Robustness of Algorithms for Causal Structure Learning to Hyperparameter Choice

Damian Machlanski D.MACHLANSKI@ESSEX.AC.UK
Department of Computer Science and Electronic Engineering, University of Essex

Spyridon Samothrakis SSAMOT@ESSEX.AC.UK
Institute for Analytics and Data Science, University of Essex

Paul Clarke PCLARKE@ESSEX.AC.UK
Institute for Social and Economic Research, University of Essex

Editors: Francesco Locatello and Vanessa Didelez

Abstract

Hyperparameters play a critical role in machine learning. Hyperparameter tuning can make the difference between state-of-the-art and poor prediction performance for any algorithm, but it is particularly challenging for structure learning due to its unsupervised nature. As a result, hyperparameter tuning is often neglected in favour of using the default values provided by a particular implementation of an algorithm. While there have been numerous studies on performance evaluation of causal discovery algorithms, how hyperparameters affect individual algorithms, as well as the choice of the best algorithm for a specific problem, has not been studied in depth before. This work addresses this gap by investigating the influence of hyperparameters on causal structure learning tasks. Specifically, we perform an empirical evaluation of hyperparameter selection for some seminal learning algorithms on datasets of varying levels of complexity. We find that, while the choice of algorithm remains crucial to obtaining state-of-the-art performance, hyperparameter selection in ensemble settings strongly influences the choice of algorithm, in that a poor choice of hyperparameters can lead to analysts using algorithms which do not give state-of-the-art performance for their data.

Keywords: Hyperparameters, model selection, causal discovery, structure learning, performance evaluation, misspecification, robustness

1. Introduction

Uncovering causal graphs is an immensely useful tool in data-driven decision-making as it helps understand the underlying data generating process. A large number of causal structure learning algorithms incorporate Machine Learning (ML) methods. These, in turn, heavily rely on hyperparameters (HPs) for accurate predictions (Bergstra et al., 2011). In addition, there has been growing evidence that correctly specified HPs can close the performance gap between State-of-the-Art (SotA) and other methods (Paine et al., 2020; Zhang et al., 2021a; Machlanski et al., 2023; Tönshoff et al., 2023). *Are hyperparameters as important in structure recovery?*

HP optimisation is extremely challenging in structure learning as the true graphs are inaccessible outside of simulated environments. This inability to reliably tune could be one of the reasons behind the struggle to apply some of the algorithms to real data problems (Kaiser and Sipos, 2021), or why HPs are often completely neglected in this area. On the one hand, benchmarks and evaluation frameworks (e.g. Raghu et al. (2018); Tu et al. (2019)) usually focus on finding a learning algorithm that works best under specific circumstances but without considering HPs as part of the problem. On the other hand, studies that address HP tuning (e.g. Strobl (2021); Biza et al. (2022)) consider individual algorithms but not the impact of tuning (or the lack of it) on selecting the best algorithm for the available data. Understanding how HPs affect algorithm choice, as well as individual methods, is clearly missing but can be a crucial next step towards more stable causal discovery in real data applications. To make matters worse, the evaluation metrics used for tuning can be imperfect and sometimes favour specific learning methods (Curth and van der Schaar, 2023). This brings us to the core questions of this paper: *Do different algorithms perform similarly given access to a hyperparameter oracle? How robust are they against misspecified hyperparameters?*

In this work, we set out to address these questions and investigate the impact HPs have on graph recovery performance of individual algorithms, as well as on the best algorithm choice (see Figure 1). We start by showing how a single HP plays a crucial role in the simplest graph problem (two variables). More extensive experiments strengthen this observation and confirm it as a more general phenomenon. The experimental setup involves many seminal structure learning algorithms tested against real and simulated datasets.

Contributions. a) Compare algorithms' performances and their winning percentages across hyperparameters, b) Compare algorithms' performances under well-specified and misspecified hyperparameters, and c) Compare algorithms' winning percentages under well-specified and misspecified hyperparameters.

Related work. This work connects with the existing literature mainly through the topics of performance evaluation and HP analysis. The performance of structure learning algorithms has been evaluated from a number of different perspectives, such as mixed data types (Raghu et al., 2018) or time series data (Assaad et al., 2022). In an attempt to strengthen the evaluation, there have been efforts to develop testing environments that closely resemble real-life datasets. Some examples include simulators based on gene regulatory networks (Van den Bulcke et al., 2006) or neuropathic pain pathology (Tu et al., 2019). Grünbaum et al. (2023) take evaluation further by proposing to test algorithms on the parts of real-life datasets that are known a priori. Furthermore, to improve reproducibility, Rios et al. (2021) developed a benchmarking platform that covers a wide range of learning methods and data scenarios. The importance of HPs and their impact on performance have been mostly studied in other areas outside of structure learning. In the offline Reinforcement Learning (RL) setting, Paine et al. (2020) reported, among other aspects, that robustness to HP choices is an important issue and that careful tuning can deliver close to optimal policies. Zhang et al. (2021a), on the other hand, make a case for HP tuning in model-based RL. Similar observations have been reported in causal effect estimation (Machlanski et al., 2023), graph learning (Tönshoff et al., 2023), code classification (Shen et al., 2020) and evolutionary algorithms (Eiben and Smit, 2011), where SotA performance levels are attributed to HPs alone. HPs have been also studied in the broader context of "tunability" (Probst et al., 2019) and optimisation approaches (Yu and Zhu, 2020). Some notable recent works even challenge our current understanding of how HPs interact with loss functions (Huang and Li, 2023) and decision boundaries (Sohl-Dickstein, 2024). HPs in structure learning have mostly been discussed in the context of tuning. One common approach is to select HPs that result in stable structure predictions across random data samples (Liu et al., 2010; Sun et al., 2013; Strobl, 2021). Another strand of work performs out-of-sample validation for tuning purposes based on predictive accuracy of models fitted in accordance with the recovered graph structure (Biza et al., 2022), or assigned scores developed specifically for structure recovery tuning (Chobtham and Constantinou, 2023). Metrics based on regression error have been also considered (Marx and Vreeken, 2019), though in the context of two variables.

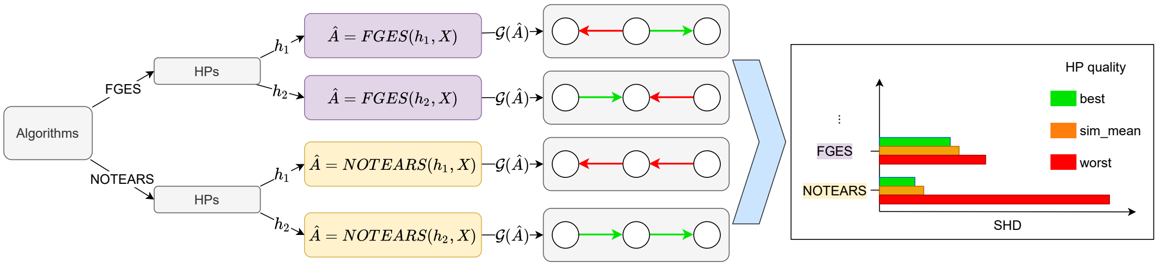


Figure 1: Summary of the main idea of the paper. We explore various structure learning algorithms and investigate how hyperparameters affect their performance. Notation: h_1 and h_2 are different hyperparameter values; X denotes i.i.d. data provided to algorithms; $\hat{A}$ is recovered adjacency matrix while $\mathcal{G}(\hat{A})$ is a causal graph based on $\hat{A}$; SHD is structural Hamming distance (lower is better). Note how recovered graphs differ between different hyperparameters of the same algorithm (green edges are correct; red incorrect). *sim_mean* are hyperparameters that achieved the best **average** performance across all simulations.

Structure. In Section 2 we briefly discuss the basics of structure learning. Section 3 demonstrates the importance of hyperparameters in structure learning via a bivariate example, further motivating more extensive numerical experiments presented in Section 4. Section 5 concludes the paper and offers potential future work directions.

2. Structure Learning

We briefly describe the notation and data assumptions used throughout the paper as well as important details of structure learning methods necessary to read the technical parts of the document. For a more detailed review, see recent literature (e.g. Eberhardt (2017); Glymour et al. (2019)).

Graphs. Let $\mathcal{G} = (V, \mathcal{E})$ be a graph with nodes/vertices $V = \{1, \ldots, p\}$ and edges $\mathcal{E} \subseteq V^2$. Edges are pairs of nodes $(j, k) \in \mathbf{V}$ where $(v, v) \notin \mathcal{E}$ to exclude self-cycles. Nodes j, k are **adjacent** in $\mathcal{G}$ if either $(j, k) \in \mathcal{E}$ or $(k, j) \in \mathcal{E}$. An edge is **undirected** if $(j, k) \in \mathcal{E}$ and $(k, j) \in \mathcal{E}$, whereas it is **directed** if only one pair appears in $\mathcal{E}$; if this pair is (j, k) then j is called a **child** of **parent** k. The set of parents of j in $\mathcal{G}$ is denoted by $\mathbf{PA}_j^{\mathcal{G}}$. We call $\mathcal{G}$ directed if all its edges are directed. A **mixed** graph consists of both directed and undirected edges. The **skeleton** of any directed or mixed graph $\mathcal{G}$ is an equivalent graph with all directed edges replaced by undirected ones. A **fully connected** graph $\mathcal{G}$ is one where all pairs of nodes are adjacent. A (directed) **path** is a sequence of nodes connected by (directed) edges. A **partially directed acyclic graph** (PDAG) is a mixed graph such that there is no pair (j, k) such that there are directed paths from j to k and vice versa. Then, $\mathcal{G}$ is a **directed acyclic graph** (DAG) if it is a PDAG and is directed. Two graphs are *Markov equivalent*, or belong to the same *equivalence class*, when they involve the same sets of *d-separations* (Pearl, 2000). A **completed PDAG** (CPDAG) can encode such a class of graphs, in which undirected edges mean that the graphs within the class may contain a directed edge in either direction; directed edges denote agreement in edge direction in subsumed graphs.

Assumptions. Now consider a vector of random variables $\mathbf{X} = (X_1, \ldots, X_p)$ generated according to an unknown data generating process (DGP) leading to joint distribution $\mathcal{L}(\mathbf{X})$. The node $j \in \mathbf{V}$ represents random variable X_j and the edge between nodes j and k in $\mathcal{E}$ is directed if and only if X_k is used in the DGP to generate X_j. We further assume that: a) there are no hidden confounders (*sufficiency*); b) two variables are independent in $\mathcal{L}(\mathbf{X})$ if they are *d-separated* in $\mathcal{G}$

(*Markov condition*); and c) two variables are *d-separated* in $\mathcal{G}$ if they are independent in $\mathcal{L}(\mathbf{X})$ (*faithfulness*).

Learning Methods. The goal of causal structure learning is to infer (or identify) graph $\mathcal{G}$ given the distribution $\mathcal{L}(\mathbf{X})$. If it is possible to do this, we say $\mathcal{G}$ is **identifiable** from $\mathcal{L}$. Traditional methods, such as PC (Spirtes and Glymour, 1991) or GES (Chickering, 2002), were often built around assumptions a-c) above, which are in most cases not enough to identify a unique DAG solution, only the class of CPDAGs. If one, however, assumes the DGP is a Structural Causal Model (SCM) then identification is possible, for example, if the DAG is a linear SCM with additive non-Gaussian noise: $X_j = f_j(X_{\mathbf{PA}_j^{\mathcal{G}}}) + \epsilon_j$ (Shimizu et al., 2006). Subsequent research has extended this result to nonlinear SCMs with additive noise (Hoyer et al., 2008), linear models with Gaussian noise terms of equal variances (Peters and Bühlmann, 2014), and additive models of the form $X_j = \sum_{k \in \mathbf{PA}_j^{\mathcal{G}}} f(X_k) + \epsilon_j$ (Bühlmann et al., 2014). More recent approaches also involve neural network-based algorithms that specifically restate the learning task as a continuous optimisation problem (Zheng et al., 2018, 2020).

3. Hyperparameters in Structure Learning

3.1. Bivariate Example

An illustrative example, strongly inspired by Marx and Vreeken (2019), is the classic *cause-effect pairs* challenge (Guyon et al., 2019) that consists of two (synthetically generated here) variables X and Y, with the goal of establishing the existence and direction of the causal link between them ($X \rightarrow Y$, $X \leftarrow Y$, no link) given only observed data. One possible solution is to fit two regressors $y = f(x)$ and $x = g(y)$, and predict the causal direction based on the lower prediction error of the two models ($\epsilon_f = [y - \hat{f}(x)]^2$ and $\epsilon_g = [x - \hat{g}(y)]^2$; no link if the errors are comparable). As shown in Figure 2, changing the HP that controls the number of regression parameters can result in a different causal direction being predicted. This is precisely what constitutes the problem, since the true DGP and the correct HP value are unknown.

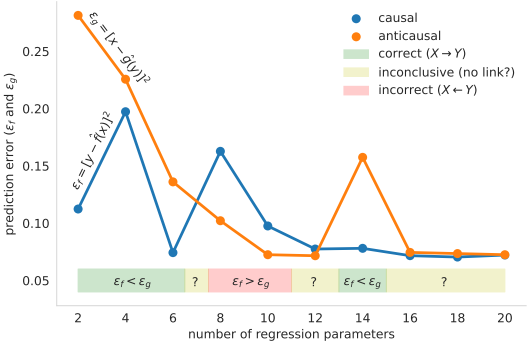


Figure 2: The number of allowed regression *parameters*, a **hyperparameter**, clearly affects prediction error of the two models and can determine predicted causal direction (see decision colours at the bottom). The algorithm predicts $X \rightarrow Y$ if $\epsilon_f < \epsilon_g$, $X \leftarrow Y$ if $\epsilon_f > \epsilon_g$, and is inconclusive otherwise. Note that the true causal direction is unknown in practice.

