Benchmarking Causal Discovery Under Analyst Misspecification

Edgar Davtyan

January 5, 2026

Abstract

This thesis investigates how reliably commonly used causal discovery algorithms recover known causal structures from synthetic data generated from accepted benchmark networks. It also explores how mis–specification of an analyst's directed acyclic graph (DAG) can be detected through data–driven checks, and surveys open–source tools available for constructing and validating causal graphs. We develop a reproducible benchmarking framework that runs PC, GES, NOTEARS and COSMO on five benchmark networks (Asia, Sachs, ALARM, Child, and Insurance), computing precision, recall, F_1 and structural Hamming distance (SHD). We then design controlled scenarios where a human analyst omits a true causal link or adds a spurious edge and show how conditional independence tests and bootstrap stability metrics can help notify the analyst of such errors. Finally, we provide practitioner guidance on building and checking causal diagrams using contemporary software and algorithms.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbrev.	Meaning
CI	Conditional independence
CPDAG	Completed partially directed acyclic graph
COSMO	Constrained Orientations by Sequential M Operation
DAG	Directed acyclic graph
FDR	False discovery rate
GES	Greedy Equivalence Search
IID	Independent and identically distributed
NOTEARS	Non-combinatorial optimisation via trace exponential and augmented lagrangian for structure learning
PC	Peter–Clark algorithm
SCM	Structural causal model
SEM	Structural equation model
SHD	Structural Hamming distance
SID	Structural intervention distance

1 Introduction

Causal diagrams—directed acyclic graphs (DAGs) that encode cause–effect relationships between variables—are indispensable for reasoning about interventions and policy decisions. They consist of nodes representing variables and directed edges denoting direct causal effects. A directed graph qualifies as a DAG if it contains no directed cycles (no node can reach itself by repeatedly following arrows). When DAGs are interpreted causally, analysts typically make additional assumptions—most notably *causal sufficiency* (no unmeasured common causes of included variables) and *faithfulness* (conditional independences in the data correspond to graph separations)—that are convenient but rarely guaranteed in practice [1, 2]. When practitioners draw such diagrams incorrectly, downstream causal inference and decision-making can be compromised, motivating systematic ways to benchmark discovery methods and to diagnose analyst misspecification. We operationalize analyst misspecification as an analyst-proposed DAG that omits a true causal link or includes a non-existent link.

This thesis pursues three goals. First, we benchmark multiple structure–learning methods—PC, GES, NOTEARS and COSMO—on established benchmark networks to evaluate how well they recover known causal structures. The standardized framework ensures that comparisons are meaningful by using shared metrics and reproducible implementations. Second, we study how analyst mis–specification propagates: if a practitioner erroneously removes an edge or inserts a spurious link, can the data alert them? We design controlled experiments where the true DAG generates data but the analyst's DAG deviates from it. Third, we survey open–source tools for drawing and testing DAGs to provide practical guidance on constructing and validating causal diagrams.

These goals translate into three research questions:

RQ1 *How do causal discovery algorithms compare in recovering ground–truth network structures across different data types and network sizes?* We hypothesise that algorithm performance depends strongly on the match between algorithmic assumptions (e.g. linearity, Gaussianity) and data characteristics.

RQ2 *Can conditional independence tests reliably detect when an analyst's DAG omits a true edge or includes a spurious one?* We hypothesise that omitted edges produce significant dependence signals, while spurious edges yield non–significant results, enabling data–driven model criticism.

RQ3 *What practical guidance can we offer practitioners for selecting algorithms and validating their causal assumptions?* We synthesise our empirical findings into actionable recommendations.

The remainder of the thesis is organized as follows. Section 2 reviews the basics of causal DAGs, conditional independence (CI) and common structure–learning algorithms. Section 3 summarises related work on benchmarking causal discovery, mis–specification detection and DAG drawing software. Section 4 describes our datasets, data generation procedures, algorithms, metrics and mis–specification protocols. Section 5 presents benchmark and sensitivity results. Section 6 discusses practical implications, limitations and recommendations for practitioners. Section **??** concludes.

2 Theoretical Framework

This chapter establishes the mathematical foundations of causal discovery. We define the probabilistic and graphical concepts necessary to reason about causality from observational data, including d-separation, the Causal Markov Condition, and Faithfulness. We then provide a detailed theoretical treatment of the four algorithms evaluated in this thesis: PC, GES, NOTEARS, and COSMO. Finally, we formalise the evaluation metrics used to benchmark these algorithms.

2.1 Probabilistic Graphical Models

A probabilistic graphical model (PGM) uses a graph-based representation to encode the conditional independence structure of a multivariate probability distribution.

2.1.1 Directed Acyclic Graphs (DAGs)

A graph $G = (V, E)$ consists of a set of vertices $V = \{X_1, \ldots, X_d\}$ and a set of directed edges $E \subseteq V \times V$. An edge $(X_i, X_j) \in E$ is denoted as $X_i \to X_j$. A path is a sequence of distinct nodes connected by edges. A directed path is a path where all edges point in the same direction. A cycle is a directed path that starts and ends at the same node. A Directed Acyclic Graph (DAG) is a directed graph with no cycles.

2.1.2 The Causal Markov Condition

The link between the graph structure G and the joint probability distribution $P(V)$ is provided by the Causal Markov Condition.

- **Local Markov Property:** Each variable X_i is conditionally independent of its non-descendants given its parents $\mathsf{Pa}(X_i)$ in G.
- **Factorisation:** If (G, P) satisfies the Markov condition, the joint distribution factorises as:

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{d} P(X_i \mid \mathsf{Pa}(X_i)) \tag{1}$$

This factorisation allows complex joint distributions to be represented compactly.

2.1.3 d-Separation

To derive all conditional independence relations implied by the Markov condition, we use the graphical criterion of d-separation (directional separation). A path p between nodes X and Y is *blocked* by a set of nodes Z if:

1. The path contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in Z.

2. The path contains a collider $i \to m \leftarrow j$ such that the collider m is not in Z and no descendant of m is in Z.

If all paths between X and Y are blocked by Z, then X and Y are d-separated given Z, denoted $X \perp_G Y \mid Z$. The global Markov property states that $X \perp_G Y \mid Z \implies X \perp_P Y \mid Z$.

2.1.4 Faithfulness

While the Markov condition allows us to infer independencies from the graph, it does not guarantee that *all* independencies in the distribution are represented in the graph. For example, parameters might cancel out exactly to create an independence not implied by the structure. The **Faithfulness Assumption** states that the distribution P contains *only* those independencies implied by the d-separation structure of G.

$$X \perp_P Y \mid Z \iff X \perp_G Y \mid Z \tag{2}$$

Faithfulness is crucial for causal discovery because it allows us to infer edges (or their absence) from observed conditional independencies.

2.1.5 Structural causal models and the Markov factorisation

A useful formal lens for the assumptions in the previous paragraph is the *structural causal model* (SCM). Let $G = (V, E)$ be a DAG with $V = \{1, \dots, d\}$ indexing random variables $X = (X_1, \dots, X_d)$. An SCM specifies a set of structural assignments

$$X_i \;:=\; f_i(X_{\mathrm{Pa}(i)}, U_i), \qquad i = 1, \dots, d, \tag{3}$$

where $\mathrm{Pa}(i)$ denotes the parents of node i in G, f_i is a (measurable) function, and $U = (U_1, \dots, U_d)$ are *exogenous* noise variables. A standard sufficient condition for the *Causal Markov condition* is that the exogenous noises are jointly independent and that the graph encodes the causal ordering of the assignments [1, 2].

From the SCM to a factorised likelihood (proof sketch). Assume G admits a topological ordering of variables such that parents precede children. By the chain rule, any joint distribution can be written as

$$p(x_1, \dots, x_d) \;=\; \prod_{i=1}^{d} p(x_i \mid x_{1:i-1}). \tag{4}$$

Under the Markov property implied by the SCM graph, each X_i is independent of its non-descendants given its parents. In particular, conditioning on all previous variables $X_{1:i-1}$ is equivalent to conditioning only on $X_{\mathrm{Pa}(i)}$, because the remaining variables in $\{1, \dots, i-1\} \setminus \mathrm{Pa}(i)$ are non-descendants of i. Therefore,

$$p(x_i \mid x_{1:i-1}) \;=\; p(x_i \mid x_{\mathrm{Pa}(i)}), \tag{5}$$

and we obtain the *Markov factorisation*

$$p(x_1, \ldots, x_d) \;=\; \prod_{i=1}^{d} p(x_i \mid x_{\mathrm{Pa}(i)}). \tag{6}$$

This identity is the key bridge between graphs and data: (6) is exactly the decomposition exploited by both score-based methods (which optimise a factorised likelihood or a penalised variant) and constraint-based methods (which test conditional independencies implied by the factorisation) [2, 3].

2.1.6 Worked d-separation queries in the Asia network

Figure 4 (the Asia network) is small enough to manually verify several conditional independencies. Below, each query is phrased as a d-separation statement in the graph, which under the Markov and faithfulness assumptions corresponds to a conditional independence in the distribution [1, 2].

Table 2: Worked examples of d-separation in the Asia network (Figure 4).

