Supplementary Material for: Learning Sparse Nonparametric DAGs

A Proofs

In this Appendix, we prove Proposition 1. For completeness, note that

$$\mathcal{F} = \{f \mid f(u) = \mathsf{MLP}(u; A^{(1)}, \dots, A^{(h)}), \\ f \text{ independent of } u_k\}$$

and

$$\mathcal{F}_0 = \{f \mid f(u) = \mathsf{MLP}(u; A^{(1)}, \dots, A^{(h)}), \\ A^{(1)}_{bk} = 0, \forall b = 1, \dots, m_1\}.$$

We omit the bias terms in each layer as it does not affect the statement.

Proof of Proposition 1. We will show that $\mathcal{F} \subseteq \mathcal{F}_0$ and $\mathcal{F}_0 \subseteq \mathcal{F}$.

(1) $\mathcal{F}_0 \subseteq \mathcal{F}$: for any $f_0 \in \mathcal{F}_0$, we have $f_0(u) = \mathsf{MLP}(u; A^{(1)}, \dots, A^{(h)})$, where $A^{(1)}_{bk} = 0$ for all $b = 1, \dots, m_1$. Hence the linear function $A^{(1)}u$ is independent of u_k. Therefore,

$$\begin{aligned} f_0(u) &= \mathsf{MLP}(u; A^{(1)}, \dots, A^{(h)}) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(A^{(1)}u))) \end{aligned}$$

is also independent of u_k, which means $f_0 \in \mathcal{F}$.

(2) $\mathcal{F} \subseteq \mathcal{F}_0$: for any $f \in \mathcal{F}$, we have $f(u) = \mathsf{MLP}(u; A^{(1)}, \dots, A^{(h)})$ and f is independent of u_k. We will show that $f \in \mathcal{F}_0$ by constructing a matrix $\tilde{A}^{(1)}$, such that

$$f(u) = \mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \dots, A^{(h)}) \tag{14}$$

and $\tilde{A}^{(1)}_{bk} = 0$ for all $b = 1, \dots, m_1$.

Let $\tilde{u}$ be the vector such that $\tilde{u}_k = 0$ and $\tilde{u}_{k'} = u_k$ for all $k' \neq k$. Since $\tilde{u}$ and u differ only on the kth dimension, and f is independent of u_k, we have

$$f(u) = f(\tilde{u}) = \mathsf{MLP}(\tilde{u}; A^{(1)}, \dots, A^{(h)}). \tag{15}$$

Now define $\tilde{A}^{(1)}$ be the matrix such that $\tilde{A}^{(1)}_{bk} = 0$ and $\tilde{A}^{(1)}_{bk'} = A^{(1)}_{bk}$ for all $k' \neq k$. Then we have the following observation: for each entry $s \in \{1, \dots, m_1\}$,

$$\begin{aligned} (\tilde{A}^{(1)}u)_s &= \sum_{k'=1}^{d} \tilde{A}_{sk'}u_{k'} = \sum_{k' \neq k} A_{sk'}u_{k'} \\ &= \sum_{k'=1}^{d} A_{sk'}\tilde{u}_{k'} = (A^{(1)}\tilde{u})_s. \end{aligned}$$

Hence,

$$\tilde{A}^{(1)}u = A^{(1)}\tilde{u}. \tag{16}$$

Therefore, by (15)

$$\begin{aligned} f(u) &= f(\tilde{u}) \\ &= \mathsf{MLP}(\tilde{u}; A^{(1)}, \dots, A^{(h)}) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(A^{(1)}\tilde{u}))) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(\tilde{A}^{(1)}u))) \\ &= \mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \dots, A^{(h)}) \end{aligned}$$

By definition of $\mathcal{F}_0$, we know that $\mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \dots, A^{(h)}) \in \mathcal{F}_0$. Thus, $f \in \mathcal{F}_0$ and we have completed the proof. □

B Experiment details

Baselines We consider the following baselines.