Observation 1 *Incorrect hyperparameters can cause prediction mistakes.*

Observation 2 *There might be more than one correct and incorrect hyperparameter choice.*

The problem grows in complexity as the number of graph nodes and edges increases. This is because each edge is a potentially different function to approximate that will require a different HP value (function complexity) to obtain the correct answer. In addition, many algorithms provide multiple HPs to tune, making even more room for further mistakes, effectively increasing the chance of HP misspecification. In fact, even the bivariate example can involve more HPs by, for instance, introducing a threshold such that the algorithm predicts ‘no link’ if $|\epsilon_f - \epsilon_g| < threshold$. To this end, we make the following claim and set up the following key definition:

Claim 1 *The existence and direction of an edge in the predicted graph strongly depends on algorithm’s hyperparameters.*

Definition 1 (Hyperparameter Misspecification) *Mistakes in predicted graph structure arising from incorrect hyperparameters.*

Note that in practice HP misspecification may be not the only source of prediction mistakes as not all learning algorithms are guaranteed to converge to optimal solutions.

3.2. General Form of the Problem

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ represent i.i.d. data of n observations and p features. Furthermore, let $A \in \{0,1\}^{p \times p}$ be a binary **adjacency matrix** of directed graph $\mathcal{G}(A)$ such that $a_{jk} = 1$ if $(j,k) \in \mathcal{E}$ and $a_{jk} = 0$ otherwise. A **weighted** adjacency matrix $W \in \mathbb{R}^{p \times p}$ is defined such that $A(W)_{jk} = 1$ if $w_{jk} \neq 0$ and zero otherwise, which results in a weighted graph $\mathcal{G}(W)$. Let us also define distance $d(A,B)$ between two adjacency matrices A and B such that $d(A,B) = 0$ if and only if $\mathcal{G}(A) = \mathcal{G}(B)$ are the same graph. From now on, let us denote A as the true adjacency matrix and $\hat{A}$ its estimate obtained by a programme P from i.i.d. data X using programme options O so that

$$\hat{A} = P(O, X), \tag{1}$$

where O generally involves the user specifying algorithm S and hyperparameter values H. Therefore, given K user-specified candidate programmes $P = \{P_1, \ldots, P_K\}$ and

$$\hat{A}_k = P(S_k, H_k, X), \tag{2}$$

where S_k and H_k are the algorithm and HP choices associated with program candidate k, then the best program is

$$k^* = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} \; d(A, \hat{A}_k), \tag{3}$$

Note that k^* is generally not identifiable unless A is known. Furthermore, identification of $\mathcal{G}$ does not guarantee $\hat{A}_{k^*} = A$ as $\hat{A}_{k^*}$ depends on algorithms S and their ability to identify A, as well as the choice of their HPs H.

More generally, when considering different algorithms S and HPs H, Equation 3 is the standard **model selection** problem, whereas if the choice of algorithms is fixed to a specific value, leaving HPs as the only variable, the task reduces to **hyperparameter tuning**. In practical terms, obtaining the distance $d(A, \hat{A})$ is not feasible, as the true matrix A and its corresponding graph $\mathcal{G}(A)$ are inaccessible outside of simulated environments. As algorithms can vary substantially in design, the appropriate way to compare them requires the use of distance measures that incorporate the ground truth A. This renders model selection impractical in structure learning problems. Tuning HPs of a single algorithm might be feasible by comparing its relative scores across explored HP values.

3.3. Common Hyperparameters

Despite differences in algorithms, many of them share similar HPs. Commonly used ones are briefly described here.

Significance level of independence tests. Refers to the *p-value* of independence tests and the desired level of certainty. Decreasing the value (increasing certainty) will usually result in fewer predicted edges. Often named as *alpha* and incorporated by traditional and pairwise algorithms.

Sparsity. A penalty term that encourages sparser solutions. Higher values result in fewer predicted edges. Similar in mechanism to *L1 regularisation* which discards less relevant features. Often employed by regression-based solutions, especially if they perform some form of feature selection.

Model complexity. A penalty that encourages simpler models to avoid overfitting (*L2 regularisation*). As shown in the example in Section 3.1, its influence on the final prediction is complicated. Usually applies to solutions that model the assumed form of SEMs.

Post-prunning threshold. Many SEM-based methods output weighted adjacency matrices W that need to be converted to the binary form of A. This is usually done by applying a threshold below which all edges are set to zero. That is, $a_{ij} = 1$ if $w_{ij} > w\_thresh$; 0 otherwise..

Note that *alpha* and *w_threshold* are algorithm agnostic and can be transferred between methods, whereas the other two HPs may differ in value between algorithms.

4. Experiments

Since our analysis in Section 3 is based merely on a simple and artificial example, our next step is to study the influence of HPs more rigorously in a more general setting. We devise a set of experiments consisting of diverse data sets of various size and difficulty (Section 4.1), processed by a representative set of structure learning algorithms (Section 4.2). Different HP selection scenarios are also detailed (Section 4.4).

The experimental framework is implemented through Benchpress (Rios et al., 2021), a benchmarking platform to evaluate structure learning algorithms. Performances of all algorithms are collected from Benchpress, followed by some mild post-processing of the results to suit our analysis of HPs. The code and data are available via Appendix A.5.

4.1. Graphs and Data

4.1.1. SIMULATIONS

We follow recent literature in structure learning when it comes to simulating different DGPs. The simulation procedure starts with generating a random DAG $\mathcal{G}$ with p nodes and d edges, built according to a random graph model. The resulting graph is binary, with $A \in \{0, 1\}^{p \times p}$. Next, i.i.d. data X $\in \mathbb{R}^{n \times p}$ are sampled from a simulated SEM of choice, with n being the sample size. Each individual combination of settings is repeated for 10 seeds and forms a single experiment. In our experiments, we explore $p \in \{10, 20, 50\}$, $d \in \{1p, 4p\}$, and $n \in \{200, 1000\}$. Included random graph models are Erdös-Rényi (ER) (Erdös and Rényi, 1959) and Barabási-Albert (Barabási and Albert, 1999), with the latter also known as scale-free (SF). We also explore $n = 10,000$ but only for sparse ER graphs with $p = 50$ nodes due to computational limitations. As for explored SEMs, we include the following:

Linear Non-Gaussian (gumbel). $X = XW^T + z \in \mathbb{R}^p$, with $W \in \mathbb{R}^{p \times p}$ as edge weights assigned independently from $U([-2, -0.5] \cup [0.5, 2])$ and based on A. Noise z follows the Gumbel distribution $z \sim$ Gumbel$(0, I_{p \times p})$.

Nonlinear Gaussian (gp). $X_j = f_j(X_{\mathbf{PA}_j^{\mathcal{G}}}) + z_j$ for all $j \in [p]$ in the topological order of $\mathcal{G}$. Noise z_j follows Gaussian distribution $z_j \sim \mathcal{N}(0, 1)$, $j = 1, \ldots, p$. Where functions f_j represent a draw from a Gaussian process with a unit bandwidth RBF kernel.

Note that both settings have been shown to be identifiable. That is, linear non-Gaussian additive models (Shimizu et al., 2006) and nonlinear additive models (Hoyer et al., 2008).

4.1.2. REAL DATASETS

We also tested structure learning algorithms against real or semi-real datasets. The most popular ones in the literature are *protein signaling* and *SynTReN*.
Protein signaling comes from Sachs et al. (2005) which measures protein and phospholipid expression levels in human cells. The ground truth causal graph has been established and accepted by the experts in the field. We use the second dataset that is already logged and standardised and consists of $n = 902$ observations, $p = 11$ nodes and $d = 17$ edges.
SynTReN is a generator of synthetic transcriptional regulatory networks and related gene expression data that simulate a real experiment (Van den Bulcke et al., 2006)[1]. We use the same data as in Lachapelle et al. (2019), which consist of 10 random seeds, $n = 500$ samples and $p = 20$ nodes.

4.2. Structure Learning Algorithms

We consider in our setup the following algorithms. PC (Spirtes and Glymour, 1991), FCI (Spirtes et al., 1993), FGES (Ramsey et al., 2017), LiNGAM (Shimizu et al., 2006), ANM (Hoyer et al., 2008), CAM (Bühlmann et al., 2014), NOTEARS (Zheng et al., 2018) and NOTEARS_MLP (Zheng et al., 2020). Due to high computational demands, we only focus on well-established and seminal algorithms that, in our view, effectively represent different classes of solutions. More details about the algorithms and HPs involved can be found in Appendix A.2. Note the algorithms may have different termination criteria due to design differences, but they all produce (CP)DAGs.

4.3. Evaluation

We compare algorithms' performances across three commonly used metrics: *structural Hamming distance* (SHD), *false positives* (FPs), and *false negatives* (FNs). All three accommodate for the fact that the ground truth is always a DAG but some of the incorporated algorithms output CPDAGs. The implementation of SHD we use counts not only the number of edge additions, removals and reversals, but also edge orientations, needed to turn the predicted graph into the true DAG. FPs and FNs count false edges and are calculated based on graph skeletons. For this purpose, predicted and true graphs are converted to skeletons to obtain the two metrics. This allows us to include methods that output CPDAGs even though the primary focus of this study is DAG recovery. See Appendix A.4 for more detailed definitions of the metrics.

4.4. Hyperparameters

All incorporated learning algorithms have at least one HP. We collect performances of algorithms across all HP combinations (exhaustive grid search; see Appendix A.1). Note that while many HPs are defined on continuous spaces, we search over a pre-defined set of points chosen to cover the

[1] http://bioinformatics.intec.ugent.be/kmarchal/SynTReN/index.html

space as completely as possible. To better understand the influence of HPs on structure recovery performance, we experiment with four different HP selection strategies described below.

BEST. To simulate the choice of the best HPs (as if we had access to the HP oracle), we pick HP values that achieved **the lowest** metric value in that particular data setting. Each data setting can have a different set of the best HPs.

WORST. Identified similarly to 'best' except the criterion here is **the highest** metric value.

DEFAULT. Default HP values recommended by the authors of an algorithm. See Appendix A.2.

SIM_MEAN. An alternative to 'default'. We found a single set of HP values per algorithm that achieved **the lowest average** metric value across all simulations. These are *simulation-derived* default values. See Appendix A.1 for identified values.

4.5. Results

Presented results employ the following naming convention with respect to the DGP: *graph_p* (number of nodes), *graph_d* (edge density), *graph_type* (graph models; ER or SF), *data_n* (sample size), *data_sem* (SEM type; gumbel or gp). Error bars are standard errors unless stated otherwise. Due to space limitation only the most important results are presented in the main content of the paper. The rest of the results, that do not change conclusions, are in Appendix B.

Performance Distribution Across Hyperparameters. As per Figure 3, all algorithms perform similarly when averaged across all simulations and hyperparameters. This confirms that no single algorithm is the best in all conditions; they rather specialise in solving specific challenges that are ingrained in their design via assumptions. Some minor deviations from this general observation can be noticed when considering FPs (false positives) only (see FCI, PC and ANM), but they become negligible when considering the main metric (SHD).

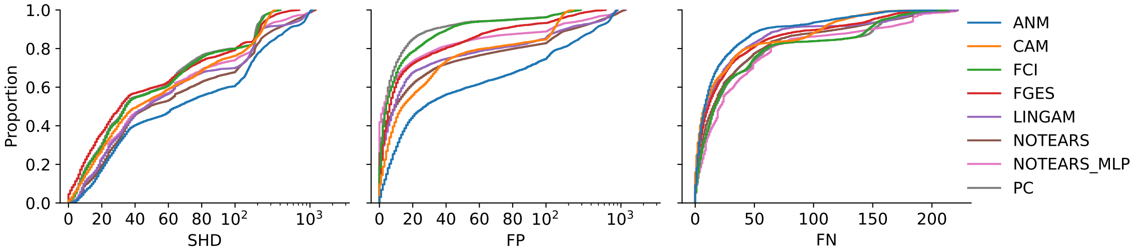


Figure 3: Proportions of performances (lower is better) across all HP values and simulations. Interpretation: algorithms perform similarly when averaged over all DGPs and HPs; no algorithm is the best in all conditions.

Performance vs. Hyperparameter Quality. As shown in Figure 4, achieved performance is clearly affected by both algorithm and HP choices. Even assuming access to HP oracle (blue colour), selecting different algorithms will have a significant impact on the result (blue bars differ among algorithms). This is because assumptions, either implicit or explicit, about the DGP play a critical role in controlling the performance of each method. It is worth noting that, in contrast to previous studies of HP choice for other causal methods, the absolute performance level achieved by the algorithms is low: no algorithm comes close to achieving SHD $= 0$. The complexity of the learning task is such that accurate structure discovery currently lies beyond the state of the art. In this context, fixed HPs (orange and green) seem to be a viable strategy as they are relatively close to the

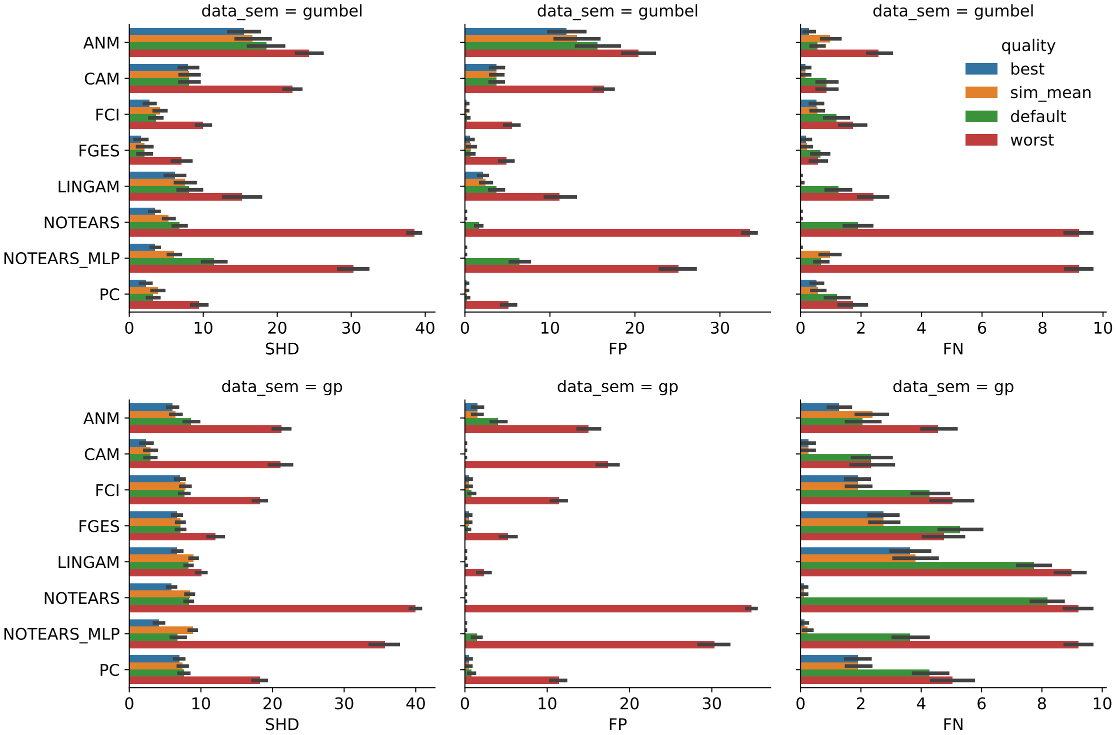


Figure 4: Performances (lower is better) for small sparse graphs ($p = 10$, $d = 1$) with linear (*gumbel*) and nonlinear (*gp*) SEMs depending on the **quality** of selected HPs (see the legend). Interpretation: a) fixed HPs (orange and green) perform similarly as the optimal ones (blue), b) algorithm selection is important even with optimal HPs (blue bars differ among algorithms), c) certain algorithms and setups are more risky (differences among red bars), and d) methods with optimised HPs provide only true edges (see blue bars in FPs).