Query (graph statement)	d-separated?	Reasoning (path-wise)
`Smoking` $\perp$ `Tuberculosis`	Yes	The main connecting path `Smoking` $\rightarrow$ `LungCancer` $\rightarrow$ `Either` $\leftarrow$ `Tuberculosis` is blocked by the collider at `Either`. No other open path exists without conditioning.
`Smoking` $\perp$ `Tuberculosis` $\mid$ `Either`	No	Conditioning on the collider `Either` *opens* the previously blocked path, producing the classic "explaining-away" dependence.
`Smoking` $\perp$ `Xray` $\mid$ `Either`	Yes	`Xray` is a child of `Either`. Once `Either` is conditioned on, the path `Smoking` $\rightarrow$ `LungCancer` $\rightarrow$ `Either` $\rightarrow$ `Xray` is blocked at `Either` (a conditioned non-collider).
`Tuberculosis` $\perp$ `Bronchitis`	Yes	Two prominent paths are blocked: (i) `Tuberculosis` $\rightarrow$ `Either` $\rightarrow$ `Dyspnea` $\leftarrow$ `Bronchitis` is blocked by the collider at `Dyspnea`; (ii) the path through `Either` is blocked as above.

These concrete queries illustrate the central mechanism behind constraint-based learning: algorithms such as PC recover a skeleton by repeatedly testing whether statements like those in Table 2 are supported by the data (for various conditioning sets), and then orient edges to form an equivalence-class representation.

2.1.7 Extended d-separation examples: ALARM and Child

To further illustrate the complexity of d-separation in larger networks, we consider examples from the ALARM and Child networks. These networks contain more complex structures, including multiple paths and nested colliders, which challenge discovery algorithms.

ALARM Network Examples. The ALARM network [32] contains 37 nodes and 46 edges, representing a monitoring system for intensive care patients. Consider the relationship between `Pulmonary Embolism` (PE) and `Central Venous Pressure` (CVP).

- **Query:** $\texttt{PE} \perp \texttt{CVP}$?
- **Analysis:** There is a path $\texttt{PE} \rightarrow \texttt{PAP} \rightarrow \texttt{CVP}$ (where PAP is Pulmonary Artery Pressure). This is a chain. Thus, unconditionally, they are d-connected (dependent).
- **Query:** $\texttt{PE} \perp \texttt{CVP} \mid \texttt{PAP}$?
- **Analysis:** Conditioning on the middle node `PAP` blocks the chain. Thus, they become d-separated. This conditional independence allows PC to remove the direct edge between PE and CVP if one were hypothesised.

Consider a more complex case involving the collider at `Heart Rate` (HR).

- **Query:** $\texttt{Catecholamine} \perp \texttt{Hypovolemia}$?
- **Analysis:** They are connected via $\texttt{Catecholamine} \rightarrow \texttt{HR} \leftarrow \texttt{Hypovolemia}$. This is a v-structure (collider) at HR. Unconditionally, the path is blocked. They are d-separated.
- **Query:** $\texttt{Catecholamine} \perp \texttt{Hypovolemia} \mid \texttt{HR}$?
- **Analysis:** Conditioning on the collider `HR` opens the path. They become d-connected. This "explaining away" phenomenon is crucial for orienting the edges pointing into HR.

Child Network Examples. The Child network [**?**] models the diagnosis of "blue baby" syndrome. Consider the path between `Birth Asphyxia` and `Age`.

- **Query:** $\texttt{BirthAsphyxia} \perp \texttt{Age}$?
- **Analysis:** In the graph, these are marginally independent roots (or close to roots) with no open path connecting them unconditionally.
- **Query:** $\texttt{BirthAsphyxia} \perp \texttt{Age} \mid \texttt{Disease}$?
- **Analysis:** Both variables are ancestors of `Disease`. If `Disease` is a collider (or descendant of one) on a path between them, conditioning on it could induce dependence. In the Child network, complex interactions often make such determinations non-trivial, requiring careful path tracing.

2.1.8 Identifiability Limits and Interventional Distributions

A fundamental limitation of causal discovery from observational data is *identifiability*. As discussed in Section 2.1.11, multiple DAGs can imply the same set of conditional independencies. These DAGs form a Markov Equivalence Class (MEC).

Observational Equivalence. Two DAGs G_1 and G_2 are observationally equivalent if they have the same skeleton and the same v-structures. For example, the chain $X \to Y \to Z$, the chain $X \leftarrow Y \leftarrow Z$, and the fork $X \leftarrow Y \to Z$ are all Markov equivalent. They all imply $X \perp Z \mid Y$ and no other independencies. Without additional assumptions (like time ordering or non-Gaussianity), no algorithm can distinguish between them using only observational data $P(X, Y, Z)$.

Interventional Equivalence. Interventional data can break this symmetry. An intervention do$(Y = y)$ modifies the causal mechanism of Y.

- In $X \to Y \to Z$, intervening on Y breaks the link $X \to Y$. X and Z become independent.
- In $X \leftarrow Y \to Z$, intervening on Y breaks links from parents of Y (none here) and fixes Y. X and Z become independent (as they are effects of Y).
- In $X \leftarrow Y \leftarrow Z$, intervening on Y breaks $Z \to Y$. Z no longer influences X.

While this thesis focuses on observational discovery, understanding these limits is crucial for interpreting our results. The "Directed F_1" metric penalises algorithms for choosing the "wrong" member of an equivalence class, even if that choice was theoretically impossible to avoid. This explains the large gap between skeleton and directed performance observed in Section 5.

2.1.9 Theoretical Guarantees: Constraint-based vs Score-based

It is instructive to compare the theoretical guarantees of the two main families of algorithms evaluated.

Constraint-Based (PC).

- **Assumption:** The distribution P is faithful to a DAG G.
- **Guarantee:** In the infinite sample limit, PC recovers the true CPDAG of G.
- **Convergence:** The error probability depends on the Type I and Type II errors of the CI tests. Kalisch and Bühlmann [**?**] proved that PC is consistent for high-dimensional Gaussian graphs if the partial correlations decay sufficiently slowly (strong faithfulness) and the maximum degree is bounded. Specifically, if the significance level $\alpha_n \to 0$ at an appropriate rate, $P(\hat{G} \neq CPDAG(G)) \to 0$.

Score-Based (GES).

- **Assumption:** The scoring function is locally consistent. BIC is locally consistent for exponential families (including Gaussian and discrete).
- **Guarantee:** Chickering [3] proved that GES identifies the perfect map (true CPDAG) in the limit of large sample size.

- **Comparison:** While both are consistent, their finite-sample behaviour differs. PC is sensitive to individual test failures (error propagation). GES is more robust to local noise but can get stuck in local optima in the greedy search (though the two-phase structure mitigates this). Furthermore, score-based methods implicitly assume a specific parametric form (e.g., linear-Gaussian for BIC), whereas constraint-based methods only require a valid independence test (which can be non-parametric).

2.1.10 A minimal Markov equivalence example and the CPDAG representation

Markov equivalence can be made explicit with a three-variable skeleton $A{-}B{-}C$. Consider the two DAGs

$$A \to B \to C \qquad \text{and} \qquad A \leftarrow B \to C.$$

Both graphs have the same skeleton and neither contains a v-structure (no node has two incoming arrows from non-adjacent parents), so they entail the same set of conditional independencies (in particular, $A \perp C \mid B$) and are Markov equivalent [2]. The corresponding *completed partially directed acyclic graph* (CPDAG) therefore contains *undirected* edges $A{-}B$ and $B{-}C$, indicating that their orientations are not identifiable from observational data alone.

This example is directly relevant to the interpretation of directed metrics in later sections: an algorithm may correctly identify an edge in the skeleton while choosing an arbitrary orientation among several Markov-equivalent possibilities. In such cases, skeleton metrics can reflect the identifiable component of structure, while directed metrics penalise choices that may be unidentifiable without additional assumptions or interventional information.

2.1.11 Markov Equivalence Classes

Two DAGs are Markov equivalent if they encode the same set of conditional independence relations. Verma and Pearl [30] proved that two DAGs are Markov equivalent if and only if they have the same skeleton (underlying undirected graph) and the same set of v-structures (unshielded colliders $X \to Z \leftarrow Y$). A Markov Equivalence Class (MEC) can be represented by a Completed Partially Directed Acyclic Graph (CPDAG), where:

- Directed edges represent causal relationships common to all DAGs in the class.
- Undirected edges represent relationships that can be oriented in either direction within the class.

Observational data alone (assuming faithfulness) can only identify the CPDAG, not the unique DAG. This is the fundamental limit of observational causal discovery.

2.2 Constraint-Based Discovery: The PC Algorithm

The PC algorithm (named after Peter Spirtes and Clark Glymour) is the standard for constraint-based discovery. It relies on the faithfulness assumption to reconstruct the CPDAG.