- Fast greedy equivalence search (FGS)[2] (Ramsey et al., 2017) is based on greedy search and assumes linear dependency between variables.
- Greedy equivalence search with generalized scores (GSGES)[3] (Huang et al., 2018) is also based on greedy search, but uses generalized scores without assuming a particular model class.
- DAG-GNN (GNN)[4] (Yu et al., 2019) learns a (noisy) nonlinear transformation of a linear SEM using neural networks.
- NOTEARS (Linear)[5] (Zheng et al., 2018) learns a linear SEM using continuous optimization.
- Causal additive model (CAM)[6] (Bühlmann et al., 2014) learns an additive SEM by leveraging efficient nonparametric regression techniques and greedy search over edges.

For all experiments, default parameter settings are used, except for CAM where both preliminary neighborhood selection and pruning are applied.

Simulation Given the graph G, we simulate the SEM $X_j = f_j(X_{\mathrm{pa}(j)}) + z_j$ for all $j \in [d]$ in the topological order induced by G. We consider the following instances of f_j:

[2] https://github.com/bd2kccd/py-causal
[3] https://github.com/Biwei-Huang/Generalized-Score-Functions-for-Causal-Discovery/
[4] https://github.com/fishmoon1234/DAG-GNN
[5] https://github.com/xunzheng/notears
[6] https://cran.r-project.org/package=CAM

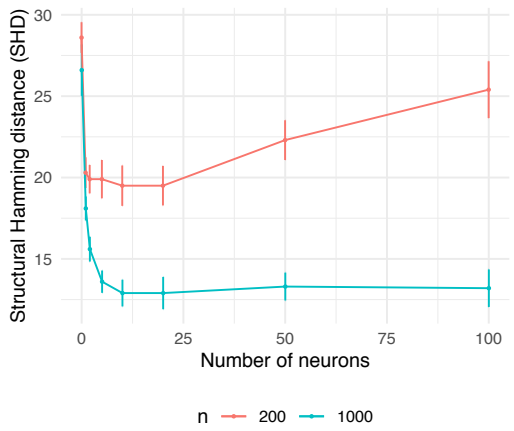

Figure 3: SHD (lower is better) with varying hidden layer size in NOTEARS-MLP.

- Additive GP: $f_j(X_{\mathrm{pa}(j)}) = \sum_{k\in\mathrm{pa}(j)} f_{jk}(X_k)$, where each f_{jk} is a draw from Gaussian process with RBF kernel with length-scale one.
- Index model: $f_j(X_{\mathrm{pa}(j)}) = \sum_{m=1}^{3} h_m(\sum_{k\in\mathrm{pa}(j)} \theta_{jmk} X_k)$, where $h_1 = \tanh$, $h_2 = \cos$, $h_3 = \sin$, and each θ_{jmk} is drawn uniformly from range $[-2,-0.5]\cup[0.5,2]$.
- MLP: f_j is a randomly initialized MLP with one hidden layer of size 100 and sigmoid activation.
- GP: f_j is a draw from Gaussian process with RBF kernel with length-scale one.

In all settings, z_j is i.i.d. standard Gaussian noise.

C Sensitivity to number of hidden units

We also investigated the effect of number of hidden units in the NOTEARS-MLP estimate. It is well-known that as the size of the hidden layer increases, the functions representable by an MLP become more flexible. On the other hand, larger networks require more samples to estimate the parameters. Indeed, Figure 3 confirms this intuition. We plot the SHD with varying number of hidden units ranging from zero (*i.e.* linear function) to 100 units, using $n = 1000$ and $n = 200$ samples generated from the additive GP model on SF2 graph with $d = 20$ nodes. One can first observe a sharp phase transition between zero and very few hidden units, which suggests the power of nonlinearity. Moreover, as the number of hidden units increases to 20, the performance for both $n = 1000$ and $n = 200$ steadily improves, in which case the increased flexibility brings benefit. However, as we further increase the number of hidden units, while SHD for $n = 1000$ remains similar, the SHD for $n = 200$ deteriorates, hinting at the lack of samples to take advantage of the increased flexibility.