best cases (blue). This shows that HP values transfer well between different DGPs, which can be exploited in practice as a countermeasure to challenging HP optimisation. The differences between simulation-derived and paper-default values (orange and green respectively) are negligible in most cases with the exception of FNs (false negatives) where the former perform better. This indicates that the recommended defaults often prioritise sparsity, which reduces FPs, at the cost of increased FNs. The worst HPs (red), on the other hand, can result in performances substantially worse than the fixed ones. This shows the risks of HP misspecification, the degree of which clearly varies across algorithms (different robustness), which could be due to again different data assumptions in algorithms or varied degrees of freedom via algorithm's HPs. Note also how the majority of mistakes under misspecified HPs (red) is due to FPs (red bars in FPs larger than in FNs). However, once HPs are optimised (blue), FPs are mostly eliminated and the remaining mistakes are now due to FNs (blue bars larger in FNs than FPs). This has critical implications for practice: the predicted edges can be trusted (FP small) but the absence of an edge cannot (FN large). Remaining FNs could be also a sign of too aggressive sparsity/prunning or simply failed identification, with the latter being an indicator for future algorithmic improvements.

Performance vs. Cardinality of Hyperparameters. A speciously interesting comparison that emerges from our study is the relationship between the cardinality of the HPs we explore and al-

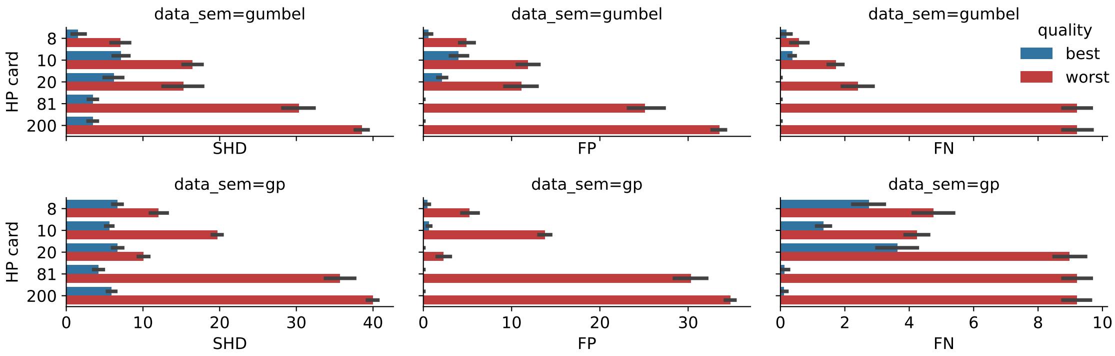


Figure 5: The influence of explored HP cardinalities (*HP card*) across algorithms on performance (lower is better), depending on the **quality** of selected HPs (see legend). DGP: small sparse graphs ($p = 10$, $d = 1$) with linear (*gumbel*) and nonlinear (*gp*) SEMs. Note how higher cardinalities lead to small SHD gains under optimal HPs (blue), but at the cost of much worse performances under misspecified HPs (red). Warning: different cardinalities may involve different algorithms (possible coincidental trends).

gorithm performance. Our study was not set up to explore this issue so we now clarify what can and cannot be concluded about this relationship from our study. As presented in Figure 5, higher cardinalities (81 and 200) lead to only small performance improvements under optimised HPs (blue bars), but they involve significantly higher risks of poor performances if HPs are misspecified (red). This shows that larger HP search spaces should be explored with caution as they provide little gain for a much higher risk of poor performances. Note, however, that different cardinalities presented here may involve different algorithms, making room for coincidental trends. For example, it is unclear given the data if robustness to misspecified HPs is due to HP cardinalities or algorithms as the two highest cardinalities were explored exclusively with (potentially volatile) neural networks.

Winning Algorithms vs. Hyperparameter Quality. Previous analysis revealed that the best algorithm choice may depend on specific DGP properties as well as HP choices. To make it clearer, we collect winning algorithms across different DGP properties and HP selection strategies. An algorithm with the lowest SHD in a given setting wins. Table 1 presents top performers across different

Table 1: Top algorithm choices based on the highest winning percentages, grouped by DGP properties (graph_p, data_sem, graph_d) and quality of selected HPs (best, sim_mean, worst). If there is no clear winner, multiple top performers are reported. Notice how the winners change across DGPs (columns), but also across HPs (rows), showing algorithm selection is nuanced. N_MLP denotes NOTEARS_MLP.

graph_p	10				20				50			
data_sem	gumbel		gp		gumbel		gp		gumbel		gp	
graph_d	1	4	1	4	1	4	1	4	1	4	1	4
HP best	FGES FCI	FGES	CAM	CAM	FGES	PC N_MLP	CAM	CAM	FGES	PC N_MLP	CAM	CAM
HP sim_mean	FGES PC	FGES CAM ANM	CAM	CAM	FGES	PC	CAM	CAM	FGES	PC	CAM	CAM
HP worst	FGES	FGES PC CAM	LiNGAM FGES	CAM	FGES	PC	LiNGAM	CAM LiNGAM	FGES	PC	LiNGAM	LiNGAM

settings based on accumulated winning percentages. The results confirm that no single algorithm wins in all settings. Specific DGP properties may favour certain methods. When looking at how top performers change depending on different HP choices, it is clear that the best algorithm selection depends not only on DGP properties, but also on the type of available HPs. For instance, to minimise the risk of poor performances, one can select algorithms from the 'HP worst' category.

Performance vs. Sample Size. Figure 6 confirms that increased sample size generally helps, even with relatively large graphs ($p = 50$), though some algorithms need more data to notice major benefits (see LiNGAM in gumbel and NOTEARS_MLP in gp). Positive effects can be noticed with respect to improved best HP cases (min values) and an increased proportion of good performances (mean values). This case also confirms that relatively large and sparse graphs can be recovered with high accuracy given the right HPs (min values of some green boxes are close to 0).

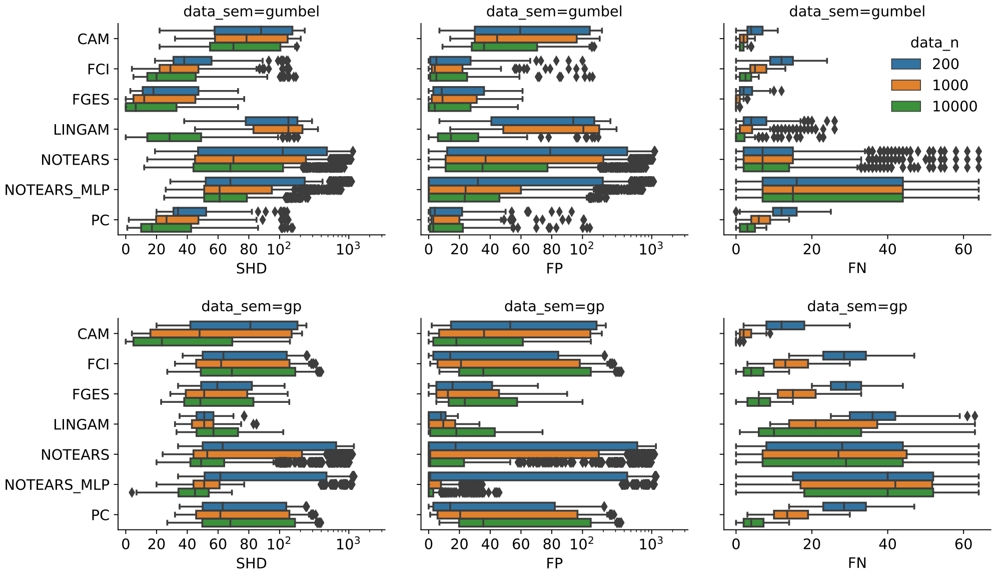
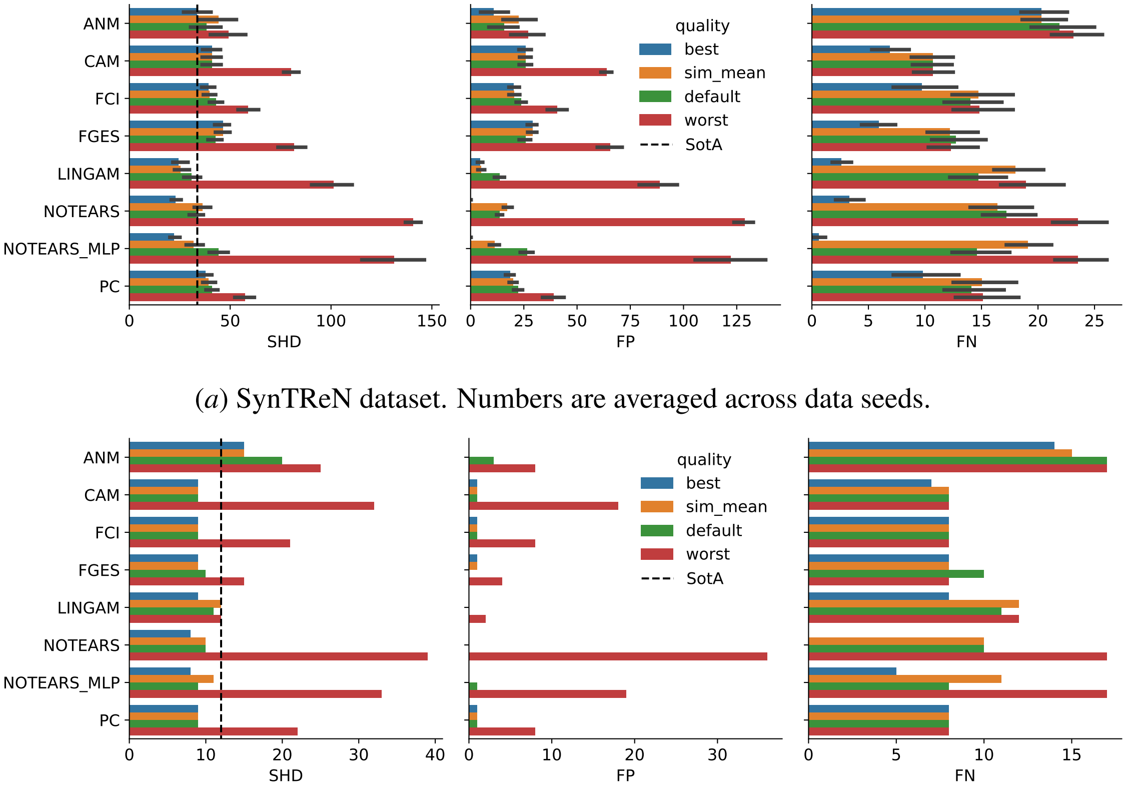


Figure 6: The influence of sample size (*data_n*; colours) on performances (lower is better) across all HPs. DGP: sparse ($d = 1$) large ($p = 50$) ER graphs with linear (*gumbel*) and nonlinear (*gp*) SEMs. ANM was excluded due to long execution time against $10,000$ samples. Note how increased sample size not only improves performances under the best and worst HPs (min and max decrease), but also the proportion of good performances (means decrease as well), suggesting increased sample size helps with HP misspecification.

Semi-Synthetic and Real Data. We put our simulation-derived findings to a test by performing structure recovery on SynTReN and Sachs datasets (Figure 7). SHD numbers are compared to SotA performances retrieved from Lachapelle et al. (2019), which are 33.7 ± 3.7 and 12 SHD for SynTReN and Sachs respectively. Both cases generally confirm that fixed HPs (*sim_mean* and *default*) can work almost as well as the best HPs, and that even the best HPs may not be enough to reach the best possible performance as some algorithms perform better than others under those conditions. It is also clear from both cases that HPs play an important role and, in fact, can decide whether an algorithm reaches or beats SotA. For instance, against SynTReN, both NOTEARS methods and LiNGAM seem to be good options under the best and fixed HPs. But under the worst HPs, NOTEARS methods can be extremely inaccurate, making ANM the safest choice in this case. In the Sachs dataset, this is no longer the case with ANM, showing that the best algorithm pick indeed



(*a*) SynTReN dataset. Numbers are averaged across data seeds.

(*b*) Sachs dataset

Figure 7: Performances (lower is better) against SynTReN (top) and Sachs (bottom) datasets depending on the **quality** of selected HPs (colours). Notice that: a) HP values derived from simulations perform well here (orange), and b) the quality of HPs and algorithm choice are both important for beating SotA.

strongly depends on DGP properties. All algorithms except ANM can, in fact, beat SotA on Sachs data. However, when it comes to robustness to HP misspecification and safety of use, NOTEARS methods appear to be the most risky, with LiNGAM being extraordinarily robust as it beats SotA even with its worst HPs.