2.2.1 Algorithm Steps

1. **Skeleton Identification:** Start with a complete undirected graph. For each pair of vertices (X, Y), test for marginal independence ($X \perp Y$). If independent, remove the edge. Then, for each remaining edge (X, Y), test for conditional independence given subsets of neighbours of size $k = 1, 2, \ldots$. If $X \perp Y \mid Z$, remove the edge and store Z as the separating set S_{XY}.
2. **Collider Identification:** For every triple $X - Z - Y$ where X and Y are not adjacent, check if $Z \in S_{XY}$. If $Z \notin S_{XY}$, then the structure must be a v-structure $X \to Z \leftarrow Y$. Orient these edges.
3. **Orientation Propagation (Meek Rules):** Apply the following rules repeatedly until no more edges can be oriented:
   - **R1:** If $X \to Y - Z$ and X, Z are not adjacent, orient as $Y \to Z$ (to avoid creating a new v-structure).
   - **R2:** If $X \to Y \to Z$ and $X - Z$, orient as $X \to Z$ (to avoid cycles).
   - **R3:** If $X - Y \to Z$ and $X - Z$ and $X - W \to Z$ and $X - W$, orient $X \to Z$ (to avoid cycles and new v-structures).

2.2.2 Complexity and Stability

The worst-case complexity is exponential in the maximum degree of the graph, as the number of conditioning sets grows combinatorially. However, for sparse graphs, PC is efficient. A key issue is **order dependence**: the order in which variables are processed can affect the output in finite samples. The PC-Stable procedure [16] is an order-independent variant of PC that produces the same adjacencies regardless of variable ordering.

2.3 Score-Based Discovery: GES

Score-based methods frame structure learning as a model selection problem. We seek the graph G that maximises a score function $S(G, D)$.

2.3.1 Scoring Functions

Common scores include the Bayesian Information Criterion (BIC) and the BDeu score.

- **BIC:** For score-based learning we use the BIC score [26], which adds a complexity penalty to the log-likelihood to avoid overfitting. $S_{BIC}(G) = \log L(G; \hat{\theta}) - \frac{d}{2} \log n$, where L is the likelihood, d is the number of parameters, and n is the sample size. BIC is consistent and decomposable.
- **BDeu:** We evaluate discrete networks with the BDeu score, a Bayesian Dirichlet score with a uniform prior equivalent sample size [27]. It satisfies score equivalence (Markov equivalent DAGs get the same score).

2.3.2 Greedy Equivalence Search (GES)

Searching the space of all DAGs is super-exponential. GES searches the space of equivalence classes (CPDAGs) using a two-phase greedy strategy:

1. **Forward Equivalence Search (FES):** Start with an empty graph. At each step, consider all single-edge additions that result in a valid CPDAG. Choose the addition that maximises the score gain. Repeat until no addition improves the score.
2. **Backward Equivalence Search (BES):** Start with the graph from FES. Consider all single-edge deletions. Choose the deletion that maximises the score. Repeat until convergence.

Chickering [3] proved that the two-phase Greedy Equivalence Search will, under appropriate assumptions and with enough data, find the true DAG that perfectly maps the generative distribution.

2.3.3 Score functions in score-based discovery: BIC and BDeu

Score-based structure learning searches for a graph G that optimises a goodness-of-fit criterion while controlling model complexity. A key property is *decomposability*: for many common likelihoods, the (log-)likelihood under G factorises over nodes exactly as in (6), yielding

$$\log p(D \mid \hat{\theta}_G, G) \;=\; \sum_{i=1}^{d} \log p(D_i \mid D_{\mathrm{Pa}(i)}, \hat{\theta}_i), \tag{7}$$

where D denotes the dataset and $\hat{\theta}_G$ are maximum-likelihood parameters under G. GES requires the score to be *score-equivalent* and *decomposable* so that it can search over equivalence classes [3].

BIC. The Bayesian Information Criterion (BIC) score is commonly written as

$$\mathrm{BIC}(G; D) \;=\; \log p(D \mid \hat{\theta}_G, G) \;-\; \frac{k_G}{2} \log n, \tag{8}$$

where n is the sample size and k_G is the number of free parameters in the model under G. For linear-Gaussian Bayesian networks, each node corresponds to a regression of X_i on its parents,

$$X_i \;=\; \beta_{i0} + \sum_{j \in \mathrm{Pa}(i)} \beta_{ij} X_j + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_i^2), \tag{9}$$

so that the local contribution to the log-likelihood depends on the residual sum of squares for that regression. The penalty term in (8) grows with the number of parameters (approximately $|\mathrm{Pa}(i)| + 2$ per node if an intercept and variance are included), discouraging dense graphs as n grows.

BDeu (discrete case). For discrete Bayesian networks, a standard Bayesian score is the Bayesian Dirichlet equivalent uniform (BDeu) score. Let r_i be the number of states of X_i and q_i the number of configurations of its parents. Writing N_{ijk} for the number of samples where $X_i = k$ and $\mathrm{Pa}(i) = j$, the BDeu score has a closed form in terms of Gamma functions and a single equivalent sample size

hyperparameter that is shared across conditional probability tables. (This score is widely used in discrete structure learning because it is score-equivalent.)

Connection to the experiments. In later experiments, GES is configured with the BIC score across all datasets (including discretised data), which intentionally stresses robustness to modelling mismatch and keeps the scoring choice uniform across domains.

2.4 Continuous Optimisation: NOTEARS

Traditionally, the acyclicity constraint (no cycles) was considered discrete and combinatorial. NOTEARS (Non-combinatorial Optimisation via Trace Exponential and Augmented lagRangian for Structure learning) reformulates this as a continuous constraint.

2.4.1 The Acyclicity Constraint

For a weighted adjacency matrix $W \in \mathbb{R}^{d\times d}$, the graph is acyclic if and only if:

$$h(W) = \mathsf{tr}(e^{W\circ W}) - d = 0 \tag{10}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential. This constraint is smooth and differentiable, allowing the use of standard nonlinear programming techniques.

2.4.2 The Optimisation Problem

NOTEARS solves:

$$\min_W \ell(W;X) + \lambda\|W\|_1 \quad \text{subject to} \quad h(W) = 0 \tag{11}$$

where $\ell(W;X)$ is a loss function (e.g., least squares for linear SEMs) and $\|W\|_1$ induces sparsity. The problem is solved using the Augmented Lagrangian method. While NOTEARS is strictly defined for linear SEMs, extensions exist for nonlinear models (using MLPs or Sobolev spaces).

2.4.3 NOTEARS acyclicity constraint: a differentiable characterisation of DAGs

A central difficulty in optimisation-based discovery is enforcing that the learned adjacency matrix corresponds to an acyclic graph. NOTEARS introduces a smooth function $h(\cdot)$ such that $h(W) = 0$ if and only if the weighted adjacency matrix $W \in \mathbb{R}^{d\times d}$ represents a DAG [19]. Define the elementwise (Hadamard) product by $(A \circ B)_{ij} = A_{ij}B_{ij}$ and set $M := W \circ W$. NOTEARS uses

$$h(W) \;:=\; \mathrm{tr}(\exp(M)) - d. \tag{12}$$

Intuitively, the matrix exponential $\exp(M)$ expands into a power series whose trace aggregates weighted counts of closed walks; if and only if there are no directed cycles does the trace reduce to d.

Augmented Lagrangian objective. For a linear SEM with squared loss, NOTEARS solves (schematically)

$$\min_{W} \ \mathcal{L}(W) + \lambda_1 \|W\|_1 \quad \text{s.t.} \quad h(W) = 0, \tag{13}$$

often via an augmented Lagrangian of the form

$$\min_{W} \ \mathcal{L}(W) + \lambda_1 \|W\|_1 + \frac{\rho}{2} h(W)^2 + \alpha \, h(W), \tag{14}$$

where ρ and α are iteratively updated to enforce acyclicity.

Gradient of the acyclicity penalty. Because $h(\cdot)$ is smooth, first-order methods can be used. A convenient identity is

$$\nabla_W h(W) \ = \ 2W \circ (\exp(W \circ W))^\top, \tag{15}$$

obtained by applying the chain rule to (12): the derivative of $\mathrm{tr}(\exp(M))$ with respect to M is $(\exp(M))^\top$, and $\partial(W \circ W)/\partial W = 2W$ elementwise. This explicit gradient highlights why NOTEARS is computationally attractive: acyclicity becomes a differentiable constraint rather than a combinatorial one, albeit at the cost of relying on the correctness of the underlying functional assumptions (here, a linear SEM).

2.5 Regression-Based Discovery: COSMO

Massidda et al. [23] introduce COSMO, a continuous DAG learning method that encodes acyclicity via a differentiable 'priority' function; this makes the algorithm $O(n^2)$ in the number of nodes (instead of cubic), while maintaining competitive accuracy.

2.5.1 Smooth Orientation

COSMO assigns a latent scalar θ_i to each node X_i, representing its position in a topological ordering. An edge $X_i \to X_j$ is allowed only if $\theta_i < \theta_j$. This is enforced via a smooth penalty function.

2.5.2 Algorithm Logic

COSMO iterates between:

1. **M-Step:** Given the ordering Θ, estimate the parents of each node using regularised regression (Lasso), penalising edges that violate the ordering.
2. **O-Step:** Update the ordering Θ to minimise the loss given the current edges.

To improve robustness, COSMO uses **stability selection**: the algorithm is run on many bootstrap resamples, and edges are kept only if they appear in a high fraction of runs. This provides a natural measure of edge confidence.

2.6 Evaluation Metrics

To benchmark these algorithms, we compare the learned graph $\hat{G}$ to the ground truth G.