D Additional results

Full comparison We show {SHD, FDR, TPR, FPR} results on all {ER1, ER2, ER4, SF1, SF2, SF4} graphs in Figure 4, 5, 6, 7 respectively. Similarly, see Figure 8, 9, 10, 11 for full comparison with CAM. As in Figure 1, each row is a random graph model, each column is a type of SEM. Overall NOTEARS-MLP has low FDR/FPR and high TPR, and same for NOTEARS-Sob on additive GP. Also observe that in most settings GNN has low FDR as well as low TPR, which is a consequence of only predicting a small number of edges.

Complexity and runtime Recall that numerical evaluation of matrix exponential involves solving linear systems, hence the time complexity is typically $O(d^3)$ for a dense $d \times d$ matrix. Taking NOTEARS-MLP with one hidden layer of m units as an example, it takes $O(nd^2m + d^2m + d^3)$ time to evaluate the objective and the gradient. If $m/d = O(1)$, this is comparable to the linear case $O(nd^2 + d^3)$, except for the inevitable extra cost from using a nonlinear function. This highlights the benefit of Proposition 1: the acyclicity constraint almost comes for free. Furthermore, we used a quasi-Newton method to reduce the number of calls to evaluate the gradient, which involves computing the matrix exponential. Table 1 contains runtime comparison of different algorithms on ER2 graph with $n = 1000$ samples. Recall that the kernel-based approach of GSGES comes with a $O(n^3)$ computational complexity, whereas NOTEARS-MLP and NOTEARS-Sob has $O(n)$ dependency on n. This can be confirmed from the table, which shows GSGES has a significantly longer runtime.

Comments on hyperparameter tuning The experiments presented in this paper were conducted under a fixed (and therefore suboptimal) value of λ and weight threshold across all graph types, sparsity levels, and SEM types, despite the fact that each configuration may prefer different regularization strengths. Indeed, we observe substantially improved performance by choosing different values of hyperparameters in some settings. As our focus is not on attaining the best possible accuracy in all settings by carefully tuning the hyperparameters, we omit these results in the main text and only include here as a supplement. For instance, for ER4 graph with $d = 40$ variables and $n = 200$ samples, when the SEM is additive GP and MLP, setting $\lambda = 0.03$ and threshold $= 0.5$ gives results summarized in Table 2.

	NOTEARS-MLP	NOTEARS-Sob	FGS	Linear	GNN	GSGES
$d = 20$	92.12 ± 22.51	62.90 ± 16.83	0.55 ± 0.43	10.95 ± 4.52	498.32 ± 43.72	1547.42 ± 109.83
$d = 40$	282.64 ± 67.46	321.88 ± 57.33	0.59 ± 0.17	43.15 ± 12.43	706.35 ± 64.49	6379.98 ± 359.67

Table 1: Runtime (in seconds) of various algorithms on ER2 graph with $n = 1000$ samples.

SEM	Method	SHD	FDR	TPR	FPR	Predicted #
Additive-GP	NOTEARS-MLP	124.3 ± 6.65	0.30 ± 0.07	0.35 ± 0.04	0.04 ± 0.01	81.70 ± 10.49
	GSGES	121.3 ± 5.02	0.36 ± 0.05	0.28 ± 0.03	0.04 ± 0.00	69.30 ± 5.01
MLP	NOTEARS-MLP	88.40 ± 11.29	0.18 ± 0.08	0.57 ± 0.06	0.03 ± 0.02	111.70 ± 15.97
	GSGES	121.60 ± 11.95	0.33 ± 0.09	0.33 ± 0.06	0.04 ± 0.01	77.10 ± 7.13

Table 2: ER4, $d = 40$, $n = 200$ with $\lambda = 0.03$ and threshold $= 0.5$.