5. Conclusion

In this work, we have successfully shown that HPs play an important role in causal structure learning. However, the way HPs influence the methods is somewhat different than recent results from the ML literature. More specifically, Machlanski et al. (2023); Tönshoff et al. (2023) found that many learners can reach SotA performance levels with the right HPs, reducing the importance of model selection. But in this study, we observe that algorithms still differ significantly in performance even with access to the HP oracle. However, reliable tuning is not always available in structure learning, leading to HP misspecification and prediction errors. This is where HPs become important as we showed that different learning algorithms vary in robustness to HP misspecification, and that strong performance under the right HPs does not necessarily translate to misspecification robustness. As a consequence, an algorithm that is the best pick under correct HPs, might be a suboptimal choice when its HPs are misspecified; another algorithm with better misspecification robustness might be safer to use, especially in those cases where minimising the consequences of potential misspecification is a priority. Thus, overall, the best algorithm choice may depend not only on the properties of the data generating process, but also on the quality of selected HP values. In terms of secondary findings, default HPs seem to perform surprisingly well in many cases, and hence may constitute a viable alternative to tuning. Moreover, in the case of sparse graphs, predictions with optimised HPs seem to include only true edges (no FPs, some FNs), which has critical implications for practice. Another interesting observation is that relatively large sparse graphs (50 nodes) can be recovered with high accuracy, subject to large sample size and the right HPs. It is also important to acknowledge the possibility that robustness to misspecified HPs might be impacted by the cardinality of explored HPs as larger search spaces may increase the probability of poor performances. Our results show this is indeed possible, but crucially the chances of good performances also slightly rise in this case. This suggests that higher HP cardinalities can be advantageous and disadvantageous, which motivates including the worst performances in this analysis. Furthermore, while we agree that the probability of getting the worst performances is low in practice, it is undeniably greater than zero and hence worth considering.

Recommendations. As for practical advice, we stress the fact that there are no universal answers in structure learning; there are many forces at play (DGP properties, algorithms, HPs) that make the choices highly nuanced. Algorithm selection seems to be the most important for performance, though HPs should not be neglected (see Table 1 for guidance). Default HPs (from packages or this paper) should be a reasonable starting point. For further tuning, one can consider prediction stability across HPs (Liu et al., 2010; Sun et al., 2013; Strobl, 2021), though larger HP search spaces should be considered with care (risk of poor performances). Focusing on “tunable” HPs (Probst et al., 2019) may help reduce the search space.

Limitations. This study is naturally limited by our choice of explored algorithms, HPs and simulation properties. It is worth noting, however, that we do not intend to identify the best possible learning algorithm or HPs. On the contrary, the objective of this work is to show that the appropriate algorithm choices are nuanced, as also recently shown in the treatment effect estimation domain (Curth and van der Schaar, 2023), and that HPs should be part of that subtle decision-making process, further confirming the importance of HPs reported in the literature (Paine et al., 2020; Zhang et al., 2021a; Machlanski et al., 2023; Tönshoff et al., 2023). And while more extensive search spaces are unlikely to negate such conclusions, it is worth pointing out that in our experiments we explore discrete HP values of continuous search spaces. As for the choice of HPs and values, we believe our setup accurately reflects common practice.

Future work. As increasing the sample size increases the proportion of good performances across HPs, a next step would be to examine if HP misspecification vanishes with infinite data. More challenging “anterial graphs” (Sadeghi, 2016) with cycles and undirected edges could be another direction. The surprising effectiveness of fixed HPs might be also worth a systematic study, with an emphasis on transfer between DGPs. Further study into the source of robustness to misspecified HPs is also in order (HP cardinality or algorithms). Furthermore, making advanced tuning metrics more accessible could help facilitate better practice. Finally, although some HP tuning metrics are general enough to perform algorithm selection (e.g. Liu et al. (2010); Biza et al. (2022)), doing so on real-life datasets is still a challenge. One promising direction could be validation that incorporates domain knowledge (Grünbaum et al., 2023).

Acknowledgments

We would like to thank the reviewers for their insightful comments and suggestions that helped us improve this paper. This research was supported by the ESRC Research Centre on Micro-Social Change (MiSoC) – ES/S012486/1, and in part through computational resources provided by the Business and Local Government Data Research Centre BLG DRC (ES/S007156/1) funded by the Economic and Social Research Council (ESRC).

References

Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Konstantina Biza, Ioannis Tsamardinos, and Sofia Triantafillou. Out-of-Sample Tuning for Causal Discovery. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022. ISSN 2162-2388. doi: 10.1109/TNNLS.2022.3185842.

Peter Bühlmann, Jonas Peters, and Jan Ernest. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014. ISSN 0090-5364.

David Maxwell Chickering. Optimal Structure Identification With Greedy Search. *Journal of Machine Learning Research*, 3(Nov):507–554, 2002. ISSN ISSN 1533-7928.

Kiattikun Chobtham and Anthony C Constantinou. Tuning structure learning algorithms with out-of-sample and resampling strategies. *arXiv preprint arXiv:2306.13932*, June 2023.

Alicia Curth and Mihaela van der Schaar. In Search of Insights, Not Magic Bullets: Towards Demystification of the Model Selection Dilemma in Heterogeneous Treatment Effect Estimation. *arXiv preprint arXiv:2302.02923*, June 2023.

Frederick Eberhardt. Introduction to the foundations of causal discovery. *International Journal of Data Science and Analytics*, 3(2):81–91, March 2017. ISSN 2364-4168. doi: 10.1007/s41060-016-0038-6.

Agoston E Eiben and Selmar K Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

P. Erdös and A. Rényi. On random graphs. *Publ. math. debrecen*, 6(290-297):18, 1959.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10, 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00524.

Daniel Grünbaum, Maike L. Stern, and Elmar W. Lang. Quantitative probing: Validating causal models with quantitative domain knowledge. *Journal of Causal Inference*, 11(1), January 2023. ISSN 2193-3685. doi: 10.1515/jci-2022-0060.

Isabelle Guyon, Alexander Statnikov, and Berna Bakir Batu. *Cause effect pairs in machine learning*. Springer, 2019.

Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Mingyu Huang and Ke Li. On the hyperparameter landscapes of machine learning algorithms. *arXiv preprint arXiv:2311.14014*, 2023.

A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999. doi: 10.1109/72.761722.

Marcus Kaiser and Maksim Sipos. Unsuitability of NOTEARS for Causal Graph Discovery. *arXiv:2104.05441 [cs, math, stat]*, April 2021.

Diviyan Kalainathan, Olivier Goudet, and Ritik Dutta. Causal Discovery Toolbox: Uncovering causal relationships in Python. *Journal of Machine Learning Research*, 21(37):1–5, 2020. ISSN 1533-7928.

Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H. Maathuis, and Peter Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012. doi: 10.18637/jss.v047.i11.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-Based Neural DAG Learning. In *International Conference on Learning Representations*, September 2019.

Han Liu, Kathryn Roeder, and Larry Wasserman. Stability Approach to Regularization Selection (StARS) for High Dimensional Graphical Models. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Damian Machlanski, Spyridon Samothrakis, and Paul Clarke. Hyperparameter Tuning and Model Evaluation in Causal Effect Estimation. *arXiv preprint arXiv:2303.01412*, March 2023.

Alexander Marx and Jilles Vreeken. Identifiability of Cause and Effect using Regularized Regression. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 852–861, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330854.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter Selection for Offline Reinforcement Learning. *arXiv preprint arXiv:2007.09055*, July 2020.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, March 2000. ISBN 978-0-521-77362-1.

J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, March 2014. ISSN 0006-3444. doi: 10.1093/biomet/ast043.

Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1): 1934–1965, 2019.

Vineet K. Raghu, Allen Poon, and Panayiotis V. Benos. Evaluation of Causal Structure Learning Methods on Mixed Data Types. In *Proceedings of 2018 ACM SIGKDD Workshop on Causal Discovery*, pages 48–65. PMLR, August 2018.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: The Fast Greedy Equivalence Search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3(2):121–129, March 2017. ISSN 2364-4168. doi: 10.1007/s41060-016-0032-z.

Joseph D Ramsey, Kun Zhang, Madelyn Glymour, Ruben Sanchez Romero, Biwei Huang, Imme Ebert-Uphoff, Savini Samarasinghe, Elizabeth A Barnes, and Clark Glymour. Tetrad—a toolbox for causal discovery. In *8th international workshop on climate informatics*, page 29, 2018.

Felix L Rios, Giusi Moffa, and Jack Kuipers. Benchpress: A Scalable and Versatile Workflow for Benchmarking Structure Learning Algorithms. *arXiv preprint arXiv:2107.03863*, May 2021.

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*, 308(5721): 523–529, 2005. doi: 10.1126/science.1105809.

Kayvan Sadeghi. Marginalization and conditioning for LWF chain graphs. *The Annals of Statistics*, 44(4):1792 – 1816, 2016. doi: 10.1214/16-AOS1451.

Lei Shen, Wangshu Liu, Xiang Chen, Qing Gu, and Xuejun Liu. Improving machine learning-based code smell detection via hyper-parameter optimization. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 276–285. IEEE, 2020.

Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030, 2006. ISSN ISSN 1533-7928.

Jascha Sohl-Dickstein. The boundary of neural network trainability is fractal. *arXiv preprint arXiv:2402.06184*, 2024.

Peter Spirtes and Clark Glymour. An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computer Review*, 9(1):62–72, April 1991. ISSN 0894-4393. doi: 10.1177/089443939100900106.

Peter Spirtes, Clark Glymour, and Richard Scheines. Discovery algorithms for causally sufficient structures. In *Causation, Prediction, and Search*, pages 103–162. Springer New York, New York, NY, 1993. ISBN 978-1-4612-2748-9. doi: 10.1007/978-1-4612-2748-9_5.

Eric V. Strobl. Automated hyperparameter selection for the PC algorithm. *Pattern Recognition Letters*, 151:288–293, November 2021. ISSN 0167-8655. doi: 10.1016/j.patrec.2021.09.009.

Wei Sun, Junhui Wang, and Yixin Fang. Consistent Selection of Tuning Parameters via Variable Selection Stability. *Journal of Machine Learning Research*, 14(107):3419–3440, 2013. ISSN 1533-7928.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark. *arXiv preprint arXiv:2309.00367*, September 2023.

Ruibo Tu, Kun Zhang, Bo Bertilson, Hedvig Kjellstrom, and Cheng Zhang. Neuropathic Pain Diagnosis Simulator for Causal Discovery Algorithm Evaluation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Tim Van den Bulcke, Koenraad Van Leemput, Bart Naudts, Piet van Remortel, Hongwu Ma, Alain Verschoren, Bart De Moor, and Kathleen Marchal. SynTReN: A generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics*, 7: 43, January 2006. ISSN 1471-2105. doi: 10.1186/1471-2105-7-43.

Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the Importance of Hyperparameter Optimization for Model-based Reinforcement Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, March 2021a.

Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: A python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021b.

X. Zheng, Bryon Aragam, P. Ravikumar, and E. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *NeurIPS*, 2018.

Xun Zheng, Chen Dan, Bryon Aragam, P. Ravikumar, and E. Xing. Learning Sparse Nonparametric DAGs. In *AISTATS*, 2020.

Appendix A. Experimental Details

A.1. Hyperparameters

We attempted to explore the hyperparameters as thoroughly as possible to make our findings general enough while at the same time managing computational demands that increase with each added hyperparameter and explored value. In addition, some methods are considerably more demanding than others, which makes the exploration even more difficult. With these constraints in mind, we believe our hyperparameter exploration accurately reflects common practice.

Table 2: Hyperparameter search spaces defined per algorithm. Note that hyperparameters are in most cases continuous, but we explore a discrete space of values.

algorithm	hyperparameters and values	card[a] (total)
ANM	$alpha \in \{0.001^*, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5\}$	10 (10)
CAM	$cutoff \in \{0.001^*, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5\}$	10 (10)
	$score =$ nonlinear	
	$selmethod =$ gamboost	
	$prunmethod =$ gam	
FCI	$alpha \in \{0.001^*, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5\}$	10 (10)
	$test =$ fisher-z-test	
FGES	$penaltyDiscount \in \{0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 1.5^*\}$	8 (8)
	$score =$ sem-bic	
LiNGAM	$thresh \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5^*\}$	10 (20)
	$max\_iter \in \{100^*, 1000\}$	2
NOTEARS	$lambda1 \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2^*, 0.3, 0.5\}$	10 (200)
	$max\_iter \in \{100^*, 1000\}$	2
	$w\_threshold \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2^*, 0.3, 0.5\}$	10
	$loss\_type =$ l2	
	$h\_tol = 1e-8$	
	$rho\_max = 1e+16$	
NOTEARS MLP	$lambda1 \in \{0.001, 0.01^*, 0.1\}$	3 (81)
	$lambda2 \in \{0.001, 0.01, 0.1^*\}$	3
	$w\_threshold \in \{0.1, 0.3, 0.5^*\}$	3
	$hidden\_layers = 1$	
	$hidden\_units \in \{8, 16^*, 32\}$	3
	$max\_iter = 100$	
	$h\_tol = 1e-8$	
	$rho\_max = 1e+16$	
PC	$alpha \in \{0.001, 0.002^*, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5\}$	10 (10)
	$indepTest =$ gaussCItest	

[*]Found to perform best on average across all simulations (*sim_mean*).

[a]Cardinality of the hyperparameters considered in the experiments.

A.2. Summary of Algorithms

This study incorporates the following algorithms and hyperparameters.

- **PC** (Spirtes and Glymour, 1991). Peter and Clark algorithm. Constraint-based approach that starts with a fully-connected undirected graph and removes edges based on conditional independence tests. Next, it attempts to orient as many of the remaining edges as possible. The result is a CPDAG.
  Hyperparameters: *alpha* (significance level for conditional independence tests).

- **FCI** (Spirtes et al., 1993). Fast Causal Inference. Constraint-based. An important generalisation of PC to unknown confounding variables.
  Hyperparameters: *alpha* (significance level for conditional independence tests).

- **FGES** (Ramsey et al., 2017). Fast Greedy Equivalence Search. Optimised and parallelised version of the original score-based GES algorithm (Chickering, 2002). It starts with an empty graph and adds an edge that yields maximum score improvement until no significant score gain is achieved. Then it removes edges in the same greedy manner until a plateau.
  Hyperparameters: *penaltyDiscount* (sparsity penalty).