2.6.1 Structural Hamming Distance (SHD)

We evaluate accuracy by **Structural Hamming Distance (SHD)**, the number of edge additions/deletions or reversals needed to transform the learned graph into the true graph [17].

$$\text{SHD}(G, \hat{G}) = |E \setminus \hat{E}| + |\hat{E} \setminus E| + \text{flips} \tag{16}$$

For CPDAGs, we must be careful not to penalise valid Markov equivalent differences. We use a CPDAG-aware SHD that treats undirected edges in the equivalence class as matching if the skeleton is correct.

2.6.2 Structural Intervention Distance (SID)

We also report the **SID** (Structural Intervention Distance), which counts differences in causal predictions between the estimated and true DAG [25].

2.6.3 Precision, Recall, and F1

- **Precision:** The fraction of predicted edges that are true. $P = \frac{TP}{TP+FP}$
- **Recall:** The fraction of true edges that are predicted. $R = \frac{TP}{TP+FN}$
- **F1 Score:** The harmonic mean of precision and recall. $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$

We compute these metrics for both the **skeleton** (ignoring direction) and the **directed graph**. The gap between skeleton and directed performance highlights the difficulty of orientation.

2.7 Structural Causal Models and Interventions

While this thesis focuses on *structure learning* (recovering a causal graph), it is helpful to connect DAGs to the structural causal model (SCM) view, because that perspective clarifies what structure learning can (and cannot) tell us about interventions.

In an SCM, each observed variable X_i is generated by a structural equation $X_i = f_i(\mathrm{Pa}_i, U_i)$, where Pa_i denotes the parents of X_i in the DAG and U_i is an exogenous noise term. Under standard assumptions (e.g., independent U_i), the joint observational distribution factorizes according to the DAG: $P(X_1, \ldots, X_p) = \prod_i P(X_i \mid \mathrm{Pa}_i)$.

Interventions are represented by modifying structural equations. For example, an intervention $\mathrm{do}(X_k = x)$ replaces the structural equation for X_k with the constant assignment $X_k := x$, producing an interventional distribution $P(\cdot \mid \mathrm{do}(X_k = x))$. Pearl's do–calculus [1] provides graphical rules for reasoning about such interventions.

In practice, many applied questions involve estimating causal effects rather than only learning a graph. However, a learned graph is often the first step: it defines candidate adjustment sets, highlights potential confounding, and informs what additional data (e.g., interventions) would reduce uncertainty. Our emphasis in this thesis is therefore on (i) benchmarking how reliably different algorithms recover known benchmark structures and (ii) understanding how the data can flag a mismatch between an analyst's assumed graph and the data–generating process.

2.8 Assumptions, identifiability, and scope

Structure learning from observational data typically relies on three broad categories of assumptions.

Graphical assumptions. The causal Markov condition and faithfulness connect graph separation to statistical conditional independence (Section 2.1.3). Many benchmarks additionally assume *causal sufficiency*—that there are no unmeasured common causes of the measured variables. Violations of these assumptions can lead to spurious edges, missing edges, or incorrect orientations.

Statistical assumptions. Constraint–based methods require valid conditional independence (CI) tests; score–based methods require a scoring criterion aligned with the data type and sample size. For example, Fisher's z test assumes approximately Gaussian continuous variables, while the chi–square test assumes categorical variables with sufficiently large expected counts.

Model class assumptions. Many modern methods assume specific functional forms, such as linear–Gaussian relationships (NOTEARS) or sparsity and linear regressions (COSMO). When the data–generating process violates these assumptions, performance can degrade even if the underlying graph structure is correct.

This thesis deliberately uses semi–synthetic data with known ground truth and no latent confounding, so that algorithm behaviour can be studied in a controlled setting. Real observational datasets can violate these conditions substantially; we discuss this external validity gap in Section 6.5.

2.9 Conditional independence testing in this benchmark

Conditional independence tests appear in two roles in this thesis. First, they are the core primitive of constraint–based discovery (PC): the algorithm iteratively tests whether two variables are independent given an increasing conditioning set, removing edges when independence cannot be rejected. Second, CI tests can be used directly for *model criticism*: if an analyst's DAG implies $X \perp Y \mid Z$ but the data strongly reject that hypothesis, the assumed graph is inconsistent with the observations.

For continuous data we use Fisher's z test on partial correlations. Let $r_{XY \cdot Z}$ be the sample partial correlation between X and Y given Z. Under the null hypothesis $H_0 : X \perp Y \mid Z$ in a multivariate Gaussian model, the statistic

$$z = \tfrac{1}{2} \ln\!\left(\tfrac{1+r_{XY \cdot Z}}{1-r_{XY \cdot Z}}\right) \sqrt{n - |Z| - 3} \tag{17}$$

is approximately standard normal. For discrete data we use a chi–square test on contingency tables, where the null is that the conditional distribution factorizes as $P(X, Y \mid Z) = P(X \mid Z) P(Y \mid Z)$.

In both cases we use a nominal significance level $\alpha = 0.05$ in the benchmark configuration. In PC, α controls a precision–recall trade–off: smaller α yields a more conservative skeleton (higher precision,

lower recall), while larger α tends to retain more edges (higher recall, more false positives). When using CI tests for model criticism, analysts should also consider multiple testing and the possibility that several CI statements may be violated simultaneously.

2.10 Statistical tests for comparing algorithms

Benchmarking studies often compare multiple algorithms across multiple datasets. Because performance measures (e.g., F_1) may not be normally distributed and the datasets form paired conditions (each algorithm is evaluated on the same set of benchmark networks), a common recommendation is to compare average ranks using non–parametric procedures.

For comparing multiple classifiers over many datasets, Demšar [24] recommends using a Friedman test (non-parametric two-way ANOVA by ranks) and then a Nemenyi post-hoc test to find significant pairwise differences, visualized via 'critical difference' diagrams. The CD identifies how far apart two average ranks must be to claim a statistically significant difference under the Nemenyi post–hoc test. Given that our benchmark includes only five networks, the power of these tests is limited; we treat them primarily as descriptive and emphasize per–dataset effects alongside ranks.

2.10.1 Power and multiple-testing considerations for CI-driven discovery

Constraint-based discovery relies on repeated conditional independence tests. Even when a CI test is well-specified for the data type, two statistical issues matter in practice.

Power depends on conditioning-set size. For tests based on partial correlations (e.g., Fisher-Z), the effective sample size decreases with the size of the conditioning set $|S|$. A common approximation treats the Z-transformed partial correlation as approximately normal with variance scaling as $(n - |S| - 3)^{-1}$. As $|S|$ grows, the test rapidly loses power unless n is large, which helps explain why larger graphs with hubs can be challenging: hubs induce larger candidate conditioning sets and therefore weaker tests, increasing the chance of both false negatives (missing edges) and false positives (if the algorithm compensates by retaining edges due to inconclusive tests).

Multiple testing and error propagation. PC-style procedures perform many tests and then commit to adjacency updates that affect which later tests are performed. This creates an error-propagation phenomenon: an early false decision can change the conditioning sets available later. While the benchmark reports end-to-end graph accuracy, analyst-facing deployments should additionally examine the stability of *specific* edges, for example via bootstrap resampling (as in Section 5.4) or stability-selection style summaries.

2.10.2 Robustness checks and ablations (planned analyses)

Several robustness checks are natural extensions of this work and can be conducted without changing the core benchmark design:

- **Sample-size sensitivity.** Repeating the benchmark for multiple n values can reveal whether an algorithm's apparent superiority is asymptotic or depends on a particular data regime.
- **CI-test ablations.** Holding PC fixed while swapping CI tests (where meaningful) can isolate whether performance differences come from the search procedure or the statistical test.
- **Thresholding sensitivity for optimisation methods.** For NOTEARS-like outputs, varying the edge threshold can expose whether the learned weights separate "signal" from "noise" or whether performance is dominated by a fragile operating point.

These checks strengthen the link between benchmark metrics and the operational reliability required in analyst workflows.

3 Related Work

Causal discovery has evolved from a niche topic in philosophy and statistics to a central pillar of modern machine learning. This chapter reviews the intellectual history of the field, surveys critical benchmarking studies that motivate our work, and examines the growing literature on model validation and mis–specification diagnostics.

3.1 Foundations of Causal Discovery

The formalisation of causal discovery rests on two parallel intellectual traditions that converged in the 1990s: the probabilistic graphical models framework of Judea Pearl and the constraint–based search methods of Spirtes, Glymour, and Scheines.

3.1.1 The Graphical Viewpoint

Pearl [1] introduced the Directed Acyclic Graph (DAG) as a rigorous language for causality, moving beyond the informal path diagrams of Wright. Central to this framework is the *d-separation* criterion, which connects the topological structure of the graph to the conditional independence structure of the data. This link allows researchers to test causal claims empirically: if a graph implies that $X \perp Y \mid Z$ but the data show a strong dependence, the graph is falsified. Pearl's *Ladder of Causation* distinguishes three levels of reasoning: association (seeing), intervention (doing), and counterfactuals (imagining). Structure learning primarily operates at the first level (using associations to infer structure) to enable reasoning at the second level (predicting interventions).