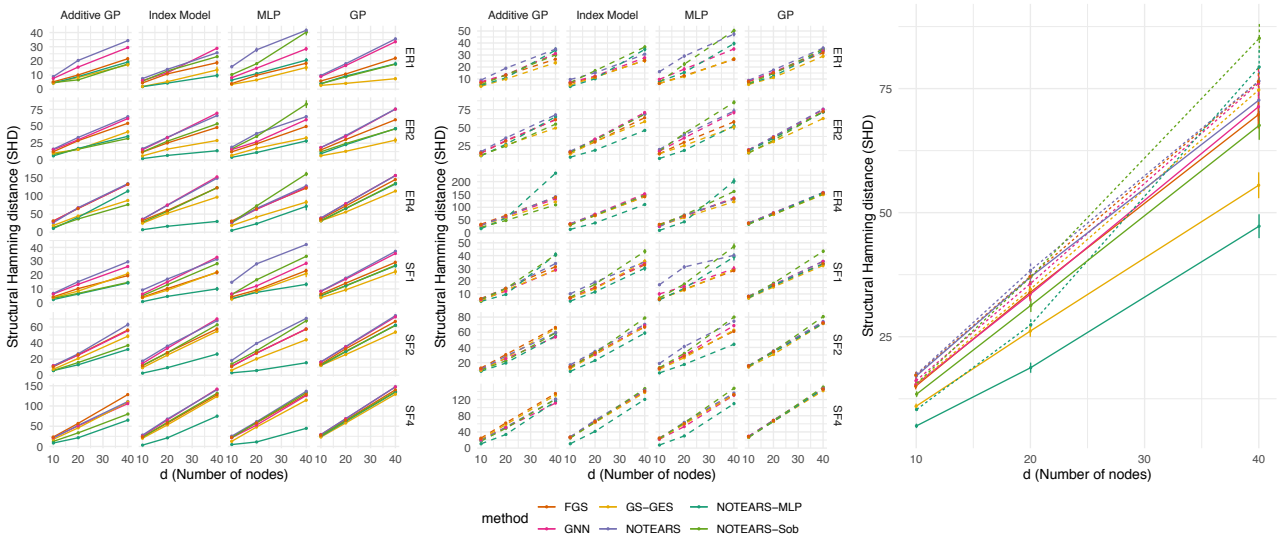

Figure 4: Structure recovery measured by SHD (lower is better) to ground truth.

