Identification With Greedy Search. *Journal of Machine Learning Research*, 2002.

Gregory F Cooper and Changwon Yoo. Causal Discovery from a Mixture of Experimental and Observational Data. *Proceedings of the Fifteenth conference on Uncertainty in Artificial Intelligence*, 1999.

Chris Cundy, Aditya Grover, and Stefano Ermon. BCD Nets: Scalable Variational Approaches for Bayesian Causal Discovery. *Advances in Neural Information Processing Systems*, 2021.

Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. *Conference on Uncertainty in Artificial Intelligence*, 2007a.

Daniel Eaton and Kevin Murphy. Belief net structure learning from uncertain interventions. 2007b.

Ralf Eggeling, Jussi Viinikka, Aleksis Vuoksenmaa, and Mikko Koivisto. On Structure Priors for Learning Bayesian Networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

Byron Ellis and Wing Hung Wong. Learning Causal Bayesian Network Structures from Experimental Data. *Journal of the American Statistical Association*, 2008.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 1960.

Gonçalo Rui Alves Faria, Andre Martins, and Mario A. T. Figueiredo. Differentiable Causal Discovery Under Latent Interventions. In *First Conference on Causal Learning and Reasoning*, 2022.

Nir Friedman and Daphne Koller. Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 2003.

Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data Analysis with Bayesian Networks: A Bootstrap Approach. *Proceedings of the Fifteenth conference on Uncertainty in Artificial Intelligence*, 1999.

Dan Geiger and David Heckerman. Learning Gaussian Networks. In *Uncertainty in Artificial Intelligence*. 1994.

Paolo Giudici and Robert Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine learning*, 2003.

Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 2020.

David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 1995.

Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 2003.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *International Conference on Machine Learning*, 2020.

Mikko Koivisto. Advances in Exact Bayesian Structure Discovery in Bayesian Networks. *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, 2006.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Jack Kuipers and Giusi Moffa. Partition MCMC for Inference on Acyclic Digraphs. *Journal of the American Statistical Association*, 2017.

Jack Kuipers, Giusi Moffa, and David Heckerman. Addendum on the scoring of gaussian directed acyclic graphical models. *The Annals of Statistics*, 2014.

Jack Kuipers, Polina Suter, and Giusi Moffa. Efficient Sampling and Structure Learning of Bayesian Networks. *Journal of Computational and Graphical Statistics*, 2021.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning Deep Generative Models of Graphs. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Lars Lorch, Jonas Rothfuss, Bernhard Schölkopf, and Andreas Krause. DiBS: Differentiable Bayesian Structure Learning. *Advances in Neural Information Processing Systems*, 2021.

David Madigan, Jonathan Gavrin, and Adrian E Raftery. Enhancing the Predictive Performance of Bayesian Graphical Models. 1994.

David Madigan, Jeremy York, and Denis Allard. Bayesian Graphical Models for Discrete Data. *International Statistical Review*, 1995.

Vikash Mansinghka, Charles Kemp, Thomas Griffiths, and Joshua Tenenbaum. Structured Priors for Structure Learning. *Conference on Uncertainty in Artificial Intelligence*, 2006.

Marina Meilă and Tommi Jaakkola. Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, 2006.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Joris M Mooij, Sara Magliacane, and Tom Claassen. Joint Causal Inference from Multiple Contexts. *Journal of Machine Learning Research*, 2020.

Teppo Niinimäki, Pekka Parviainen, and Mikko Koivisto. Structure Discovery in Bayesian Networks by Sampling Partial Orders. *The Journal of Machine Learning Research*, 2016.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2009.

Gavin A Rummery and Mahesan Niranjan. On-line Q-learning using Connectionist Systems. 1994.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 2005.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, Prediction, and Search*. MIT press, 2000.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep Reinforcement Learning and the Deadly Triad. *arXiv preprint*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017.

Jussi Viinikka, Antti Hyttinen, Johan Pensar, and Mikko Koivisto. Towards Scalable Bayesian Learning of Causal DAGs. *Advances in Neural Information Processing Systems*, 2020.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*, 2018.

Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal Discovery with Reinforcement Learning. *International Conference on Learning Representations*, 2020.

APPENDIX

A RELATED WORK

Markov chain Monte Carlo Methods based on MCMC have been particularly popular in Bayesian structure learning to approximate the posterior distribution. Structure MCMC (MC^3; Madigan et al., 1995) simulates a Markov chain in the space of DAGs, through local moves (e.g., adding or removing an edge). Working directly with DAGs leads to slow mixing though; to improve mixing, Friedman & Koller (2003) proposed a sampler in the space of node orders, that introduced a bias (Ellis & Wong, 2008). This was further refined by either modifying the underlying space of the Markov chain (Kuipers & Moffa, 2017; Niinimäki et al., 2016), or its local moves (Mansinghka et al., 2006; Eaton & Murphy, 2007a; Kuipers et al., 2021). Recently, Viinikka et al. (2020) incorporated many of these advances into an efficient MCMC sampler called *Gadget*.

Variational Inference In the context of structure learning, applying the recent advances in approximate inference based on gradient methods can be difficult due to the discrete nature of the problem (Lorch et al., 2021). Cundy et al. (2021) decomposed the adjacency matrix of a DAG into a triangular matrix and a permutation, and used a continuous relaxation to parametrize a distribution over permutations. Other methods (Annadani et al., 2021; Lorch et al., 2021) encode the acyclicity constraint into a soft prior $P(G)$, based on continuous characterizations of acyclicity (Zheng et al., 2018). While the effect of this prior can be made arbitrarily strong, this does not guarantee that the graphs sampled from the resulting distribution are acyclic. By contrast, our approach guarantees by construction that the support of the posterior approximation is exactly the space of DAGs.

Sequential decisions In this work, we treat the construction of a sample graph from the posterior as a sequential decision problem, starting from the empty graph and adding one edge at a time. Li et al. (2018) use a similar process for creating a generative model over graphs with a fixed ordering over nodes. Similarly, although they do not consider a distribution over graphs, Buesing et al. (2020) use a variant of Monte Carlo Tree Search to approximate a distribution over a pre-specified ordering of discrete random variables. Our method, based on Generative Flow Networks, does not make any assumption on the order in which the edges are added, and multiple edge insertion sequences may lead to the same DAG. Zhu et al. (2020) learn a single high-scoring structure using RL; however, unlike our approach, the creation of this graph does not involve sequential decisions.

B GFLOWNET OVER DIRECTED ACYCLIC GRAPHS

B.1 STRUCTURE OF THE GFLOWNET

We consider a GFlowNet where the states are DAGs over d (labeled) nodes. Since the states of the GFlowNet are graphs, we will use the notation G to denote a state, in favour of s as in Section 2.1, except for the terminal state s_f. A transition $G \to G'$ in this GFlowNet corresponds to adding an edge to G to obtain the graph G'; in other words, the graphs are constructed one edge at a time, starting from the initial state G_0, which is the fully disconnected graph over d nodes. Since we assume that all the states G of the GFlowNet are valid DAGs, they are all complete (i.e., connected to the terminal state s_f) with a corresponding reward $R(G)$. Figure 1 shows an illustration of the structure of such a GFlowNet, where the states are DAGs over $d = 3$ nodes. This application to graphs highlights the importance of the DAG structure of the GFlowNet, since there can be multiple paths leading to the same state: for any graph G with k edges, there are $k!$ possible paths from G_0 leading to G, because the edges of G may have been added in any order.