3.1.2 The Search and Score Viewpoint

While Pearl focused on the semantics of graphs, Spirtes, Glymour, and Scheines [2] developed practical algorithms for discovering them. Their PC algorithm showed that under assumptions of *Causal Markovness* and *Faithfulness*, it is possible to recover the causal structure (up to a Markov equivalence class) from observational data efficiently (in the limit of infinite data and assuming

valid independence tests). This work challenged the prevailing dogma that "correlation does not imply causation," demonstrating that while correlation alone is insufficient, *patterns* of conditional independence across many variables can indeed identify causal directionality (e.g., via v-structures).

3.1.3 A brief historical arc of structure learning paradigms

While the core assumptions of causal discovery (Markov, faithfulness, and adequate functional form) remain stable, the *algorithmic* landscape has expanded substantially over time. A coarse but useful organising principle is the dominant source of information used to infer edges:

- **Constraint-based discovery** infers the skeleton and (partially) the directions by testing conditional independence statements implied by a graph. The PC algorithm and its variants are canonical representatives, with extensions such as FCI targeting settings with latent confounding [2, 16].
- **Score-based discovery** searches for a graph that optimises a decomposable score trading off likelihood and complexity. GES is a prominent example that searches over Markov equivalence classes under score equivalence and decomposability [3].
- **Hybrid discovery** combines constraint-based pruning with score-based refinement; MMHC is a widely used representative [17].
- **Functional** / **distributional discovery** exploits stronger assumptions (e.g., non-Gaussianity) to identify directions that are not identifiable under purely conditional-independence semantics; LiNGAM is a classical example in the linear, non-Gaussian setting [18].
- **Optimisation- and learning-based discovery** (2018 onwards) re-casts DAG learning as continuous optimisation (e.g., NOTEARS) or leverages neural parameterisations and reinforcement learning to search large spaces efficiently [19, 20, 21, 22, 23].

Table 3: A sketch timeline of representative developments in causal structure learning. This is not exhaustive; it highlights the conceptual shift from combinatorial search and CI testing to differentiable and learning-based approaches.

Period	Representative developments
1990s	Constraint-based discovery formalised and popularised (e.g., PC/FCI) under Markov and faithfulness assumptions [2].
2000s	Score-based and hybrid methods mature (e.g., GES, MMHC), with practical software implementations becoming widely used [3, 17].
2010s	Tooling and benchmarking proliferate (e.g., `bnlearn`, `pcalg`), bringing attention to robustness and evaluation design [12, 11].
2018–present	Differentiable and learning-based approaches accelerate (NOTEARS, GOLEM, GraN-DAG, RL-based search) and new methods explicitly target practical fragilities such as orientation instability [19, 20, 21, 22, 23].

3.1.4 Algorithm families, assumptions, and outputs

Table 4 summarises representative methods across families, emphasising the assumptions that most directly interact with analyst misspecification: the validity of CI tests, score-model alignment, and functional-form fit.

Table 4: Representative causal discovery methods and the assumptions most relevant to misspecification robustness. "Output" refers to the standard graphical object returned when run on observational data.

Method	Family	Typical assumptions (most salient)	Output and practical notes
PC [2]	Constraint	Markov + faithfulness; CI test is well-calibrated for the data type; sufficient n	CPDAG (equivalence class). Sensitive to CI-test misspecification and multiple testing.
PC-Stable [16]	Constraint	As PC, but designed to reduce order-dependence	CPDAG. Often a safer default than naive PC in software implementations.
FCI [2]	Constraint	Allows latent confounding under additional assumptions; more complex independence logic	PAG. Evaluation must respect the weaker target object (not a DAG).
GES [3]	Score	Decomposable, score-equivalent score; score model matches data; sufficient n	CPDAG. Can be accurate but computationally heavy in larger graphs.
MMHC [17]	Hybrid	Reliable CI-based neighbourhood discovery; refinement by a score	DAG. Practical compromise between CI testing and scoring.
LiNGAM [18]	Functional	Linear, non-Gaussian noise; typically no latent confounders	DAG. Can identify directions not identifiable under Gaussian CI tests.
NOTEARS [19]	Optimisation	Linear SEM (in the basic form); continuous optimisation + acyclicity constraint	Weighted adjacency, thresholded to a DAG. Sensitive to mismatch for discrete/binned data.
GOLEM [20]	Optimisation	Likelihood-based continuous optimisation under specific SEM likelihoods	DAG. Emphasises likelihood modelling and regularisation.
GraN-DAG [21]	Neural	Flexible nonlinear SEM parameterisation; optimisation stability	DAG. Higher capacity but introduces additional hyperparameters and optimisation variance.
RL-based search [22]	RL/score	Score model fidelity; enough signal to guide policy search	DAG. Flexible but can be compute-intensive and sensitive to reward design.
COSMO [23]	Optimisation	Differentiable orientation and stability heuristics; practical thresholds	DAG. Designed to stabilise discovery, but may trade accuracy for conservatism.

3.1.5 Positioning of this thesis within the literature

Recent critiques of benchmarking practice emphasise that headline accuracy can be inflated by favourable synthetic generators, metric choices that ignore equivalence classes, or inadvertent

leakage of ground-truth structure into evaluation design [6, 5]. This thesis positions itself at the intersection of *method comparison* and *workflow robustness*: rather than proposing a new discovery algorithm, it empirically characterises how common families behave under a controlled form of analyst misspecification and uses sensitivity and stability diagnostics to motivate practitioner-facing guidance.

3.2 The Benchmarking Crisis in Causal Discovery

As new algorithms proliferated, the field faced a challenge: how to reliably compare them? Early evaluations often relied on small, custom simulations that failed to capture the complexity of real-world data.

3.2.1 Standardisation Efforts

In an extensive benchmark, Scutari et al. [5] found that constraint-based methods were often less accurate than score-based ones and seldom faster, while hybrids offered no clear advantage – implying no single algorithm type dominated overall performance. Tsamardinos et al. [17] showed that their hybrid Max–Min Hill-Climbing (MMHC) algorithm consistently outperformed representative constraint-based (PC) and score-based (GES) algorithms in reconstruction quality and often in speed. Their work established the importance of using standard metrics like Structural Hamming Distance (SHD) and reporting runtimes, practices we adopt in this thesis.

3.2.2 The "Scale-Free" Trap

Reisach et al. [6] show that in common simulated benchmarks, variables' variances often increase in causal order, a property ('varsortability') that allows simple methods to achieve unexpectedly high accuracy. In many standard benchmarks, variables further down the causal chain tend to have higher variance (accumulation of noise). Algorithms that simply sorted variables by variance could achieve state-of-the-art performance without learning any causal mechanisms. This "variance sorting" heuristic fails completely if data is standardised, revealing that many "advances" were illusory. This finding motivates our decision to include both standardised (NOTEARS) and non-standardised evaluations, and to focus on discrete data where variance scaling is less trivial.

3.3 Model Criticism and Mis-specification

While discovery algorithms aim to find the "best" graph, a parallel stream of research asks: how do we know if a given graph is "good enough"?

3.3.1 Global vs. Local Fit

Traditional model selection relies on global scores like BIC or BDeu. However, a graph can have a good global score while missing a critical causal link. Textor et al. [8] implemented routines (in DAGitty) to test every conditional independence implied by a DAG; any *significant violation* indicates

the DAG is misspecified. Ankan et al. [7] further demonstrated that these tests can pinpoint likely missing or spurious edges – even a single violated independence can reveal a structural error. Our work builds directly on this philosophy, systematically testing whether these local checks can detect specific analyst errors (missing or spurious edges).

3.3.2 Refutation and Stability

Beyond independence testing, recent work in causal inference (e.g., the `DoWhy` library by Sharma and Kiciman) emphasises *refutation*: adding random common causes, replacing data with a placebo, or subsetting data to check if causal estimates are robust. In structure learning, this concept translates to *stability selection* (Meinshausen and Bühlmann), which we employ via bootstrap resampling. Friedman et al. [10] introduced a bootstrap approach to assess the confidence of learned network features, such as the reliability of an edge. Scutari and Nagarajan [9] developed statistical techniques to identify significant edges in learned graphs – for instance, by bootstrapping the data and retaining only edges that appear above a chosen frequency cutoff. If an edge appears in only 50% of bootstrap resamples, it is likely an artifact of noise rather than a true mechanism.

3.4 The Differentiable Discovery Revolution

Zheng et al. [19] introduced a smooth, exact formulation of the acyclicity constraint for DAGs, turning structure learning into a continuous optimization problem solvable with standard gradient-based methods. By reformulating the combinatorial acyclicity constraint as a continuous equality constraint, they opened the door to using standard gradient-based optimisation tools (like PyTorch and TensorFlow) for structure learning.

3.4.1 Extensions and Limitations

This breakthrough led to a flurry of extensions:

- **Nonlinearity:** Lachapelle et al. [21] extended the framework to nonlinear relationships using neural networks (GraN-DAG).
- **Robustness:** Ng et al. [20] introduced GOLEM, which relaxes the hard constraints to improve convergence and robustness to initialisation.
- **Scalability:** Zhu et al. [22] applied reinforcement learning to search the graph space, treating edge addition/removal as actions.

However, these methods often require careful hyperparameter tuning (e.g., the penalty strength λ) and can be sensitive to data scaling (as noted by Reisach). Our benchmark includes NOTEARS to assess whether this "revolution" translates to reliable performance on classic discrete networks, a setting often overlooked in the deep learning literature.

3.5 Software Ecosystem

The gap between theory and practice is often bridged by software.