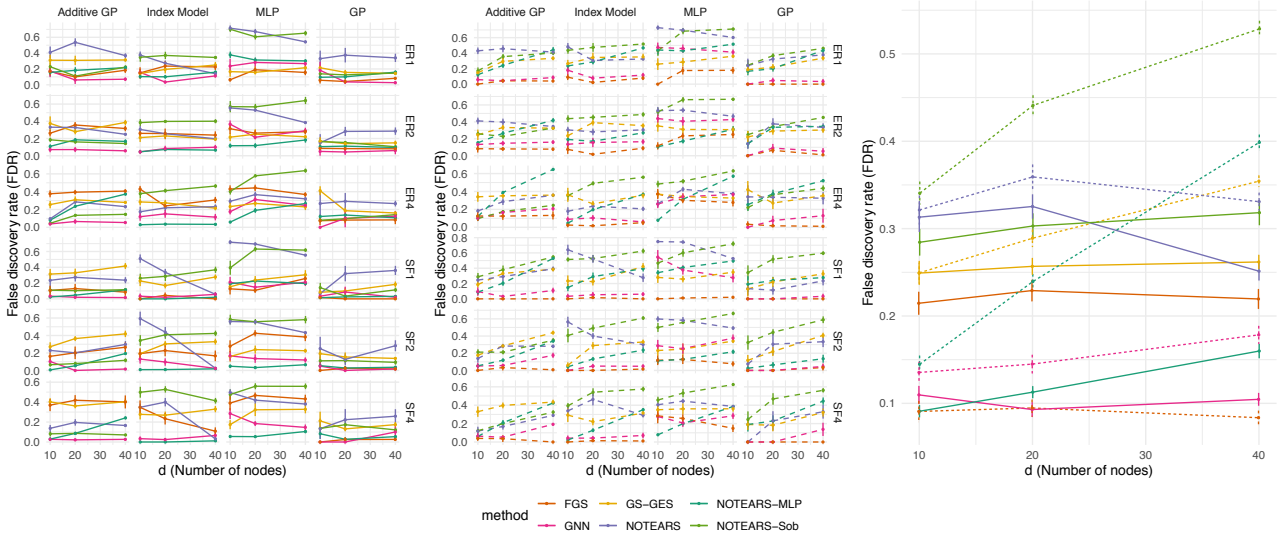

Figure 5: Structure recovery measured by FDR (lower is better) to ground truth.

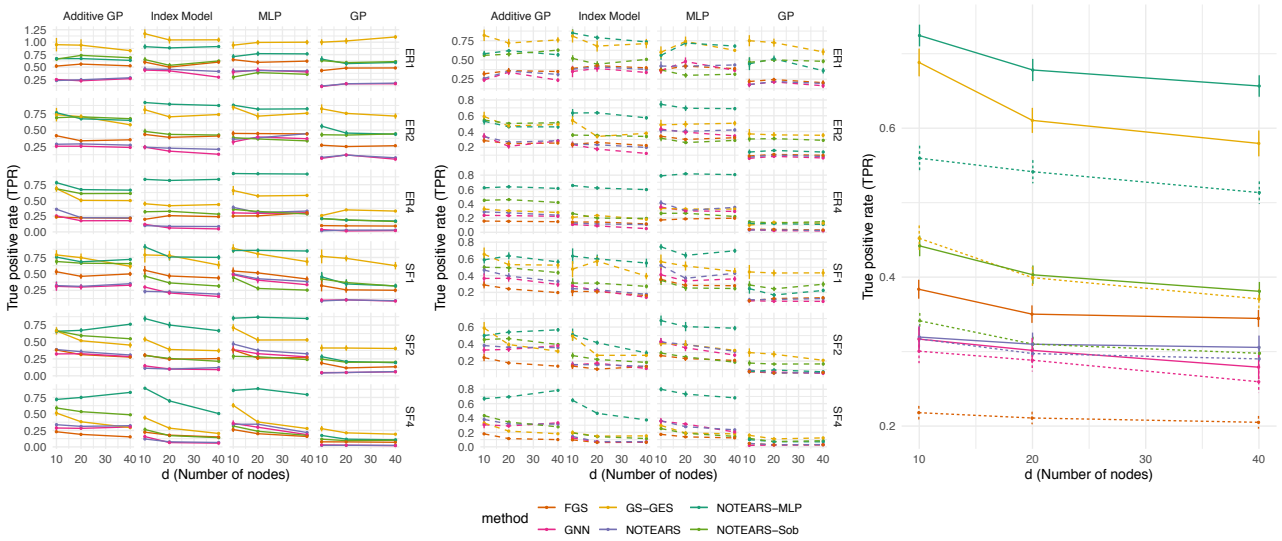

Figure 6: Structure recovery measured by TPR (higher is better) to ground truth.