- **LiNGAM** (Shimizu et al., 2006). Linear Non-Gaussian Acyclic Model. Assumes linear SEMs and non-Gaussian noise that enters additively: $X_j = \sum_{k \in \mathbf{PA}_j^{\mathcal{G}}} w_{jk} X_k + \epsilon_j$.
  Hyperparameters: *max_iter* (FastICA (Hyvarinen, 1999)), *thresh* (post-prunning threshold).

- **ANM** (Hoyer et al., 2008). Additive Noise Model. Assumes nonlinear SEMs and additive noise: $X_j = f_j(X_{\mathbf{PA}_j^{\mathcal{G}}}) + \epsilon_j$.
  Hyperparameters: *alpha* (significance level for the independence test).

- **CAM** (Bühlmann et al., 2014). Causal Additive Models. Assumes a generalised additive noise model with additive noise and functions: $X_j = \sum_{k \in \mathbf{PA}_j^{\mathcal{G}}} f(X_k) + \epsilon_j$.
  Hyperparameters: *cutoff* (variable selection threshold).

- **NOTEARS** (Zheng et al., 2018). Score-based continuous DAG optimisation with a smooth acyclicity regularisation term. Assumes linear SEMs with additive noise.
  Hyperparameters: *lambda1* (sparsity term), *max_iter* (optimisation steps) and *w_threshold* (post-prunning threshold).

- **NOTEARS MLP** (Zheng et al., 2020). Nonlinear extension of *NOTEARS* by incorporating the Multi-Layer Perceptron (MLP). Assumes nonlinear SEMs with additive noise.
  Hyperparameters: *lambda1* (sparsity term), *lambda2* (regularisation strength), *w_threshold* (post-prunning threshold), *hidden_units* (number of units in the hidden layer).

Many traditional algorithms, such as PC, FCI and FGES, make the standard set of assumptions that involve sufficiency, faithfulness and Markov condition. These, however, are often not enough to identify a unique DAG as a solution, which is a major drawback of these methods (they output CPDAGs). Making assumptions about distributions and functional forms of the data generating process seems to be critical to overcome this issue (all methods above except for PC, FCI and FGES output DAGs).

Table 3: Summary of incorporated algorithms and their sources. Recommended default hyperparameters have been derived from respective papers as much as possible. If necessary, they have been further supplemented with defaults suggested within respective packages.

algorithm	default hyperparameters	package	paper
ANM	$alpha = 0.05$	gCastle	(Hoyer et al., 2008)
CAM	$cutoff = 0.001$ $score =$ nonlinear $selmethod =$ gamboost $prunmethod =$ gam	cdt	(Bühlmann et al., 2014)
FCI	$alpha = 0.01$	tetrad	(Spirtes et al., 1993)
FGES	$penaltyDiscount = 2.0$	tetrad	(Ramsey et al., 2017)
LiNGAM	$thresh = 0.3$ $max\_iter = 1000$	gCastle	(Shimizu et al., 2006)
NOTEARS	$lambda1 = 0.1$ $max\_iter = 100$ $w\_threshold = 0.3$ $loss\_type =$ l2 $h\_tol = 1e - 8$ $rho\_max = 1e + 16$	gCastle	(Zheng et al., 2018)
NOTEARS MLP	$lambda1 = 0.01$ $lambda2 = 0.01$ $w\_threshold = 0.3$ $hidden\_layers = 1$ $hidden\_units = 10$ $max\_iter = 100$ $h\_tol = 1e - 8$ $rho\_max = 1e + 16$	gCastle	(Zheng et al., 2020)
PC	$alpha = 0.01$	pcalg	(Spirtes and Glymour, 1991)

A.3. Summary of Packages

Table 4: Summary of incorporated algorithm packages.

package	paper	link
gCastle	(Zhang et al., 2021b)	https://github.com/huawei-noah/trustworthyAI/tree/master/gcastle
cdt	(Kalainathan et al., 2020)	https://fentechsolutions.github.io/CausalDiscoveryToolbox
tetrad	(Ramsey et al., 2018)	https://cmu-phil.github.io/tetrad/manual/
pcalg	(Kalisch et al., 2012)	https://cran.r-project.org/package=pcalg

A.4. Performance Evaluation

We incorporate commonly used evaluation metrics that are provided via Benchpress (Rios et al., 2021, appendix A.1.). For the convenience of the reader, we briefly describe here those that are useful for this study.

A.4.1. SHD

Let us define E and E' as a set of edges of the true and predicted DAG respectively. Then, for $e \in E'$, true positives (TP) and false positives (FP) are assigned as follows:

$$TP(e) = \begin{cases} 1 & \text{if } e \in E \text{ and correctly oriented} \\ 0.5 & \text{if } e \in E \text{ and incorrectly oriented} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$FP(e) = \begin{cases} 1 & \text{if } e \notin E \\ 0.5 & \text{if } e \in E \text{ and incorrectly oriented} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where TP and FP are sums of all TP(e) and FP(e) scores respectively. The *structural Hamming distance* (SHD) aggregates the number additions, removals and reversals in predicted edges so they match the true ones ($E = E'$). It can be defined as:

$$SHD = |E| - TP + FP \tag{6}$$

Note the SHD defined as above allows to evaluate mixed graphs, that is, compare DAGs to CPDAGs. If, for instance, a predicted undirected edge exists in E but is supposed to be directed, it will result in $TP = 0.5$ and $FP = 0.5$, ultimately leading to $SHD = 1$. This shows that the need to orient an undirected edge is treated equally as the need to add, remove or reverse an edge so $E = E'$. Such evaluation puts algorithms outputting CPDAGs at a disadvantage compared to DAG-only methods. We justify it on the grounds that the main focus of this study is DAG recovery, hence any predicted undirected edge is treated as any other mistake. The ability to evaluate mixed graphs is an important feature for this study as it allows us to compare algorithms outputting CPDAGs and DAGs.

A.4.2. False Positives and Negatives

The *false positives* (FPs) and *false negatives* (FNs) that we use as standalone performance metrics differ from those defined as part of SHD above. Crucially, the FP and FN metrics we use are always computed based on graph skeletons. To achieve this, all directed edges of a predicted or true graph are converted to undirected ones. This modification makes the comparison between algorithms that output CPDAGs and DAGs more fair. Once the graphs in question are converted to skeletons, we can define the metrics as follows.

Let us define E and E' as a set of undirected edges of the true and predicted graph skeleton respectively. Then, for $e \in E$ and $e' \in E'$, false positives (FP) and false negatives (FN) are assigned as follows:

$$FP(e') = \begin{cases} 1 & \text{if } e' \notin E \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$FN(e) = \begin{cases} 1 & \text{if } e \notin E' \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where FP and FN are sums of all FP(e') and FN(e) scores respectively.

A.5. Code and Data

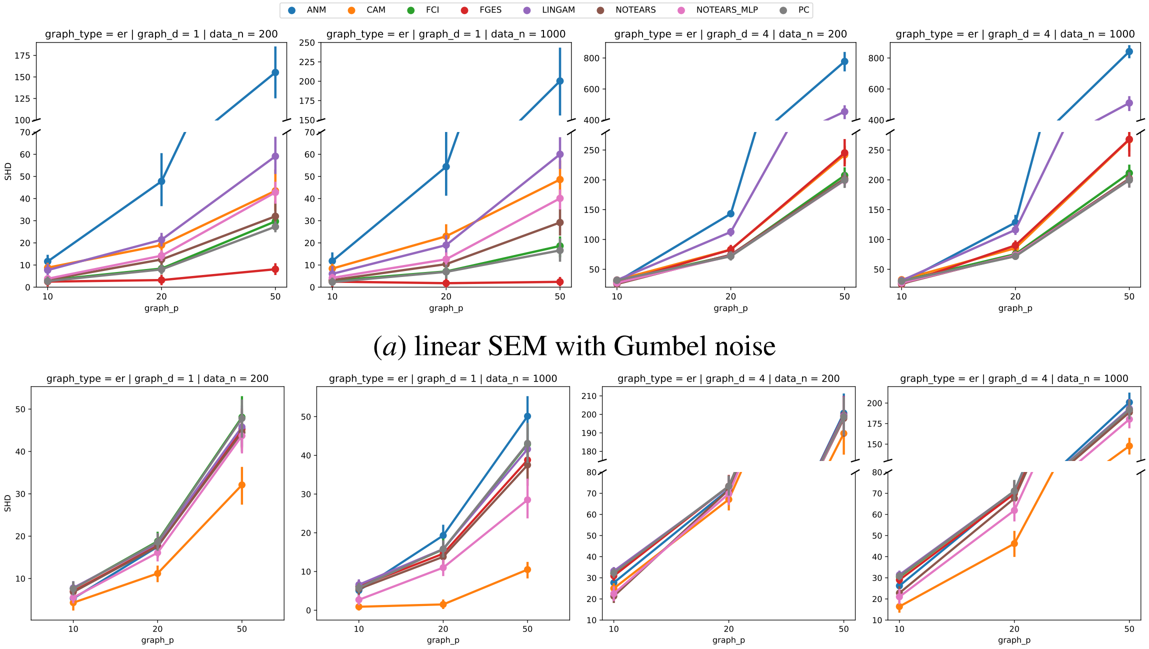
All numerical experiments can be fully replicated using the code and data that are available online at: `https://github.com/misoc-mml/hyperparams-causal-discovery`.

Appendix B. Supplemental Results

The following results complement the ones presented in the main content of the paper. Although they do not change the overall conclusions of the paper, they offer additional analysis that may facilitate a deeper understanding of the problem and the final outcomes.

B.1. Best Performances

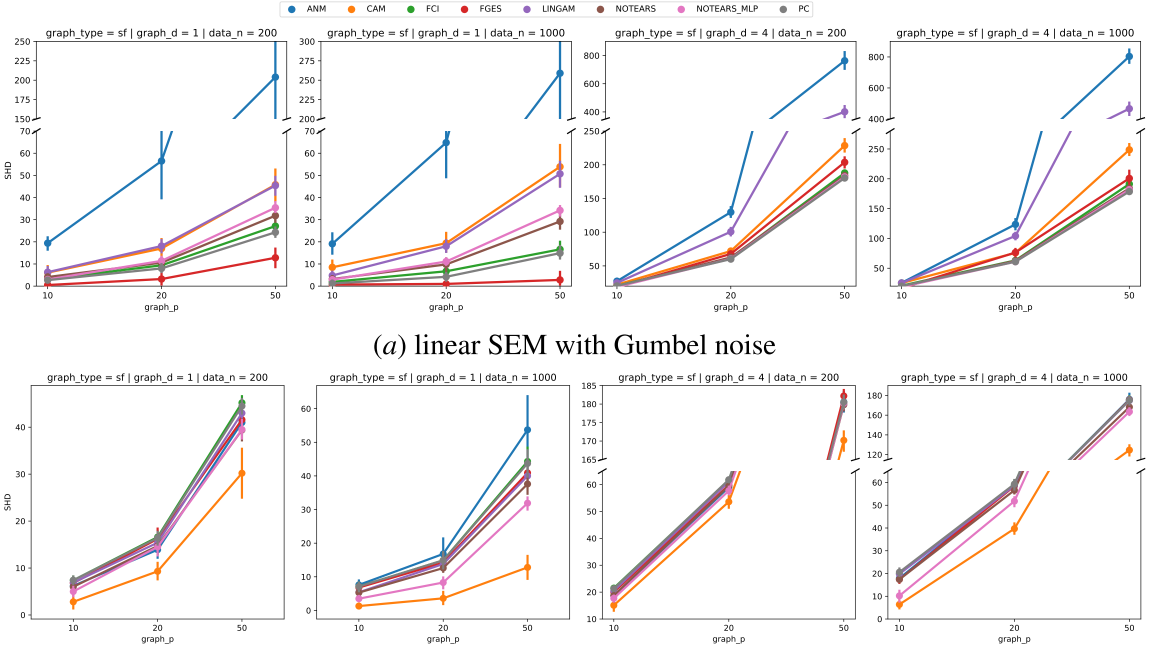
Figures 8 and 9 focus on performances achieved specifically with the best (oracle) hyperparameters. These correspond to the best performances presented in Figure 4 (blue bars). Some important observations: a) algorithms differ in performance even with access to a hyperparameter oracle, b) number of graph nodes and edge density can significantly impact performance, and c) no algorithm performs best under all conditions (linear vs. nonlinear).



(*a*) linear SEM with Gumbel noise

(*b*) nonlinear (GP) SEM with Gaussian noise

Figure 8: Best performances against ER graphs. Error bars are standard errors.



(*a*) linear SEM with Gumbel noise

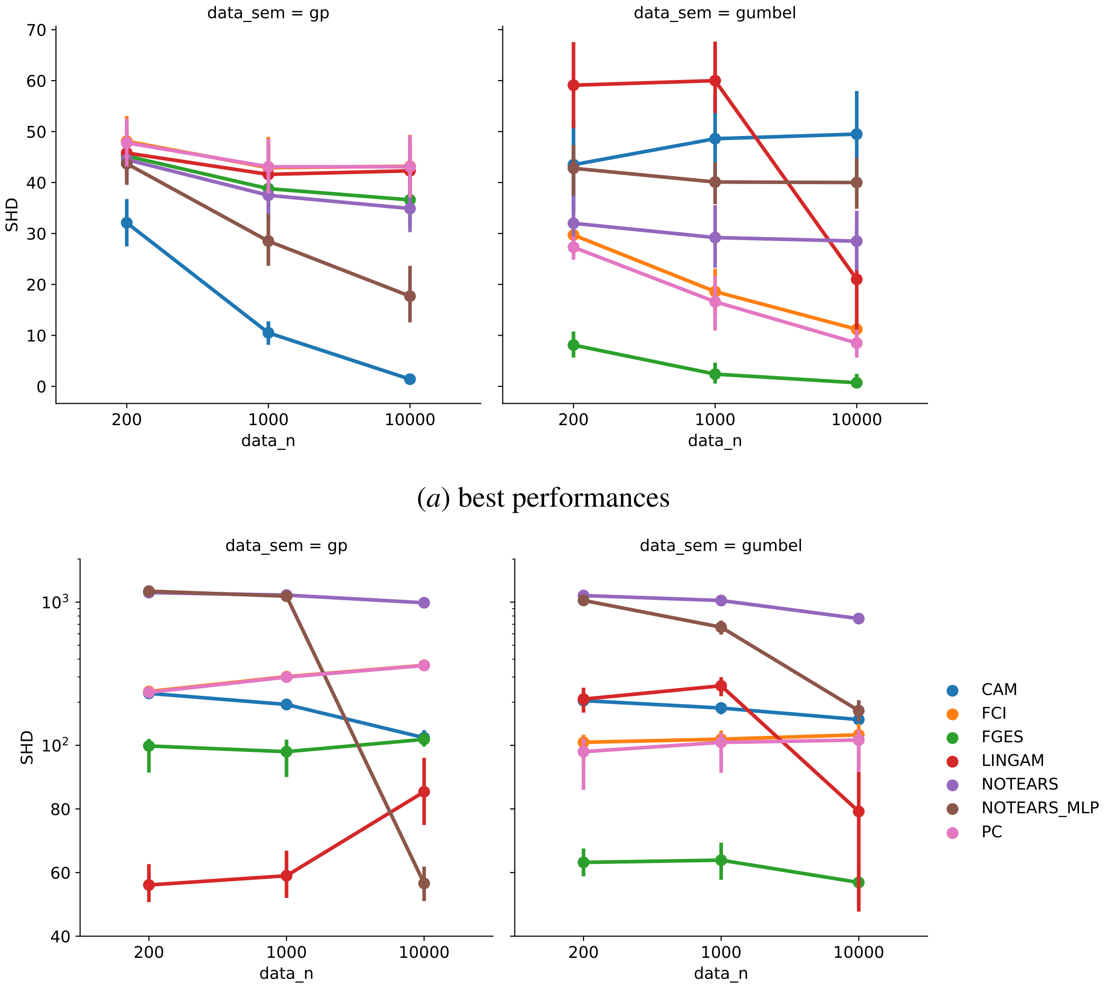
(*b*) nonlinear (GP) SEM with Gaussian noise

Figure 9: Best performances against SF graphs. Error bars are standard errors.