- **R Ecosystem:** The `bnlearn` package by Scutari [12] and `pcalg` by Kalisch et al. [11] have long been the gold standards, offering robust implementations of PC, GES, and HC.
- **Python Ecosystem:** Python has recently caught up with libraries like `causal-learn` (a port of the Tetrad Java library) [14] and `gCastle` (Huawei's toolkit).
- **Interactive Tools:** `DAGitty` and `CausalWizard` provide GUIs for domain experts, emphasising the "human-in-the-loop" aspect.

This thesis leverages the Python ecosystem (`causal-learn`, `CausalNex`) to provide a modern, reproducible benchmarking pipeline.

3.6 Positioning of this Thesis

Most prior benchmarks focus on the *algorithmic* race: which method gets the highest F1 score? This thesis shifts the focus to the *analyst*. In the real world, algorithms are rarely run in isolation; they are used to assist human experts.

1. **Analyst-Centric Evaluation:** We evaluate not just raw recovery, but whether data-driven signals (CI tests) can correct human errors.
2. **Holistic Comparison:** We compare the "old guard" (PC, GES) with the "new wave" (NOTEARS, COSMO) on a level playing field of standard discrete networks.
3. **Reproducibility:** By providing a complete Python pipeline, we aim to lower the barrier for practitioners to benchmark methods on their own data.

4 Methods

This chapter details the experimental design, including the benchmark datasets, the specific algorithm implementations, the evaluation metrics, and the protocols for simulating and detecting analyst misspecification.

4.1 Datasets and Data Generation

Our experiments use five canonical Bayesian network benchmarks summarised in Table 5. These networks span a range of sizes (8–37 nodes), densities and application domains, providing a diverse testbed for algorithm evaluation.

4.1.1 Implemented synthetic generation: linear SEM with quantile discretisation

The discrete benchmark datasets in this repository (Asia, ALARM, Child, and Insurance) are *semi-synthetic* in the following precise sense: the *graph structure* is taken from the canonical benchmark networks, but the *sampled data* are generated by a linear structural equation model (SEM) with independent Gaussian noise and then discretised variable-wise to match the intended cardinalities. This design yields a controlled setting where the true causal graph is known while allowing the sample size n to be set uniformly across datasets.

Concretely, in a topological order consistent with the ground-truth DAG, each node is generated as

$$X_i \;=\; \sum_{j \in \mathrm{Pa}(i)} w_{ji} X_j + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0,1), \tag{18}$$

where edge weights w_{ji} are sampled once from a bounded distribution (with random sign) and then held fixed for data generation. To obtain discrete observations with a specified number of states r_i, the continuous samples are transformed into

$$\tilde{X}_i \;=\; Q_i(X_i), \tag{19}$$

where $Q_i(\cdot)$ is a quantile-based binning operator that maps X_i into $\{0, 1, \ldots, r_i - 1\}$ using empirical quantile cutpoints. Intuitively, this preserves the *rank information* induced by the linear SEM while producing a discrete-valued dataset suitable for discrete CI tests.

Why this matters. This generator is intentionally simple and reproducible, but it also introduces an important nuance for interpretation: although the networks are treated as discrete in the benchmarking code, the underlying data-generating mechanism is a discretised linear-Gaussian SEM. As a result, algorithms that assume linear relations may behave differently than they would under a genuinely discrete Bayesian-network generator.

Discretization Procedure. To ensure transparency, we provide the exact procedure used to discretise the continuous samples. Let $X \in \mathbb{R}^{n \times d}$ be the data matrix generated by the linear SEM. For each variable $j \in \{1, \ldots, d\}$ with desired cardinality k_j:

1. Compute the empirical quantiles $q_0, q_1, \ldots, q_{k_j}$ of column $X_{\cdot,j}$, where $q_0 = \min(X_{\cdot,j})$ and $q_{k_j} = \max(X_{\cdot,j})$.
2. Map each value x_{ij} to a discrete category $c_{ij} \in \{0, \ldots, k_j - 1\}$ such that $q_{c_{ij}} \leq x_{ij} < q_{c_{ij}+1}$.

This quantile binning ensures that the marginal distribution of each discrete variable is approximately uniform, maximising the entropy and avoiding "rare category" issues that can destabilise CI tests.

4.1.2 Determinism and dataset reuse

To support exact reproducibility, the benchmark datasets are stored as CSV files under `causal_benchmark/data/`. Unless forced regeneration is enabled, experiments load these fixed datasets rather than resampling

```
Algorithm 1: Data Generation and Discretization
Input: DAG G, Sample size n, Cardinalities K
Output: Discrete data matrix D
1.  Initialize continuous matrix X of size (n, |V|)
2.  TopologicalSort(G) -> order
3.  For each node i in order:
4.      parents = Parents(G, i)
5.      weights = RandomUniform(-1, 1, size=|parents|)
6.      noise = Normal(0, 1, size=n)
7.      X[:, i] = X[:, parents] @ weights + noise
8.  For each node i:
9.      k = K[i]
10.     bins = Quantiles(X[:, i], k)
11.     D[:, i] = Digitize(X[:, i], bins)
Return D
```

Figure 1: Pseudocode for the semi-synthetic data generation process.

at each run. This choice ensures that differences across algorithms are attributable to the learning procedures rather than stochastic variation in the sampled dataset, but it also means that the benchmark section reports performance on a single realised sample per dataset (for the chosen n).

Table 5: Summary of benchmark networks. Edges refers to the number of directed edges in the ground–truth DAG. Density is $|E|/(|V|(|V|-1)/2)$, the fraction of possible edges present. Sample size n indicates the number of observations generated for each experiment.

Network	Nodes	Edges	Density	Data Type	n	Domain
Asia	8	8	0.29	Discrete	1000	Medical diagnosis
Sachs	11	17	0.31	Continuous	1000	Protein signalling
Child	20	25	0.13	Discrete	1000	Paediatric diagnosis
Insurance	27	52	0.15	Discrete	1000	Risk assessment
ALARM	37	46	0.07	Discrete	1000	ICU monitoring

The networks were chosen to cover diverse structural properties. Asia [31] is a small, dense network commonly used as a “sanity check” for causal discovery algorithms. The Sachs benchmark encodes a protein signalling system and is widely used in causal discovery evaluations; although it is historically associated with interventional studies, in this thesis we treat it as an observational benchmark and generate synthetic continuous samples from its graph structure. ALARM [32] is a large, sparse network originally developed for medical monitoring systems and is often considered challenging due to its size. Child and Insurance provide intermediate complexity with moderate node counts and edge densities.

4.2 Algorithms and Settings

We evaluate four causal discovery algorithms representing different methodological approaches: constraint–based (PC), score–based (GES), continuous optimisation (NOTEARS) and regression–based (COSMO). Table 6 summarises each algorithm’s key characteristics and hyperparameter settings.

Table 6: Algorithm implementations and hyperparameter settings. CI = conditional independence test. COSMO was run with a timeout of 120 seconds per dataset; PC, GES and NOTEARS were allowed to run to completion.

Algorithm	Type	Implementation	Key Settings
PC	Constraint	causal–learn	CI test: χ^2 (discrete), Fisher-z (continuous); $\alpha = 0.05$
GES	Score	causal–learn	Score: BDeu (discrete), BIC (continuous); equivalent sample size = 10 (BDeu)
NOTEARS	Optimisation	CausalNex	Threshold: 0.1 (continuous), 0.25 (discrete); other parameters: CausalNex defaults
COSMO	Regression	numpy/networkx	$n_{\text{restarts}} = 25$; auto–λ via BIC; edge threshold = 0.08

4.2.1 Hyperparameter Selection Protocol

To ensure a fair comparison, we adopted a standardised protocol for hyperparameter selection.

- **PC:** The significance level α controls the sparsity of the graph. We selected $\alpha = 0.05$ as the standard convention in the literature [2]. We performed a sensitivity analysis (Section 5.3) to confirm that this choice yields a reasonable balance between precision and recall.
- **GES:** The primary hyperparameter is the penalty weight in the BIC/BDeu score. For BIC, the penalty is fixed at $\frac{\log n}{2}$. For BDeu, the equivalent sample size (ESS) represents the weight of the prior. We used ESS=10, a common default that avoids over-smoothing the data while preventing zero-count issues.
- **NOTEARS:** The L_1 penalty λ and the edge threshold w_{thresh} are critical. We used the defaults provided by `CausalNex` ($\lambda = 0.05$) but tuned the threshold. We observed that for discrete data, NOTEARS produces many weak edges; thus, we increased the threshold to 0.25 (from the default 0.1) to improve precision.
- **COSMO:** We used the automatic λ selection feature based on BIC, which sweeps a path of regularisation parameters and selects the one minimising the BIC score. The stability threshold was set to 0.08 based on the authors' recommendations.

4.2.2 Computational Environment

All experiments were conducted on a MacBook Pro with an Apple M2 Pro chip and 16GB of RAM. The software environment was managed via Conda. Key library versions include:

- Python 3.10
- `causal-learn` 0.1.3.6 (PC, GES)
- `CausalNex` 0.12.0 (NOTEARS)

- `numpy` 1.23.5, `pandas` 1.5.3, `networkx` 2.8.8
- `scikit-learn` 1.2.2 (COSMO)

To ensure fair runtime comparisons, all algorithms were run sequentially on a single core (where possible), although NOTEARS and COSMO leverage vectorisation which may utilise multiple cores implicitly.