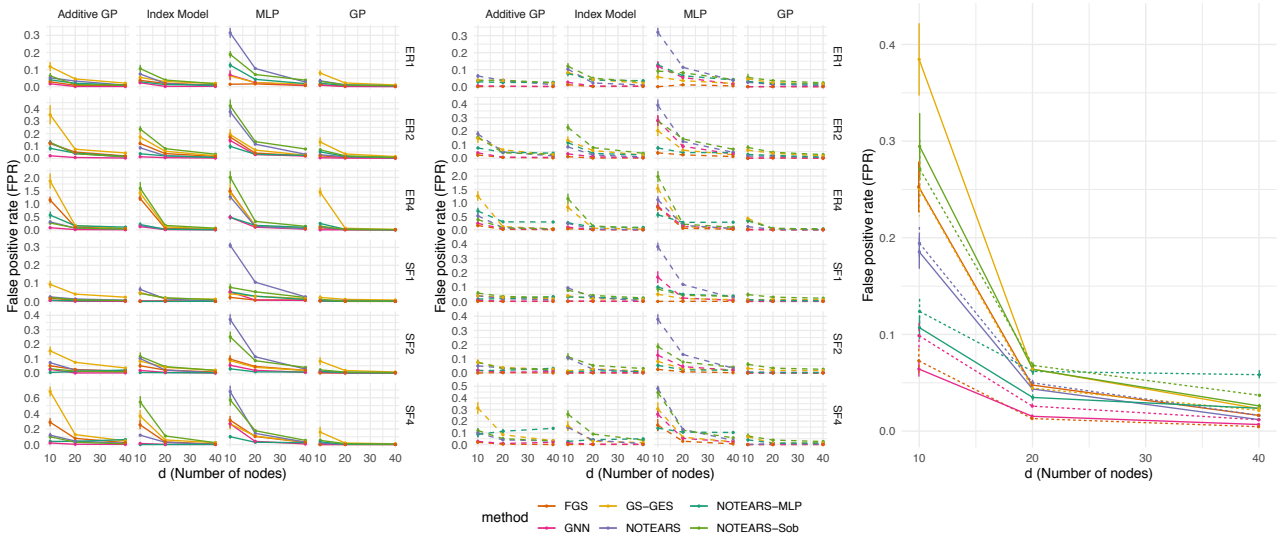

Figure 7: Structure recovery measured by FPR (lower is better) to ground truth.

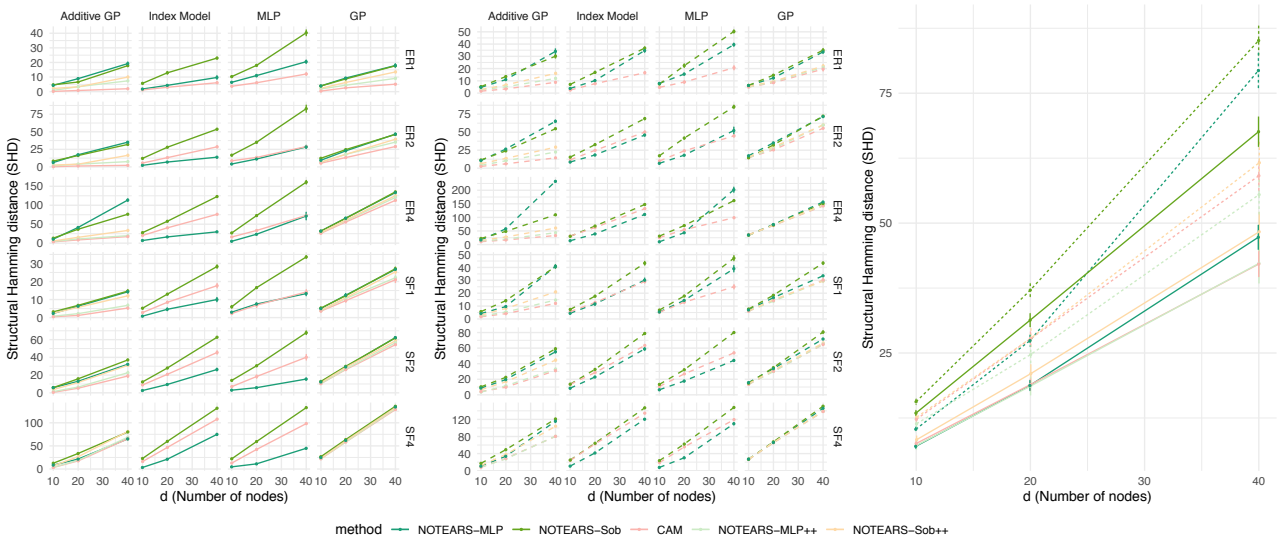

Figure 8: Structure recovery measured by SHD (lower is better) to ground truth, compared with CAM.