B.2. Large Graphs with Large Sample Size

Previous results showed that $p = 50$ graphs are much more challenging than smaller ones. Figure 10a demonstrates that with enough data samples, even such larger graphs are possible to solve accurately. As presented in the subfigure (b), increased sample size can also help with robustness. Note also how the degree of the benefits vary between algorithms.

Figures 11, 12 and 13 further extend the results to performance distributions over all hyperparameters (SHD, FP, and FN respectively). Notice how larger sample size increases the proportion of good performances (the lines shift to the left and hit higher proportion numbers for lower metric values). If the improvement trend remains for even larger sample sizes, one can wonder if the HP misspecification issue could be solved entirely by larger quantities of data alone.



(*a*) best performances

(*b*) worst performances

Figure 10: Performances for ER1 $p = 50$ graphs.

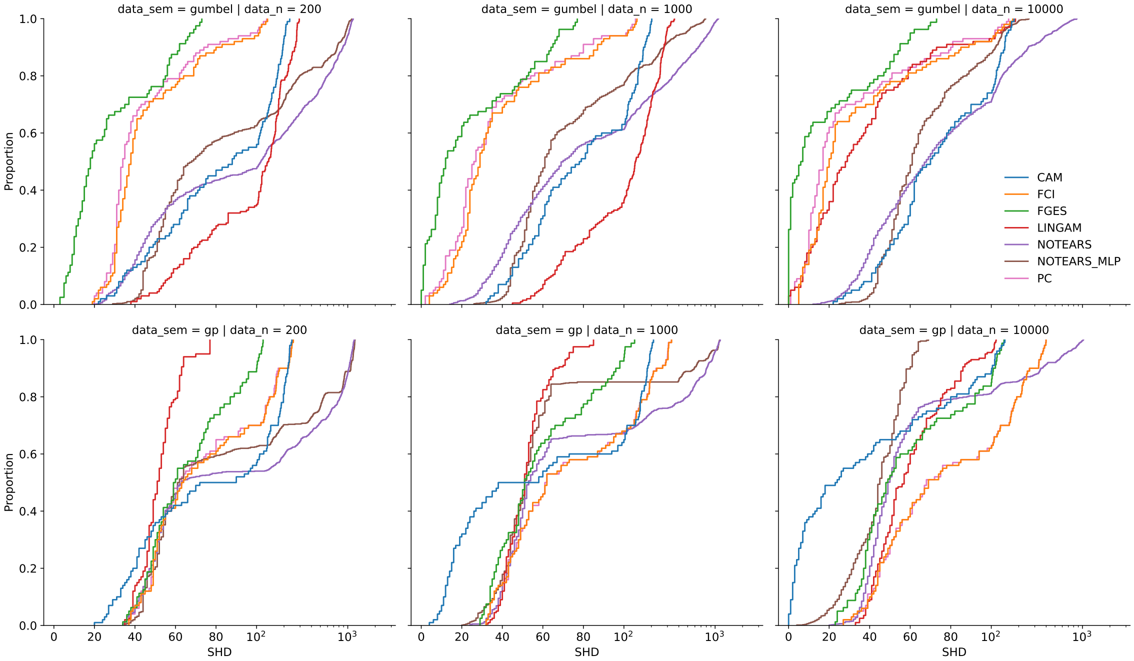


Figure 11: Distribution of SHD performances across all hyperparameters (ER1 $p = 50$ graphs).

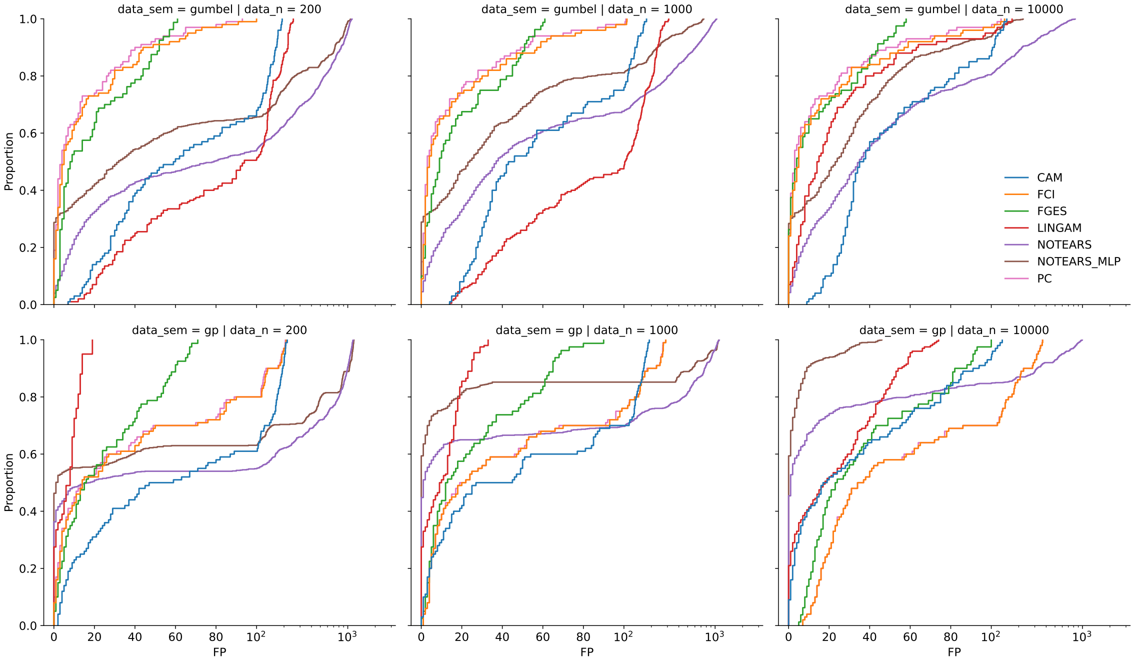


Figure 12: Distribution of false positives (FPs) across all hyperparameters (ER1 $p = 50$ graphs).

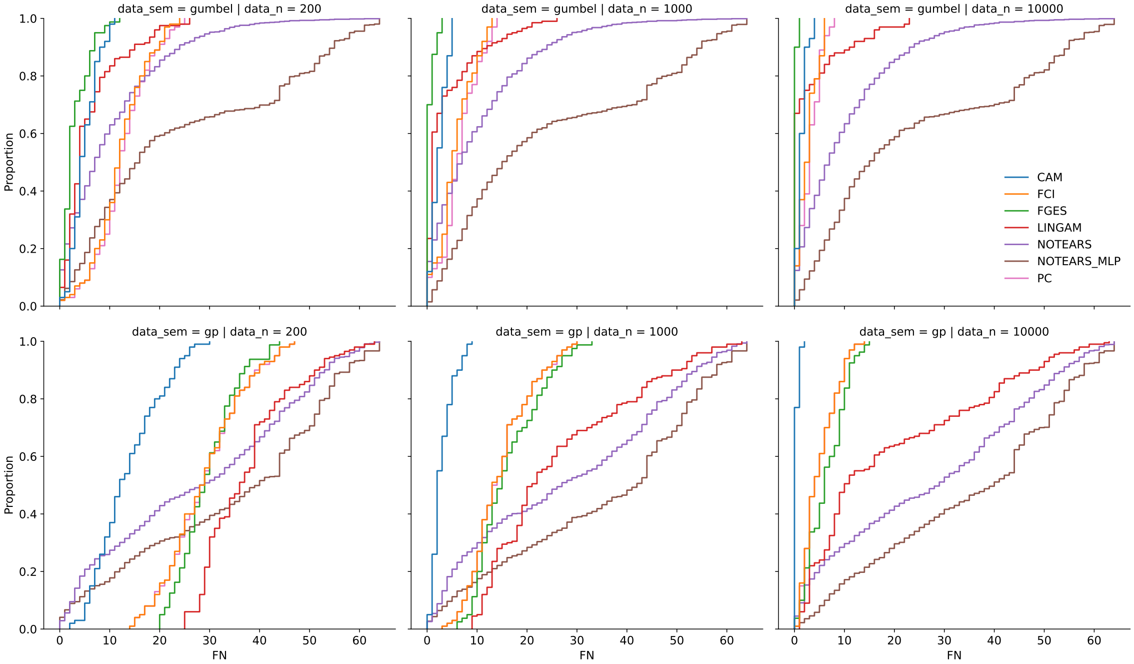


Figure 13: Distribution of false negatives (FNs) across all hyperparameters (ER1 $p = 50$ graphs).

B.3. Performance vs. Hyperparameter Quality

Figures 14, 15 and 16 complement Figure 4 from the main content by showing the results for other types of DGPs.

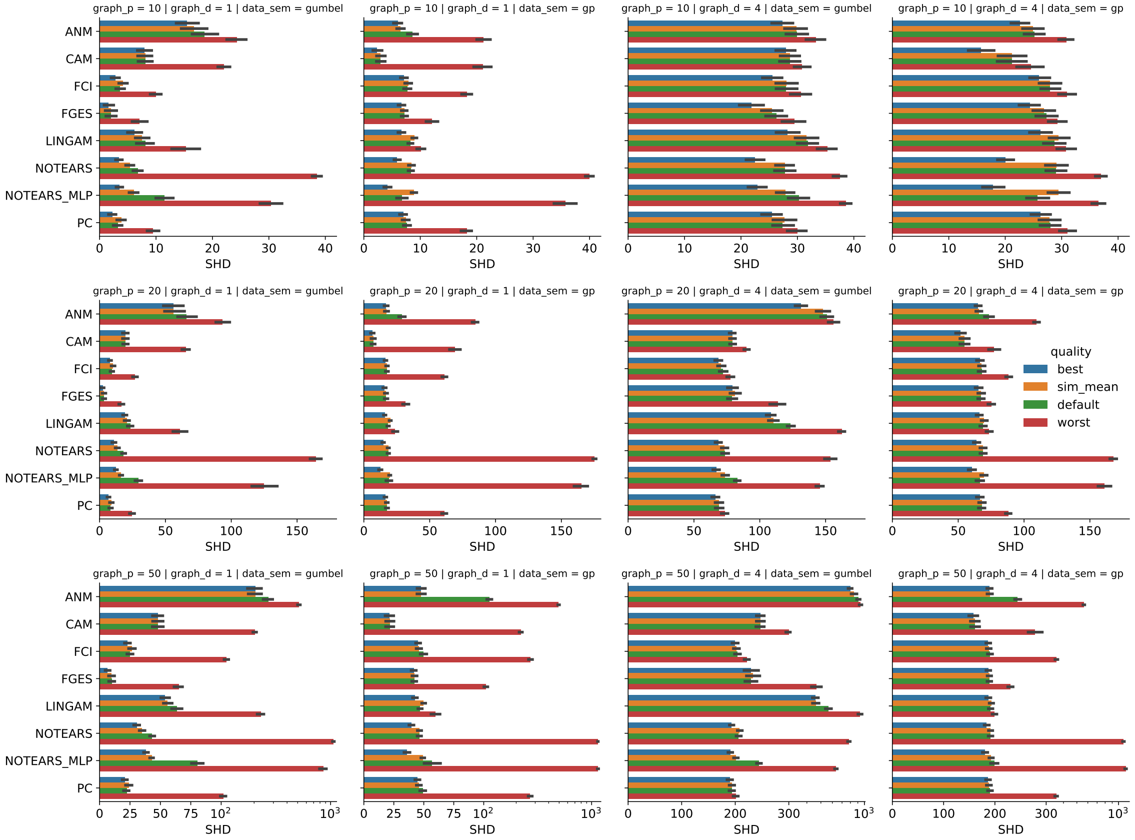


Figure 14: SHD performances depending on the quality of selected hyperparameters (colours), grouped by DGP properties such as number of nodes (*graph_p*), edge density (*graph_d*) and SEM type (*data_sem*; gumbel is linear, gp nonlinear).