Python Version Constraint: NOTEARS via CausalNex requires Python < 3.11 due to library compatibility constraints. This constraint is explicitly enforced in the implementation to ensure reproducibility.

4.2.3 Mapping algorithm settings to the implementation

All algorithm invocations are controlled by a single configuration file (`causal_benchmark/experiments/config.yaml`) and thin wrapper modules under `causal_benchmark/algorithms/`. The wrappers serve two purposes: (i) they map a common `pandas` data matrix into the specific library API required by each algorithm; and (ii) they normalise outputs into a consistent adjacency-matrix representation for evaluation.

Two implementation details are particularly relevant for later interpretation:

- **CI tests are matched to data type.** The PC wrapper selects Fisher's Z for continuous data and a χ^2 test for discrete data, aligning the conditional-independence test with the dataset type used in evaluation.
- **GES score choice is explicit.** The GES wrapper uses the score specified in the configuration (BIC in this thesis) rather than automatically switching scores by data type. This choice provides a controlled way to study how a uniform score behaves across heterogeneous datasets.

4.2.4 Runtime measurement and parallel execution

Runtimes reported in Table 15 are measured wall-clock times for each algorithm call on each dataset, using a start/stop timer around the learning function. The experiment driver supports parallel execution over dataset–algorithm pairs; however, each *individual* run (a single algorithm on a single dataset) is timed serially, so reported runtimes correspond to the effective compute cost experienced by a user for that call.

4.2.5 Reproducibility artefacts

For each dataset–algorithm pair, the benchmark script writes: (i) the predicted adjacency matrix, (ii) a human-readable edge list, and (iii) a machine-readable diff object containing the sets of extra, missing, and reversed edges. These artefacts enable post-hoc analyses beyond aggregate metrics, including the error decompositions reported in Section 5.

4.2.6 PC Algorithm

The Peter–Clark (PC) algorithm is the archetypal constraint–based method. It assumes faithfulness and causal sufficiency to recover the Markov equivalence class of the underlying DAG.

Core Mechanism: PC starts with a complete undirected graph and iteratively removes edges (X, Y) if a separating set Z can be found such that $X \perp Y \mid Z$. The algorithm proceeds by level k, where k is the size of the conditioning set $|Z|$.

1. **Skeleton Discovery:** For $k = 0, 1, \ldots$, test conditional independence for all adjacent pairs. If $p > \alpha$, remove the edge and store Z as a separating set.
2. **Orientation:** Identify v–structures $X - Z - Y$ where X, Y are not adjacent. If Z is not in the separating set of (X, Y), orient as $X \to Z \leftarrow Y$.
3. **Propagation:** Apply Meek rules to orient remaining undirected edges without creating cycles or new v–structures.

```
Algorithm 2: PC Algorithm
Input: Data D, Significance level alpha
Output: CPDAG G
1.  Form complete undirected graph G on V
2.  k = 0
3.  Repeat until k > max_degree(G):
4.      For each edge (X, Y) in G:
5.          For each subset Z of Adj(X) \ {Y} with |Z| = k:
6.              p_val = CI_Test(X, Y, Z, D)
7.              If p_val > alpha:
8.                  Remove edge (X, Y)
9.                  SepSet(X, Y) = Z
10.                 Break (move to next edge)
11.     k = k + 1
12. For each triple X - Z - Y with X, Y non-adjacent:
13.     If Z not in SepSet(X, Y):
14.         Orient X -> Z <- Y
15. Apply Meek Rules (R1-R3) to orient further edges
Return G
```

Figure 2: Pseudocode for the PC algorithm.

Implementation Details: We use the `causal-learn` implementation with `stable=True` to ensure order–independence. For discrete data (Asia, Alarm, Child, Insurance), we use the χ^2 test. For continuous data (Sachs), we use Fisher's z–test. The significance level is fixed at $\alpha = 0.05$.

Complexity and Sensitivity: In the worst case (dense graphs), PC is exponential in the number of nodes V. However, for sparse graphs with bounded degree, it is polynomial. PC is sensitive to Type II errors (failing to reject independence) early in the search, which can erroneously remove edges that are needed to condition on later.

4.2.7 Greedy Equivalence Search (GES)

GES is a score–based method that searches over the space of equivalence classes (CPDAGs) rather than individual DAGs, which makes it statistically consistent in the large–sample limit.

Core Mechanism: GES maximises a decomposable score (like BIC or BDeu) via a two–phase greedy search:

1. **Forward Phase:** Start with an empty graph. Iteratively add the single edge that maximally increases the score until no addition improves it.
2. **Backward Phase:** Iteratively remove the single edge that maximally increases the score until no removal improves it.

```
Algorithm 3: Greedy Equivalence Search (GES)
Input: Data D, Score function S
Output: CPDAG G
1.  G = EmptyGraph(V)
2.  # Forward Phase
3.  Loop:
4.      BestScore = -Infinity, BestOp = None
5.      For each valid edge addition E+ in CPDAG space:
6.          Gain = S(G + E+, D) - S(G, D)
7.          If Gain > BestScore:
8.              BestScore = Gain, BestOp = E+
9.      If BestScore > 0:
10.         G = Apply(G, BestOp)
11.     Else:
12.         Break
13. # Backward Phase
14. Loop:
15.     BestScore = -Infinity, BestOp = None
16.     For each valid edge deletion E- in CPDAG space:
17.         Gain = S(G - E-, D) - S(G, D)
18.         If Gain > BestScore:
19.             BestScore = Gain, BestOp = E-
20.     If BestScore > 0:
21.         G = Apply(G, BestOp)
22.     Else:
23.         Break
Return G
```

Figure 3: Pseudocode for the GES algorithm.

Implementation Details: We use `causal-learn`'s GES with the BIC score uniformly across all datasets (discrete and continuous). This design choice intentionally tests GES robustness to score misspecification: while BDeu is theoretically optimal for discrete networks, using BIC throughout provides a controlled assessment of how a single scoring function performs across heterogeneous data types.

Complexity and Sensitivity: GES is generally computationally intensive, as it must re–evaluate

scores for many candidates at each step. It is less sensitive to individual hypothesis test failures than PC but can be prone to overfitting if the score penalty is too weak.

4.2.8 NOTEARS

NOTEARS (Non–combinatorial Optimisation via Trace Exponential and Augmented lagRangian for Structure learning) reformulates the discrete acyclicity constraint into a continuous equality constraint.

Core Mechanism: It solves the optimisation problem:

$$\min_{W} \ell(W; X) + \lambda \|W\|_1 \quad \text{subject to} \quad h(W) = \text{tr}(e^{W \circ W}) - d = 0 \tag{20}$$

where W is the weighted adjacency matrix, ℓ is a loss function (typically least–squares), and $h(W) = 0$ ensures acyclicity.

Implementation Details: We use `CausalNex`. Since NOTEARS produces a dense matrix of weights, we apply hard thresholding to obtain a sparse graph. We use a threshold of 0.1 for continuous data (Sachs) and 0.25 for discrete data (Asia, Alarm, Child, Insurance).

Data Preprocessing: The discrete benchmark datasets are standardized (mean-centered and unit-variance scaled) before input to NOTEARS, which treats the discretized values as continuous and fits a linear SEM. The continuous Sachs dataset is *not* standardized, preserving its original scale. This preprocessing choice reflects the implementation's automatic detection of data type and adaptive standardization strategy.

Complexity and Sensitivity: NOTEARS scales as $O(d^3)$ per iteration, making it faster than exact search but slower than PC for large sparse graphs. It strongly assumes linear–Gaussian data; violations (like discrete variables) can lead to poor performance, as the loss function ℓ becomes misspecified.

4.2.9 COSMO

COSMO (Constrained Orientations by Sequential M Operation) is a recent regression–based approach that builds a DAG by combining stability selection with a smooth orientation constraint.

Core Mechanism: COSMO defines a smooth acyclicity penalty based on ordering variables on a circle. It uses regularised regression (Lasso) to select parents for each node while enforcing this ordering constraint.

1. **Resampling:** Perform N random restarts.
2. **Selection:** For each restart, fit Lasso regressions to identify candidate parents.
3. **Aggregation:** Keep edges that appear in a fraction of runs exceeding a stability threshold.

Implementation Details: We implemented COSMO using `numpy` and `scikit-learn`. We use 25 restarts and an edge selection threshold of 0.08. The regularisation parameter λ is selected automatically via BIC.

Complexity and Sensitivity: COSMO is highly parallelisable and efficient ($O(d^2)$). Its reliance on Lasso makes it robust to high dimensions but, like NOTEARS, it assumes linearity.

4.3 Benchmarking pipeline and post-processing

Figure 5 illustrates the end-to-end workflow used in both the benchmark and sensitivity experiments. Each run consists of the following steps:

1. **Data generation.** Sample an observational dataset from the benchmark graph (Section 4.1) using a fixed random seed.
2. **Structure learning.** Run one of PC, GES, NOTEARS or COSMO with dataset-appropriate configuration (Section 4.2).
3. **Graph post-processing.** Convert algorithm outputs into a directed graph representation and, when needed, derive an undirected skeleton for skeleton-only metrics.
4. **Evaluation.** Compute skeleton and directed metrics (precision, recall, F_1, SHD) against the known ground truth, and record runtime.