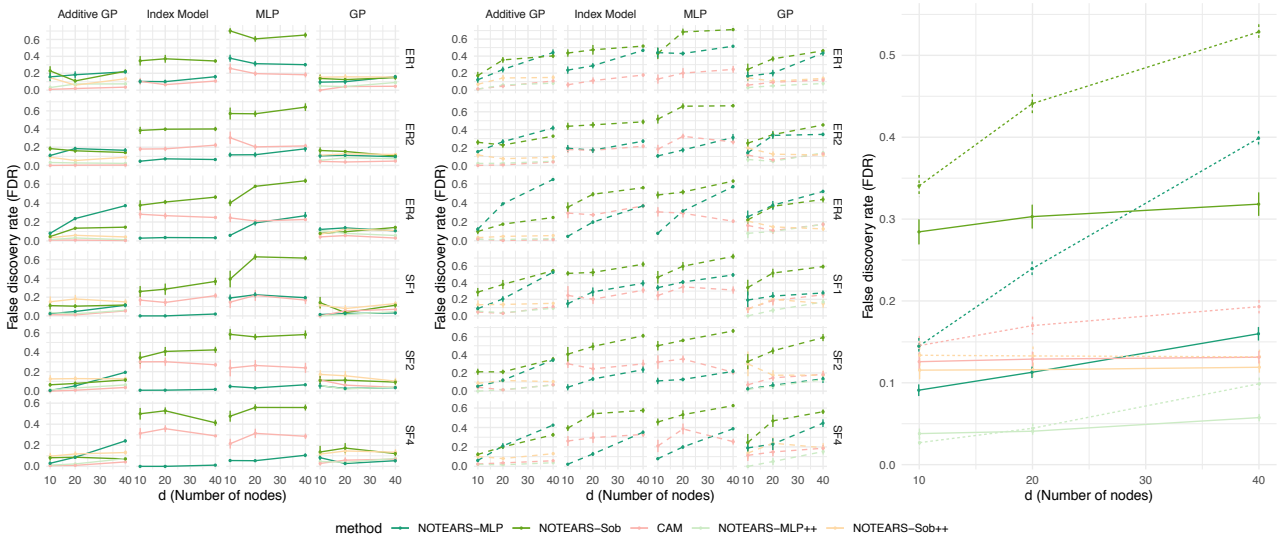

Figure 9: Structure recovery measured by FDR (lower is better) to ground truth, compared with CAM.