To guarantee the integrity of the GFlowNet, we have to ensure that adding a new edge to some state G also yields a valid DAG, meaning that this edge (1) must not be already present in G, and (2) must not introduce a cycle. Fortunately, we can filter out invalid actions using some mask $\boldsymbol{m}$ associated to the graph, built from the adjacency matrix of G and the transitive closure of its transpose, and

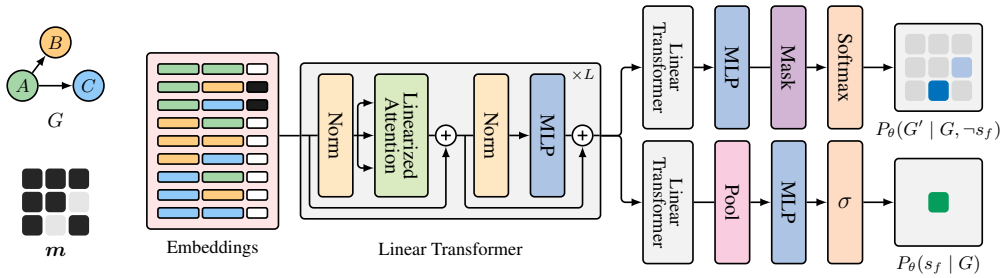


Figure 3: Neural network architecture of the forward transition probabilities $P_\theta(G_{t+1} \mid G_t)$. The input graph G is encoded as a set of d^2 possible edges (including self-loops). Each directed edge is embedded using the embeddings of its source and target, with an additional vector indicating whether the edge is present in G. These embeddings are fed into a Linear Transformer (Katharopoulos et al., 2020), with two separate output heads. The first head (above) gives the probability to add a new edge $P_\theta(G' \mid G, \neg s_f)$, using the mask $\boldsymbol{m}$ associated to G to filter out invalid actions; here, the only valid actions are either adding $B \to C$, or $C \to B$. The second head (below) gives the probability to terminate the trajectory $P_\theta(s_f \mid G)$.

that can be updated efficiently after the addition an edge (Giudici & Castelo, 2003). A description of this update is given in Appendix G for completeness.

B.2 Forward transition probabilities

Following Section 3, the GFlowNet may be parametrized only by the forward transition probabilities $P_\theta(G_{t+1} \mid G_t)$; here, G_{t+1} might be the terminal state s_f by abuse of notation. To make sure that the detailed-balance conditions can be satisfied, we need to define these transition probabilities using a sufficiently expressive function, such as a neural network. We use a hierarchical model, where the forward transition probabilities are defined using two neural networks: (1) a network modeling the probability of terminating $P_\theta(s_f \mid G)$, and (2) another giving the probability $P_\theta(G' \mid G, \neg s_f)$ of transitioning to a new graph G', given that we do not terminate. The probability of taking a transition $G \to G'$ is then given by

$$P_\theta(G' \mid G) = \big(1 - P_\theta(s_f \mid G)\big) P_\theta(G' \mid G, \neg s_f). \tag{5}$$

In practice, as G' is the result of adding an edge to the DAG G, we can model $P_\theta(G' \mid G, \neg s_f)$ as a probability distribution over the d^2 possible edges one could add to G—this includes self-loops, for simplicity, even though these actions are guaranteed to be invalid. We can use the mask $\boldsymbol{m}$ introduced in Appendix B.1 to filter out actions that would not lead to a valid DAG G' and set $P_\theta(G' \mid G, \neg s_f) = 0$ for any invalid action (as well as normalize P_θ accordingly).

B.3 Parametrization with Linear Transformers

Beyond having enough capacity to satisfy as well as possible the detailed-balance condition at all states, we choose to parametrize the forward transition probabilities with neural networks to benefit from their capacity to generalize to states not encountered during training. In practice, instead of defining two separate networks to parametrize $P_\theta(s_f \mid G)$ and $P_\theta(G' \mid G, \neg s_f)$, we use a single neural network with a common backbone and two separate heads, to benefit from parameter sharing. The full architecture is given in Figure 3.

Our choice of neural network architecture is motivated by multiple factors: we want an architecture (1) that is invariant to the order of the inputs, since G is represented as a set of edges, (2) that transforms a set of input edges into a set of output probabilities for each edge to be added, in order to define $P_\theta(G' \mid G, \neg s_f)$, and (3) whose parameters θ do not scale too much with d. A natural option would be to use a Transformer (Vaswani et al., 2017); however, because the size of our inputs is d^2, the self-attention layers would scale as d^4, and this would severely limit our ability to apply our method to model a distribution over larger DAGs.