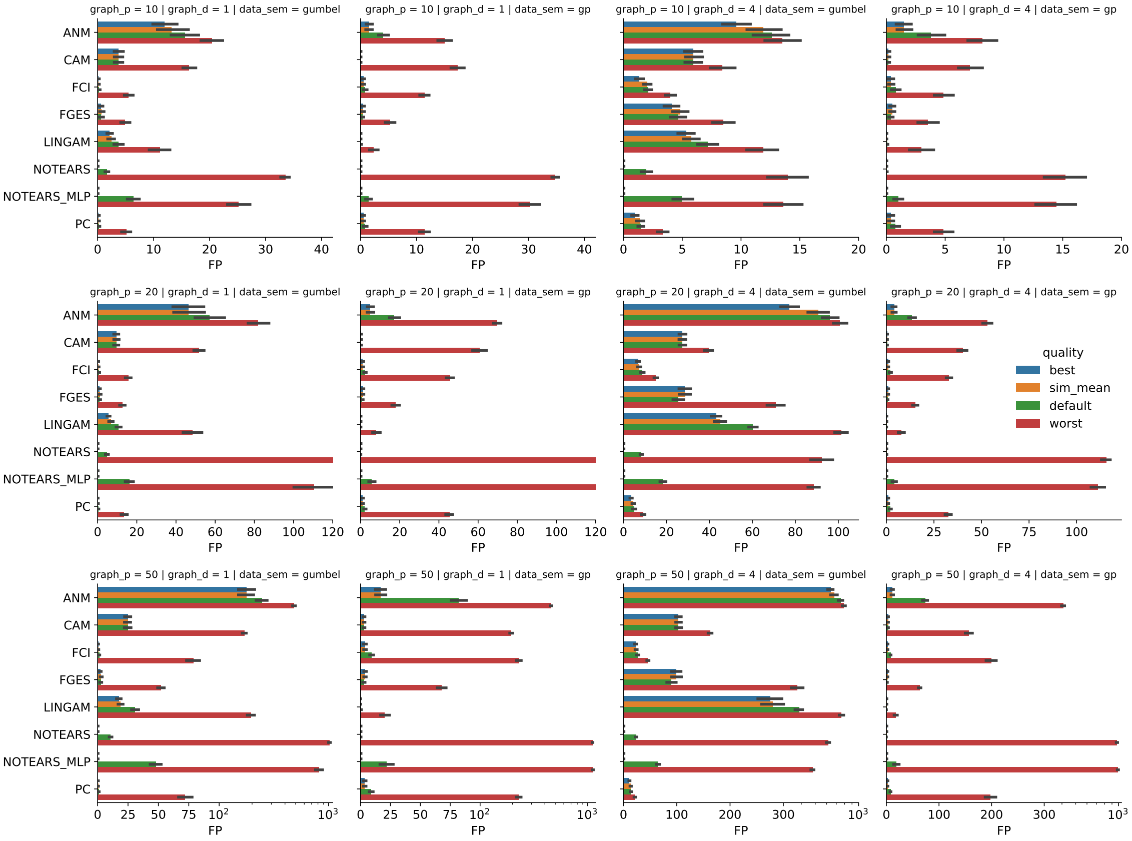


Figure 15: FP (false positive) performances depending on the quality of selected hyperparameters (colours), grouped by DGP properties such as number of nodes (*graph_p*), edge density (*graph_d*) and SEM type (*data_sem*; gumbel is linear, gp nonlinear).

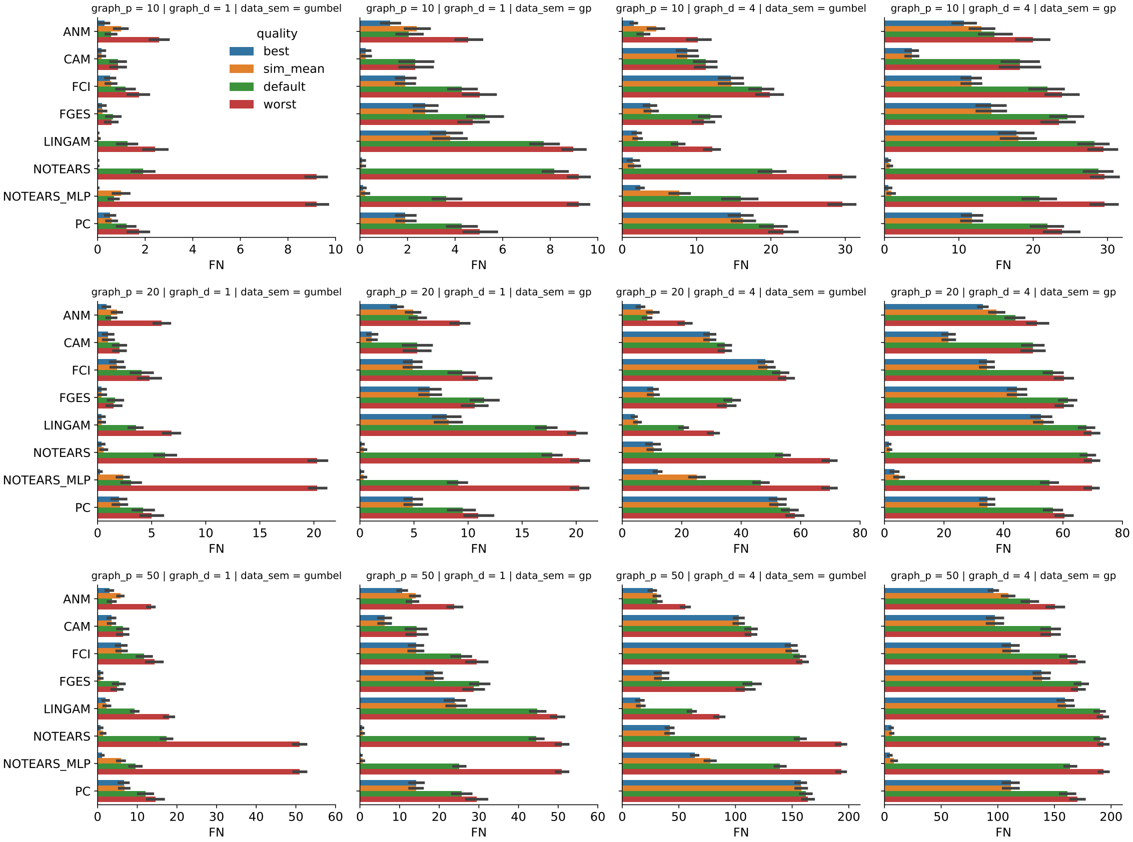
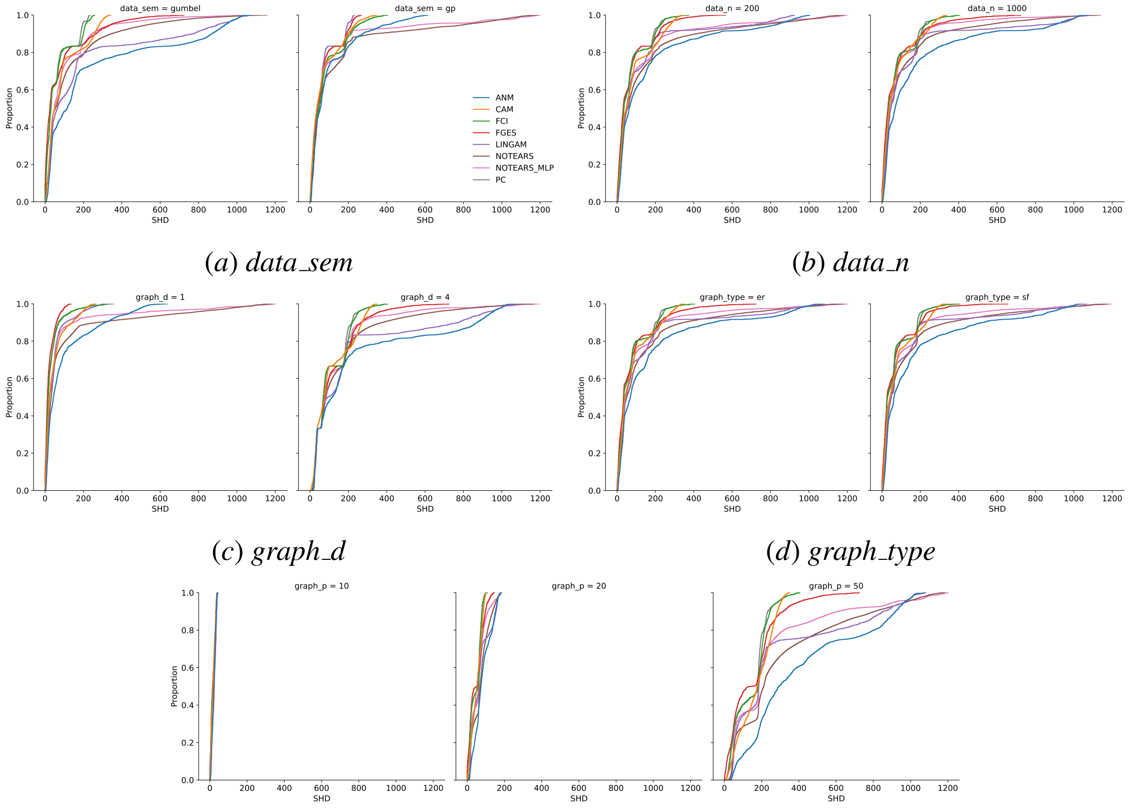


Figure 16: FN (false negative) performances depending on the quality of selected hyperparameters (colours), grouped by DGP properties such as number of nodes (*graph_p*), edge density (*graph_d*) and SEM type (*data_sem*; gumbel is linear, gp nonlinear).

B.4. Performance Distribution Across Hyperparameters

Figures 17, 18 and 19 complement Figure 3 from the main content by showing the results for other types of DGPs.



(*a*) *data_sem*

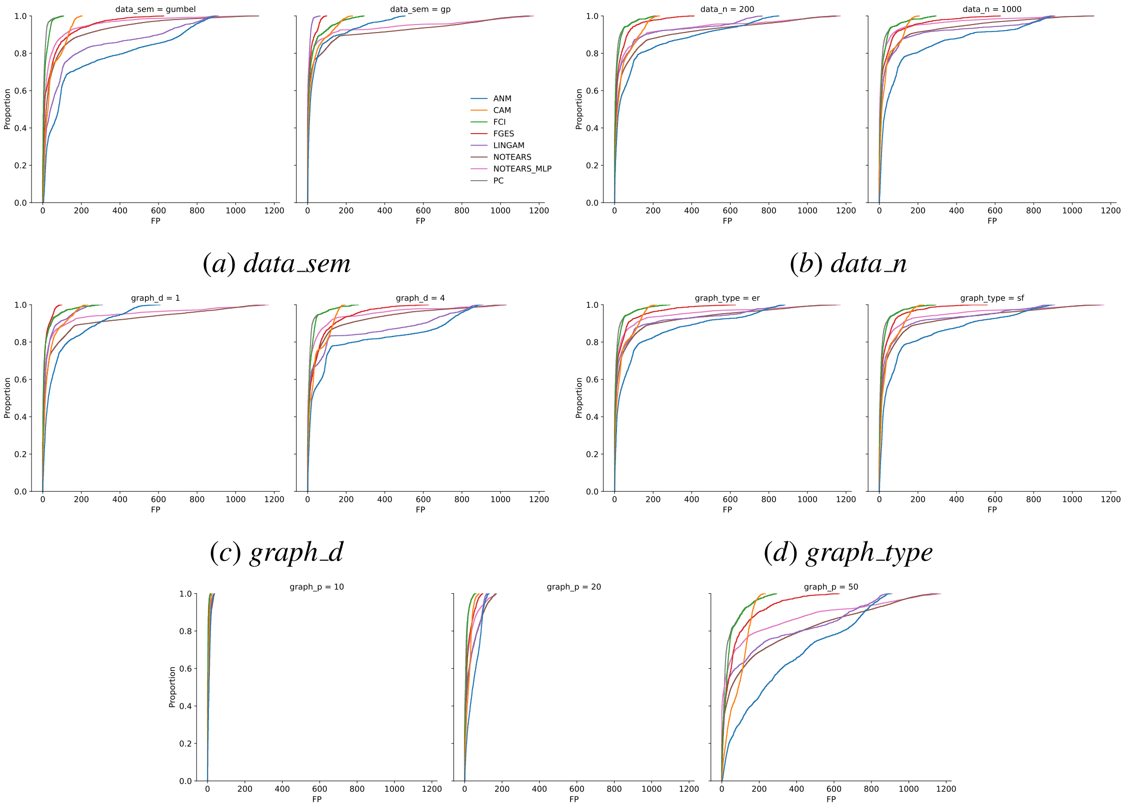
(*b*) *data_n*

(*c*) *graph_d*

(*d*) *graph_type*

(*e*) *graph_p*

Figure 17: Distributions of SHD performances across all hyperparameters, grouped by SEM types (*data_sem*), sample size (*data_n*), edge density (*graph_d*), graph type (ER or SF) and number of nodes (*graph_p*).



(*a*) *data_sem*

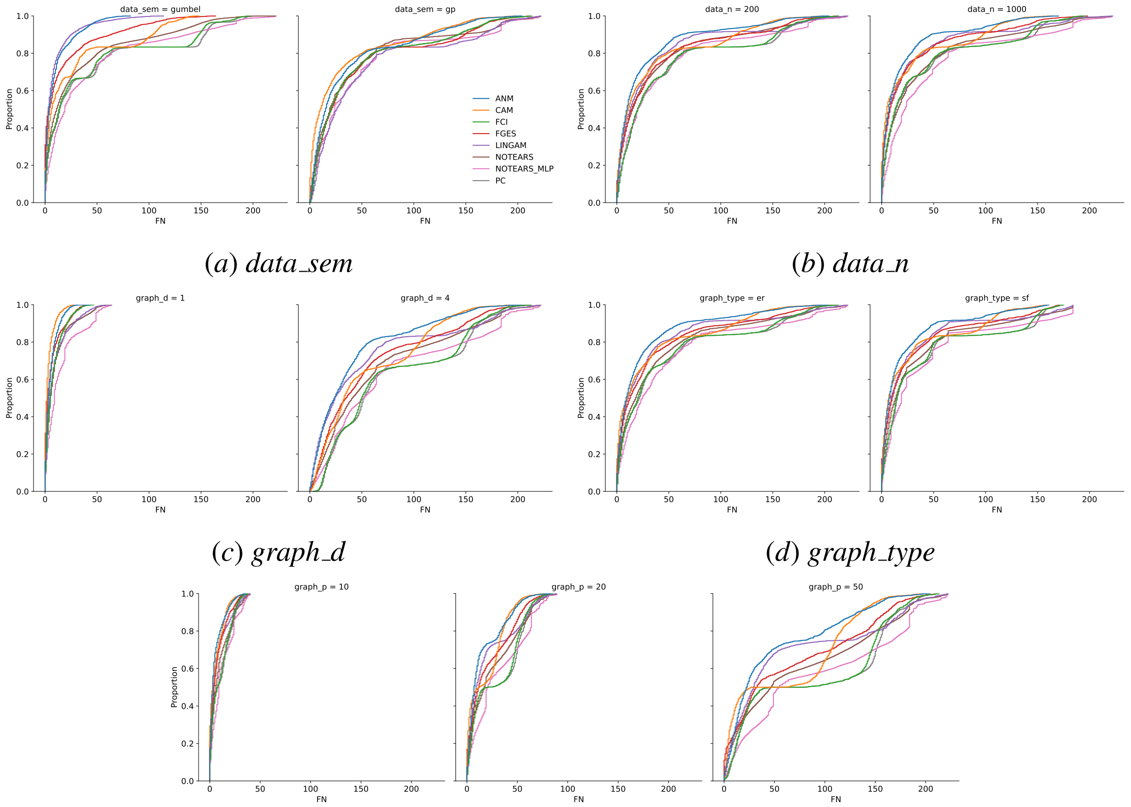
(*b*) *data_n*

(*c*) *graph_d*

(*d*) *graph_type*

(*e*) *graph_p*

Figure 18: Distributions of false positive (FP) performances across all hyperparameters, grouped by SEM types (*data_sem*), sample size (*data_n*), edge density (*graph_d*), graph type (ER or SF) and number of nodes (*graph_p*).