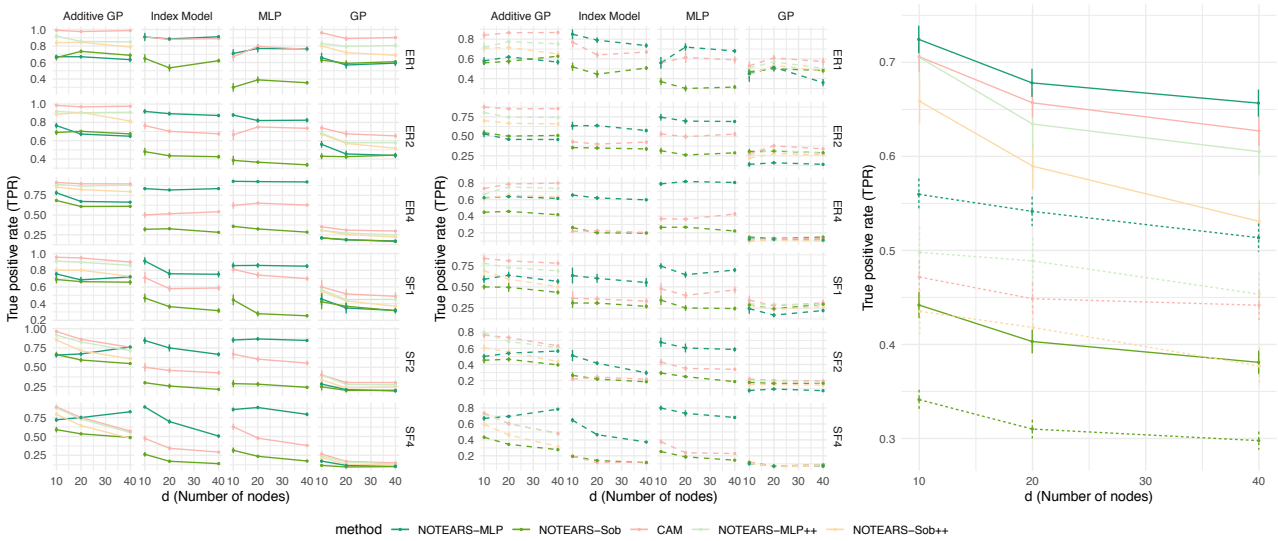

Figure 10: Structure recovery measured by TPR (higher is better) to ground truth, compared with CAM.

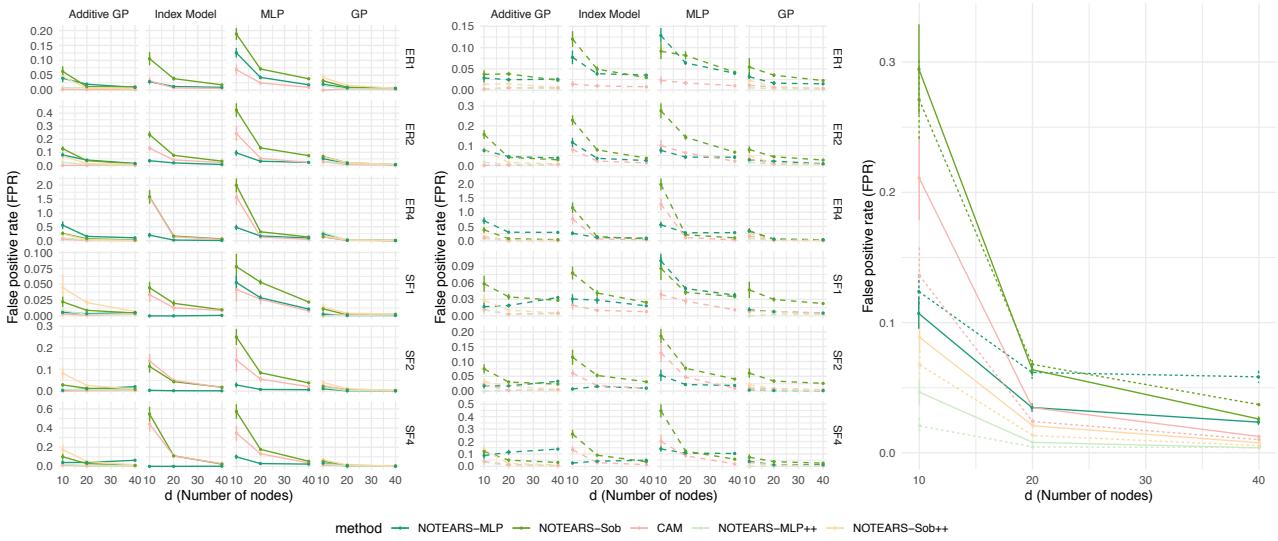

Figure 11: Structure recovery measured by FPR (lower is better) to ground truth, compared with CAM.