We opted for a Linear Transformer (Katharopoulos et al., 2020) instead, which has the advantage to not suffer from this quadratic scaling in the input size. This architecture relies on a linearized attention mechanism, defined as

$$Q = \boldsymbol{x}W_Q \qquad K = \boldsymbol{x}W_K \qquad V = \boldsymbol{x}W_V$$
$$\mathrm{LinAttn}_k(\boldsymbol{x}) = \frac{\sum_{j=1}^{J}\left(\phi(Q_k)^\top \phi(K_j)\right)V_j}{\sum_{j=1}^{J}\phi(Q_k)^\top\phi(K_j)}, \tag{6}$$

where $\boldsymbol{x}$ is the input of the linearized attention layer, $\phi(\cdot)$ is a non-linear feature map, J is the size of the input $\boldsymbol{x}$ (in our case, $J = d^2$), and Q, K, and V are linear transformations of $\boldsymbol{x}$ corresponding to the queries, keys, and values respectively, as is standard with Transformers.

C Application to Bayesian Structure Learning

We are given a dataset $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}$ of N observations $\boldsymbol{x}^{(j)}$, each consisting of d elements. We consider the task of characterizing the posterior distribution $P(G \mid \mathcal{D})$ over Bayesian networks that model these observations. We assume that the samples in $\mathcal{D}$ are iid. and fully-observed. As an alternative to MCMC (Madigan et al., 1995) or variational inference (Lorch et al., 2021), we approximate the posterior distribution over DAGs using a GFlowNet, as described in the previous section. For any DAG G, we will define its reward as the joint probability

$$R(G) = P(G)P(\mathcal{D} \mid G), \tag{7}$$

where $P(G)$ is a prior over DAGs (Eggeling et al., 2019), and $P(\mathcal{D} \mid G)$ is the marginal likelihood. In Sec. 3, we saw that if the detailed-balance conditions are satisfied for all the states of the GFlowNet, then this yields a sampling process with probability proportional to $R(G)$. Therefore, by Bayes' theorem, a GFlowNet with the specific reward function in (7) approximates the posterior distribution $P(G \mid \mathcal{D}) \propto R(G)$. We call our method *DAG-GFlowNet*.

C.1 Modularity & computational efficiency

Following prior works on Bayesian structure learning, we assume that both the priors over parameters $P(\phi \mid G)$ of the Bayesian network (required to compute the marginal likelihood) and over structures $P(G)$ are *modular* (Heckerman et al., 1995; Chickering et al., 1995). As a consequence the reward $R(G)$ is also modular, and its logarithm can be written as a sum of local scores that only depend on individual variables and their parents in G:

$$\log R(G) = \sum_{j=1}^{d} \mathrm{LocalScore}\big(X_j \mid \mathrm{Pa}_G(X_j)\big). \tag{8}$$

Note that with our choice of reward, $\log R(G)$ corresponds to the Bayesian score (Koller & Friedman, 2009). Examples of modular scores include the BDe score (Heckerman et al., 1995) and the BGe score (Geiger & Heckerman, 1994; Kuipers et al., 2014). In order to fit the parameters θ of the GFlowNet, we will use the detailed-balance loss in (4). We can observe that this loss function only involves the difference in log-rewards $\log R(G') - \log R(G)$ between two consecutive states, where G' is the result of adding some edge $X_i \to X_j$ to the DAG G. Using our assumption of modularity, we can therefore compute this difference efficiently, as the terms in (8) remain unchanged for $j' \neq j$:

$$\log R(G') - \log R(G) = \mathrm{LocalScore}\big(X_j \mid \mathrm{Pa}_G(X_j) \cup \{X_i\}\big) - \mathrm{LocalScore}\big(X_j \mid \mathrm{Pa}_G(X_j)\big). \tag{9}$$

This difference in local scores is sometimes called the *delta score*, or the *incremental value* (Friedman & Koller, 2003), and has been employed in the literature to improve the efficiency of search algorithms (Chickering, 2002; Koller & Friedman, 2009).