(*a*) *data_sem*

(*b*) *data_n*

(*c*) *graph_d*

(*d*) *graph_type*

(*e*) *graph_p*

Figure 19: Distributions of false negative (FN) performances across all hyperparameters, grouped by SEM types (*data_sem*), sample size (*data_n*), edge density (*graph_d*), graph type (ER or SF) and number of nodes (*graph_p*).

B.5. Winning Algorithms vs. Simulation Properties

Figure 20 complements Table 1 from the main content but removes hyperparameters from the picture in order to analyse how DGP properties alone affect winning odds of the algorithms. The results presented here involve the use of the best hyperparameter values. From this perspective, it is clear that no algorithm is the best under all conditions, and that SEM types involved and edge density have the strongest impact on the winning odds.

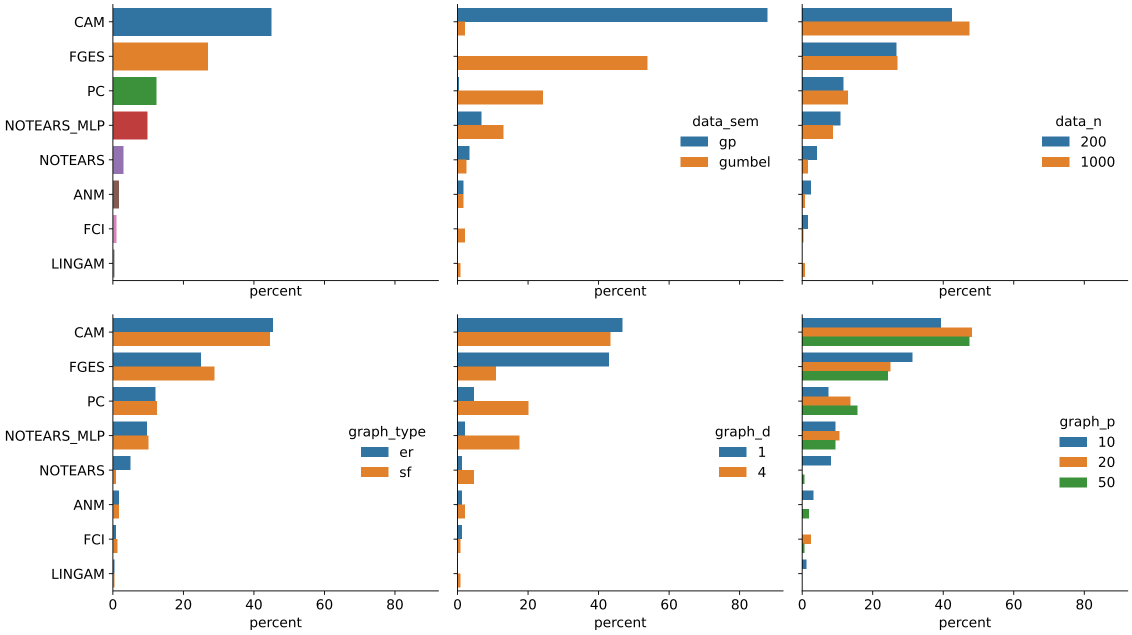


Figure 20: Percentage of wins per algorithm across different DGPs.

B.6. Winning Algorithms vs. Hyperparameter Quality

Figure 21 forms the basis for Table 1 from the main content. It presents winning percentages of algorithms across different DGP types, from which Table 1 was derived.

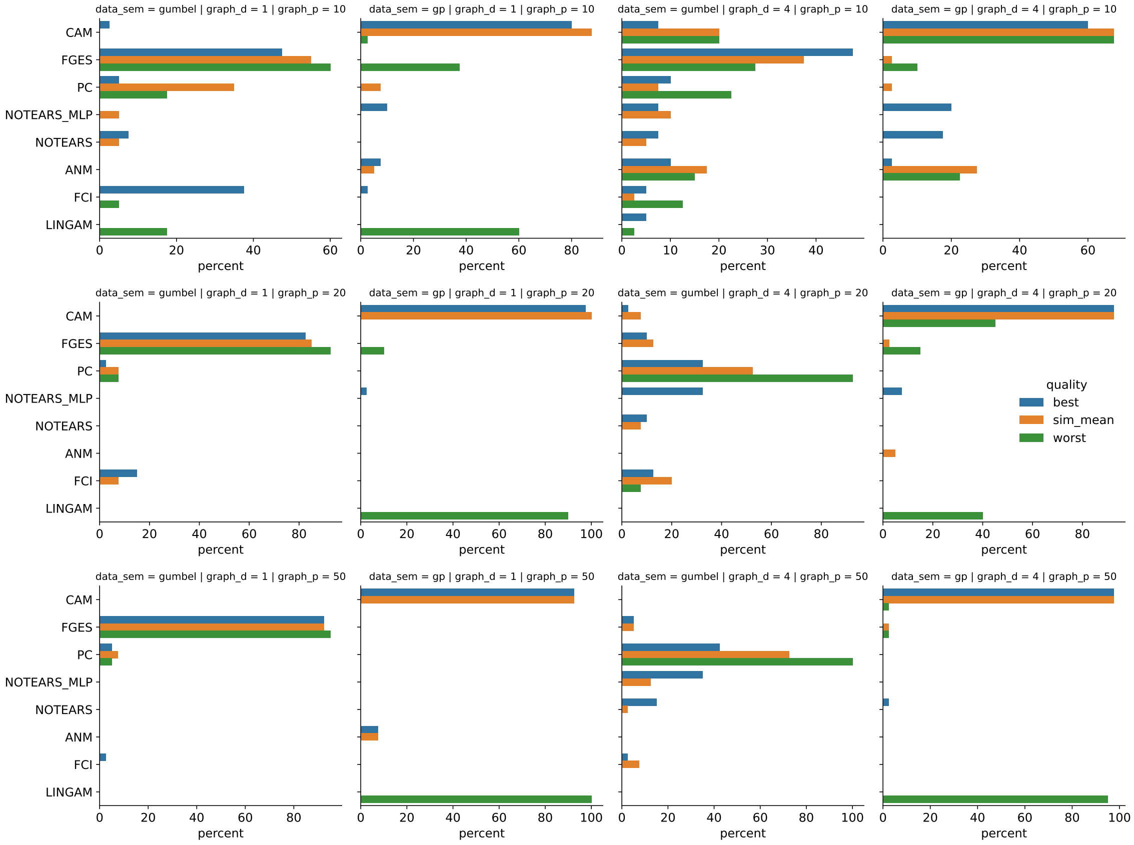


Figure 21: Percentages of winning algorithms under different DGP and hyperparameter conditions.

Appendix C. A Guide To Algorithm Selection

C.1. General Recommendations

- Current algorithms seem to work reasonably well for sparse graphs of up-to 20 nodes. Bigger graphs (50 nodes) are also possible to solve, but more data might be required ($10,000$ samples) to achieve good accuracy. The accuracy of recovered structures drops dramatically for dense graphs.
  Recommendation: Stick to sparse graphs with up-to 20 nodes (moderate amount of data) or up to 50 nodes (a lot of data). Avoid dense graphs.

- No single algorithm is the best option for all problems. Some perform the best under very specific conditions.
  Recommendation: Choose an algorithm that is the most likely to accurately solve the problem at hand based on assumptions derived from data.

- The best choice of an algorithm may depend not only on graph and data properties, but also on the availability of quality hyperparameters. This is because algorithms vary in robustness to misspecified hyperparameters.
  Recommendation: When selecting the best algorithm for the problem at hand, take into account the type of hyperparameters that are available and algorithm's robustness to misspecified hyperparameters.

C.2. Hyperparameter Selection Strategies

Optimal hyperparameters are almost never available in structure recovery problems due to inaccessible ground truth. Some methods provide scores that can be used to decide whether a set of hyperparameters is better than others for the same algorithm. However, this strategy cannot be used to compare different algorithms to each other, as they are likely to use different score metrics (while some use none at all). In addition, in order to use those scores, an algorithm's internal code must be modified in most cases, creating a substantial barrier to practitioners.

Thus, default hyperparameters might be a reasonable selection strategy as they often work almost as well as the optimal ones. This is especially the case if the recommended defaults have been derived from data problems similar in nature to the problem at hand.

If, however, the problem to solve is believed to be fairly unique, blindly using default hyperparameters without considering other factors might not be safe. In this case, the best course of action (assuming hyperparameter tuning is not an option) might be to still use default hyperparameters but choose the algorithm that is the most robust to hyperparameter misspecification under specific graph and data conditions that are believed to be applicable to the problem at hand.

C.3. How to Select Algorithms

Figure 22 summarises the best algorithm choices based on Figure 21, and which takes into consideration data and graph properties as well as the types of hyperparameters available.

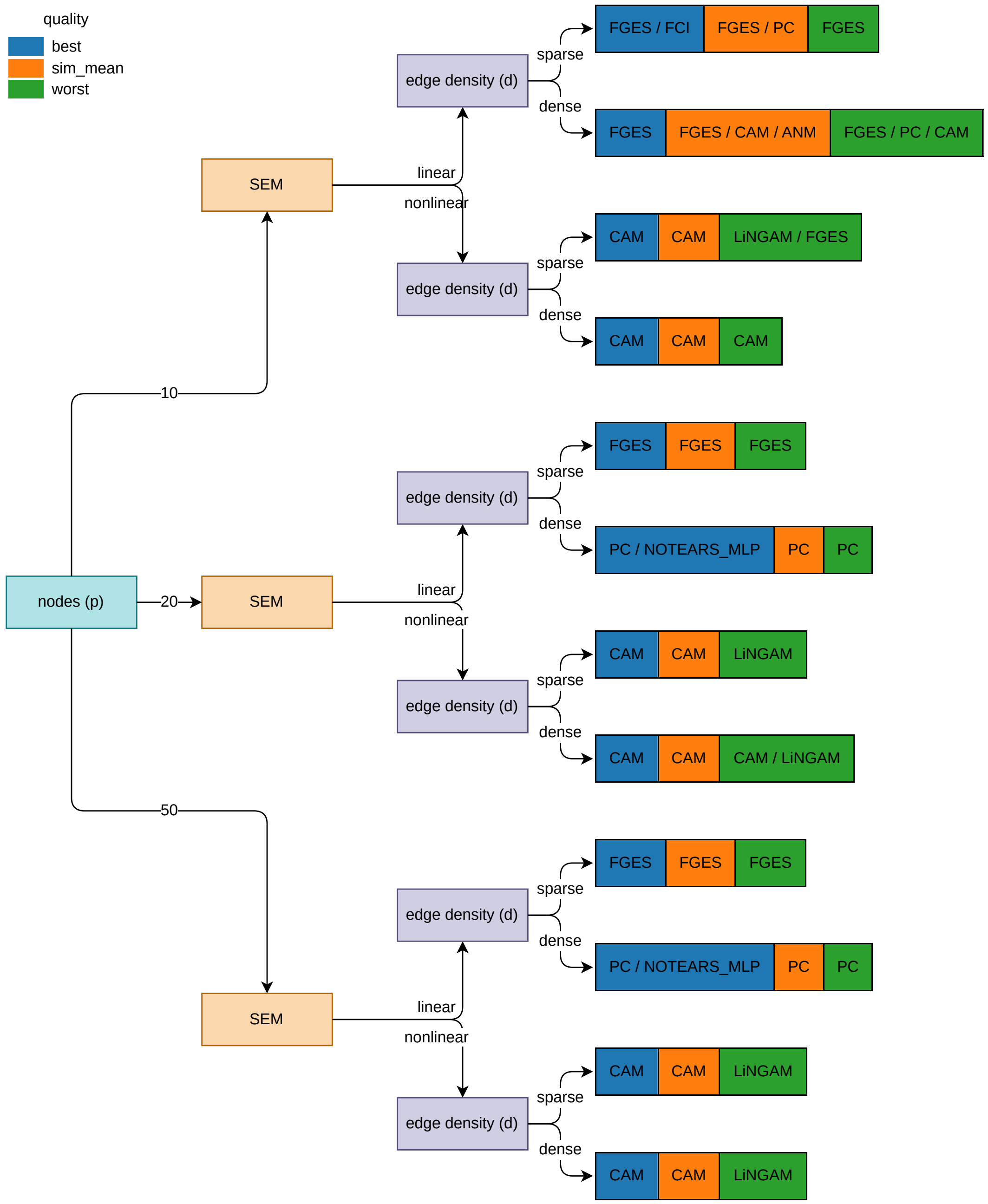


Figure 22: Recommended algorithm choices based on the number of graph nodes, SEM types and edge density in the graph. The final choice depends also on the type of hyperparameters available – see ‘quality’ colours. In case there is no clear winner, multiple choices are provided in the order of higher winning percentage.