C.2 Off-policy learning

As the number of states in DAG-GFlowNet is super-exponential in d, the number of nodes in each DAG G, it would be impractical to minimize the detailed-balance loss for all possible transitions

$G \to G'$. We can minimize this loss in expectation using a distribution $\pi(G \to G')$ with full support over transitions:

$$\mathcal{L}(\theta) = \mathbb{E}_{\pi}\left[\left[\log \frac{R(G')P_B(G \mid G')P_\theta(s_f \mid G)}{R(G)P_\theta(G' \mid G)P_\theta(s_f \mid G')}\right]^2\right]. \tag{10}$$

This distribution $\pi(G \to G')$ can be arbitrary; for example, we can use $P_\theta(G' \mid G)$ directly and learn it *on-policy* (Rummery & Niranjan, 1994), as long as it assigns non-zero probability to any next state G'.

Taking inspiration from Deep Q-learning (Mnih et al., 2015), we instead learn P_θ using *off-policy* data. Transitions $G \to G'$ are collected based on $P_\theta(G' \mid G)$, along with their corresponding delta score ((9)), and they are stored in a replay buffer. We can also sample some transitions uniformly at random, with probability ε, to encourage exploration. To estimate $\mathcal{L}(\theta)$ and update the parameters θ, we can then sample a mini-batch of transitions randomly from the replay buffer. Moreover, again inspired by Deep Q-learning (Van Hasselt et al., 2018), we found it advantageous to evaluate $P_{\bar{\theta}}(s_f \mid G')$ in (4) with a separate target network—where the parameters $\bar{\theta}$ are updated periodically.

D 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 (4) was more challenging. This can be explained by the fact that with a larger amount of data, the posterior distribution becomes very peaky (Koller & 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 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.

E 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. (2021b) 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. (2021b), 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., 2021b, Def. 17) is given by

$$F(s_t)P_F(s_{t+1} \mid s_t) = F(s_{t+1})P_B(s_t \mid s_{t+1}). \tag{11}$$

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

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

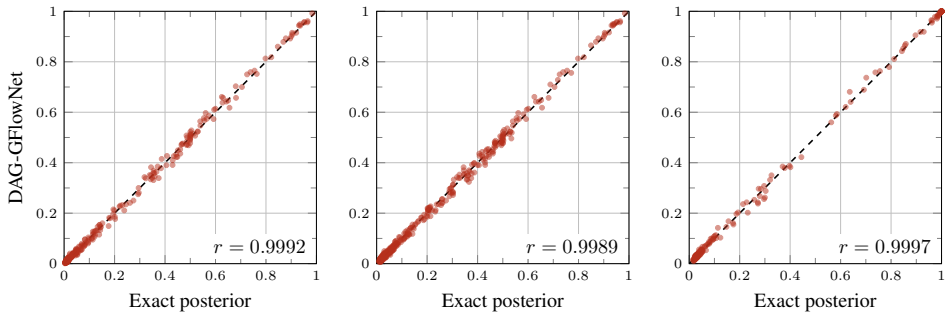
where $F(s \to s')$ represents the flow from state s to s', as described in Section 2.1, $F(s)$ is the total flow through state s, and we used Proposition 4 & Equation 34 of Bengio et al. (2021b) to introduce $F(s_t)$ and $R(s_t)$ respectively. Replacing $F(\cdot)$ in (11) yields the expected condition:

$$R(s_t)P_F(s_{t+1} \mid s_t)P_F(s_f \mid s_{t+1}) = R(s_{t+1})P_B(s_t \mid s_{t+1})P_F(s_f \mid s_t). \tag{12}$$

The original formulation in (11) 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).

F Additional Experimental results

F.1 Comparison with the exact posterior



(a) Edge features (b) Path features (c) Markov features

Figure 4: Comparison between the exact posterior distribution and the posterior approximation from DAG-GFlowNet, for different structural features: (a) edge features $X_i \to X_j$, (b) path features $X_i \rightsquigarrow X_j$, (c) Markov features $X_i \sim_M X_j$. Each point corresponds to a feature computed for specific variables X_i and X_j in a graph over $d = 5$ nodes, either based on the exact posterior (x-axis), or the posterior approximation found with the GFlowNet (y-axis). We repeated this experiment with 20 different (ground-truth) DAGs. The Pearson correlation coefficient r is included in the bottom-right corner of each plot.

In order to measure the quality of the posterior approximation returned by DAG-GFlowNet, we want to compare it with the exact posterior distribution $P(G \mid \mathcal{D})$. However, the latter requires an exhaustive enumeration of all possible DAGs, which is only feasible for graphs with no more than 5 nodes. Therefore, we sampled $N = 100$ datapoints from a randomly generated (under an Erdős-Rényi model; Erdős & Rényi, 1960) linear-Gaussian Bayesian network over $d = 5$ variables. We used the BGe score to compute the reward $R(G) = P(G)P(\mathcal{D} \mid G)$. The exact posterior distribution $P(G \mid \mathcal{D})$ is obtained by enumerating all 29,281 possible DAGs over 5 nodes and computing their respective rewards $R(G)$ (normalized to sum to 1).

We evaluated the quality of the approximation based on the probability of various structural features. For example, using samples $\{G_1, G_2, \ldots, G_n\}$ from the posterior approximation, the marginal probability of an *edge feature* $X_i \to X_j$ can be estimated with

$$P_\theta(X_i \to X_j \mid \mathcal{D}) \approx \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}(X_i \to X_j \in G_k), \tag{13}$$

where $\mathbf{1}(\cdot)$ is the indicator function. For the exact posterior, we can obtain the posterior probability of the edge feature by simply marginalizing over $P(G \mid \mathcal{D})$. Similarly, we compute (or estimate) the marginal probability of a *path feature* $X_i \rightsquigarrow X_j$, i.e., of a (directed) path existing from X_i to X_j, and the probability of a *Markov feature* $X_i \sim_M X_j$, i.e., of X_i being in the Markov blanket of X_j (Friedman & Koller, 2003). These features are computed for all variables X_i and X_j in the Bayesian network.

In Figure 4, we compare the probabilities of these features for both the exact posterior and the distribution induced by DAG-GFlowNet, where we repeated the experiment above with 20 different (ground-truth) Bayesian networks. We observe that the probabilities of all structural features estimated by the GFlowNet are strongly correlated with the exact marginal probabilities. This shows that DAG-GFlowNet is capable of learning a very accurate approximation of the posterior distribution over graphs $P(G \mid \mathcal{D})$.

F.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 5 remains unchanged. We show this comparison in Figure 5, in terms of $\mathbb{E}$-SHD,

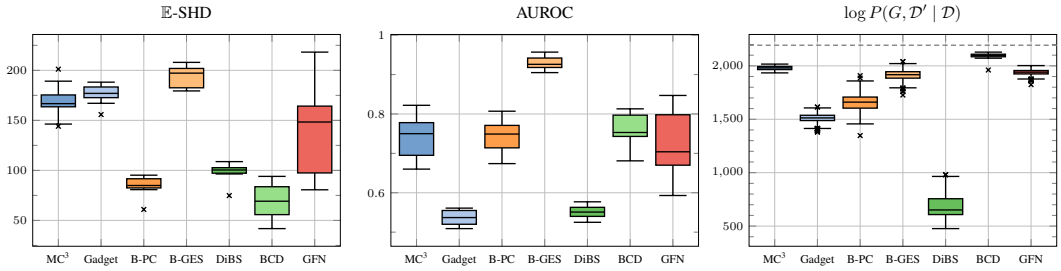


Figure 5: Bayesian structure learning of linear-Gaussian Bayesian networks with $d = 50$ nodes. Results for $\mathbb{E}$-SHD & AUROC are aggregated over 10 randomly generated datasets $\mathcal{D}$, sampled from different (ground-truth) Bayesian networks. Results for $\log P(G, \mathcal{D}' \mid \mathcal{D})$ 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.

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 appendices B.2 and B.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 Appendix F.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.

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 to $G^\star$ can be estimated as

$$\mathbb{E}-\text{SHD} \approx \frac{1}{n} \sum_{k=1}^{n} \text{SHD}(G_k, G^\star), \tag{14}$$

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

F.3 Application: Flow Cytometry Data

We also evaluated DAG-GFlowNet on real-world flow cytometry data (Sachs et al., 2005) to learn protein signaling pathways. The data consists of continuous measurements of $d = 11$ phosphoproteins in individual T-cells. Out of all the measurements, we selected the $N = 853$ observations corresponding to the first experimental condition of Sachs et al. (2005) as our dataset $\mathcal{D}$. Following prior work on structure learning, we used the DAG inferred by Sachs et al. (2005), containing $d = 11$ nodes and 17 edges, as our graph of reference (ground-truth). However, it should be noted that this "consensus graph" may not represent a realistic and complete description of the system being modeled here (Mooij et al., 2020). We standardized the data, and used the BGe score to compute $R(G)$.

In Table 1, we compare the expected SHD and the AUROC obtained with DAG-GFlowNet and other approaches. While BCD Nets and Bootstrap PC have a smaller $\mathbb{E}$-SHD, suggesting that the distribution is concentrated closer to the consensus graph, in reality they tend to be more conservative and sample graphs with fewer edges. Overall, DAG-GFlowNet offers a good trade-off between performance (as measured by the $\mathbb{E}$-SHD and the AUROC), and getting a distribution that assigns higher probability to DAGs with more edges. We also observed that 1.50% of the graphs sampled with DiBS contained a cycle.

Beyond these metrics, we would like to test if the advantages of Bayesian structure learning are also reflected in the distribution induced by DAG-GFlowNet. In particular, we want to study (1) if

Table 1: Learning protein signaling pathways from flow cytometry data (Sachs et al., 2005). All results include a 95% confidence interval estimated with bootstrap resampling.

	$\mathbb{E}$-# Edges	$\mathbb{E}$-SHD	AUROC
MC^3	10.96 ± 0.09	22.66 ± 0.11	0.508
Gadget	10.59 ± 0.09	21.77 ± 0.10	0.479
Bootstrap GES	11.11 ± 0.09	23.07 ± 0.11	**0.548**
Bootstrap PC	7.83 ± 0.04	20.65 ± 0.06	0.520
DiBS	12.62 ± 0.16	23.32 ± 0.14	0.518
BCD Nets	4.14 ± 0.09	$\mathbf{18.14 \pm 0.09}$	0.510
DAG-GFlowNet	11.25 ± 0.09	22.88 ± 0.10	0.541

this distribution covers multiple high-scoring DAGs, instead of being peaked at a single most likely graph, and (2) if the GFlowNet can sample a variety of DAGs from the same Markov equivalence class (MEC), showing the inherent uncertainty over equivalent graphs. In Figure 6, we visualize the MECs of the graphs sampled with DAG-GFlowNet, and two methods based on MCMC (MC^3 and Gadget); other baselines were excluded for clarity. The size of each point represents the number of unique DAGs in the corresponding MEC. We observe that DAG-GFlowNet largely follows the behavior of MCMC: the distribution does not collapse to a single most-likely DAG, and covers multiple MECs. Moreover, the GFlowNet is also capable of sampling different equivalent DAGs (corresponding to larger points), showing again that the distribution does not collapse to a single representative of the MECs with higher marginal probability. We also observe that the maximum a posteriori MEC found by DAG-GFlowNet reaches a higher score than the one found with Gadget, but a lower score than MC^3; as a point of reference, the score of the best MEC obtained with GES (Chickering, 2002) is $-10{,}716.12$.

F.4 Application: Interventional data

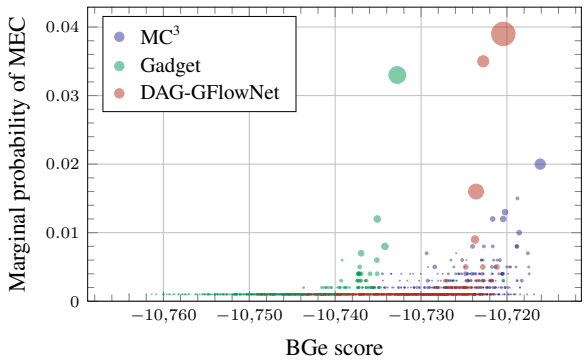


Figure 6: Coverage of the posterior approximations learned on flow cytometry data (Sachs et al., 2005). Each point corresponds to a sampled Markov equivalence class, and its size represents the number of different DAGs (in the equivalence class) sampled from the posterior approximation.

In addition to the observational data we used in Appendix F.3, Sachs et al. (2005) also provided flow cytometry data under different experimental conditions, where the T-cells were perturbed with some reagents; this effectively corresponds to interventional data (Pearl, 2009). Although a molecular intervention may be imperfect and affect multiple proteins (Eaton & Murphy, 2007b), we assume here that these interventions are perfect, and the intervention targets are known. We used a discretized dataset of $N = 5{,}400$ samples from 9 experimental conditions—of which 6 are interventions. We modified the BDe score to handle this mixture of observational and interventional data (Cooper & Yoo, 1999).

In Table 2, we compare with Eaton & Murphy (2007b), which compute the AUROC of the exact posterior using dynamic programming, therefore working as an upper bound for what a posterior approximation can achieve. They achieve this at the expense of computing only edge marginals, without providing access to a distribution over DAGs. We also use the modified BDe score with MC^3, which predicts sparser graphs with higher SHD than DAG-GFlowNet, but lower AUROC. Note that this setup is different from previous works which use continuous data instead (Brouillard et al., 2020; Faria et al., 2022).

Table 2: Combining discrete interventional and observational flow cytometry data (Sachs et al., 2005). ⋆Result reported in Eaton & Murphy (2007b).

	$\mathbb{E}$-# Edges	$\mathbb{E}$-SHD	AUROC
Exact posterior⋆	—	—	$\mathbf{0.816}$
MC^3	25.97 ± 0.01	$\mathbf{25.08 \pm 0.02}$	0.665
DAG-GFlowNet	30.66 ± 0.04	27.77 ± 0.03	0.700

G Definition and update of the mask over actions

In Appendix B.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 & 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 7, 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.

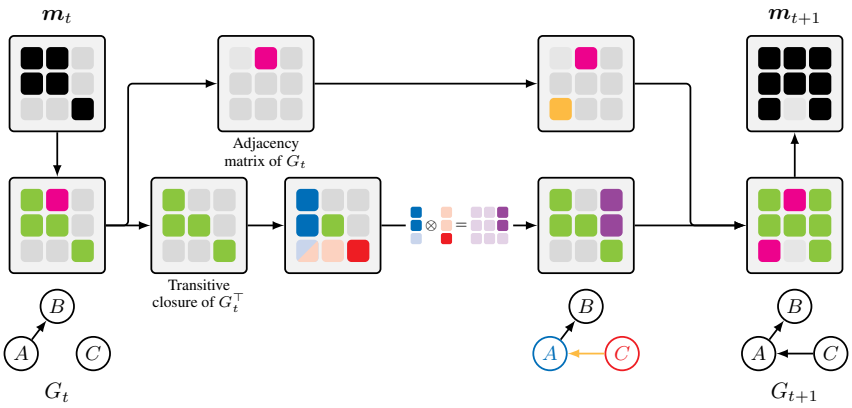


Figure 7: Online update of the mask $\boldsymbol{m}$. The mask $\boldsymbol{m}_t$ associated with G_t represents (in black) the edges that cannot be added to G_t to obtain a valid DAG. $\boldsymbol{m}_t$ is decomposed in two parts: the adjacency matrix of G_t (top), and the transitive closure of $G_t^\top$ (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.