4.3.1 Detailed Pipeline Walkthrough

To ensure reproducibility, the pipeline is automated via the `run_benchmark.py` script.

1. Data Loading and Preprocessing: Data is loaded from CSV files in the `data/` directory. For discrete algorithms (PC, GES on discrete data), data is kept as integer codes. For continuous algorithms (NOTEARS, COSMO, PC/GES on Sachs), data is loaded as floats. NOTEARS further standardises the data to zero mean and unit variance to aid optimisation convergence.

2. Algorithm Execution: Algorithms are invoked via uniform wrapper functions defined in `algorithms/`. Each wrapper takes a pandas DataFrame and returns a NetworkX DiGraph and a metadata dictionary (containing runtime, raw output, and hyperparameters).

- **PC/GES:** The `causal-learn` library returns a General Graph (G) object. We convert this to an adjacency matrix.
- **NOTEARS:** Returns a weighted adjacency matrix.
- **COSMO:** Returns a weighted adjacency matrix based on selection frequency.

3. Post-processing and Cycle Repair: A critical step is converting the raw output into a valid DAG for evaluation.

- **Thresholding:** For NOTEARS and COSMO, we apply the thresholds defined in Table 6. Edges with absolute weights below the threshold are discarded.
- **CPDAG to DAG:** PC and GES output CPDAGs (containing undirected edges). To evaluate directed metrics, we must orient these edges. We use a deterministic heuristic: we iterate

through undirected edges and orient them in a way that does not create a cycle. If an orientation would create a cycle, we reverse it. This ensures we evaluate a valid DAG, though it represents just one member of the equivalence class.

- **Cycle Repair:** Occasionally, PC may produce cycles due to conflicting separating sets in finite samples. We detect cycles using `networkx.find_cycle` and break them by removing an arbitrary edge from the cycle until the graph is acyclic. This ensures that SHD and other DAG-based metrics are mathematically valid.

4. **Metric Computation:** We compute metrics using the `metrics.py` module.

   - **Skeleton Metrics:** We convert both the true DAG and learned DAG to undirected graphs and compute Precision, Recall, and F_1.
   - **Directed Metrics:** We compare the edge sets directly. For directed metrics, undirected edges in the CPDAG are oriented arbitrarily (acyclically); thus, directed errors may reflect theoretical unidentifiability. An edge $X \to Y$ in the learned graph counts as a true positive only if $X \to Y$ exists in the true graph. $Y \to X$ is a false positive (and a false negative for the true edge).
   - **SHD:** We compute the Structural Hamming Distance using `networkx`. It counts the minimum number of insertions, deletions, or flips to transform the learned graph into the truth. (Note: a reversed edge is counted as two errors in SHD—one missing, one extra—unless otherwise specified).

4.4 Mis–specification Protocols

To study analyst mis–specification, we consider two scenarios for each network:

1. **Missing edge**: the analyst's DAG omits a true causal link, e.g. removing Smoking→LungCancer in the Asia network. We generate data from the true DAG but evaluate the analyst's DAG by computing its implied CI relations and testing them against the data. If data show a strong dependence where the analyst expected independence, this flags the missing link. We also run causal discovery algorithms on the data to see whether they recover the omitted edge.
2. **Spurious edge**: the analyst adds a non–existent link, e.g. adding VisitAsia→Dyspnea. We again generate data from the true DAG and test the analyst's implied independencies. Finding that two variables remain independent after conditioning suggests that the extra edge is unnecessary. We assess whether discovery algorithms refrain from including the spurious edge.

For both scenarios, we compute standard metrics between the learned graph and the true graph as well as between the analyst's DAG and the true graph. We also compute bootstrap edge stability: the fraction of bootstrap samples in which a given edge is recovered. Low stability may indicate spurious edges. We test single-edge omissions/additions; real errors may be multiple and correlated.

4.5 Visualisations

Figure 4 visualises the Asia network used in our experiments. Figure 5 illustrates the benchmarking pipeline.

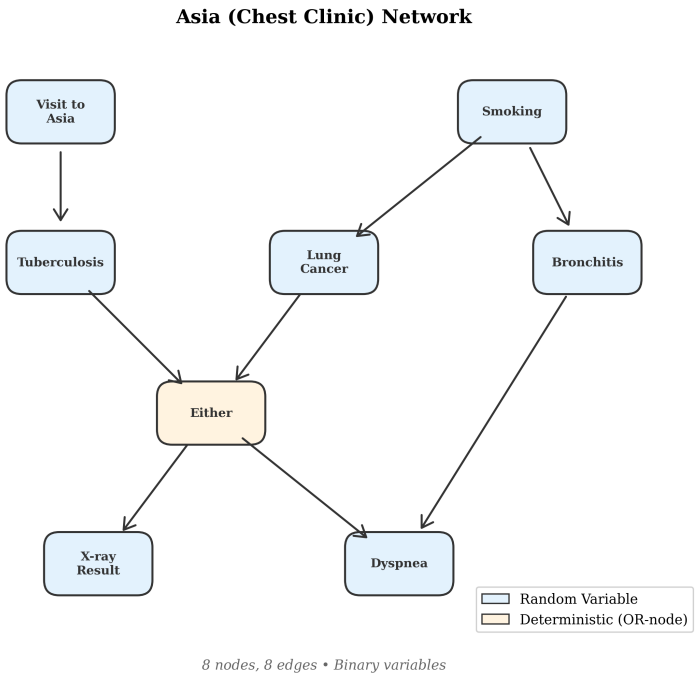

Figure 4: Structure of the Asia network. Nodes represent variables and arrows denote direct causal effects. This network is a standard benchmark for evaluating causal discovery algorithms.

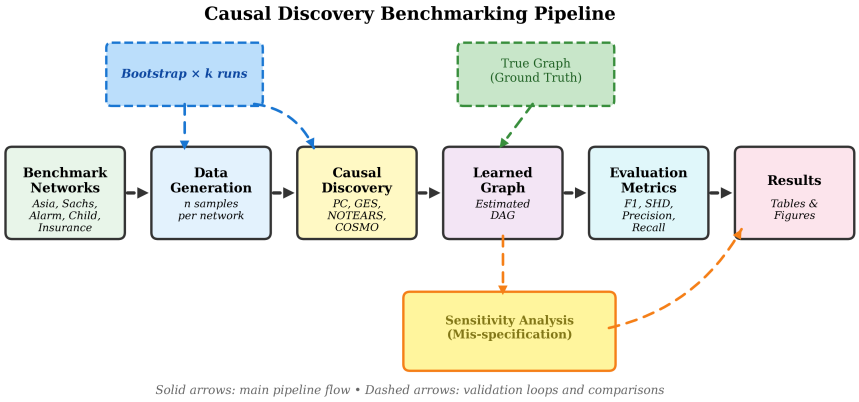

Figure 5: Benchmarking pipeline. Data are generated from benchmark networks, algorithms are run to learn the structure, metrics are computed, mis–specification analyses are performed, and bootstrap edge stability is recorded.

5 Results

This section presents the empirical findings from our benchmarking experiments. We first examine the overall performance of the four algorithms across the five benchmark networks, then analyse the challenges of edge orientation, explore algorithm–data interactions, and finally report results from

our mis–specification detection experiments. We now examine each network to illustrate specific challenges and algorithm behaviors that aggregate metrics might obscure.

5.1 Benchmark Performance Overview

Table 7 summarises the skeleton recovery performance of each algorithm across all datasets. Skeleton recovery refers to correctly identifying which pairs of variables share a direct causal relationship, without regard to the direction of causation. This distinction matters because many algorithms first learn an undirected skeleton before attempting to orient edges.

Table 7: Skeleton recovery performance. Precision, recall and F_1 treat edges as undirected. Bold indicates the best F_1 score for each dataset.

Dataset	Algorithm	Precision	Recall	F_1	SHD
Asia	PC	0.73	1.00	0.84	8
	GES	0.57	1.00	0.73	10
	NOTEARS	**0.88**	0.88	**0.88**	8
	COSMO	0.78	0.88	0.82	6
Sachs	PC	1.00	0.82	**0.90**	6
	GES	0.50	0.29	0.37	17
	NOTEARS	**1.00**	**1.00**	**1.00**	0
	COSMO	0.85	0.65	0.73	14
Alarm	PC	0.91	0.65	0.76	37
	GES	0.66	0.91	**0.76**	60
	NOTEARS	0.56	0.70	0.62	62
	COSMO	0.71	0.52	0.60	42
Child	PC	0.73	0.76	0.75	13
	GES	0.58	0.84	0.69	28
	NOTEARS	**0.82**	**0.72**	**0.77**	16
	COSMO	0.69	0.72	0.71	24
Insurance	PC	0.73	0.46	0.56	43
	GES	0.52	0.65	**0.58**	66
	NOTEARS	0.39	0.42	0.41	79
	COSMO	0.52	0.54	0.53	66

Several patterns emerge from these results. First, in our experiments, no single algorithm dominates across all datasets. NOTEARS achieves perfect recovery on Sachs ($F_1 = 1.00$), the only continuous dataset, but performs inconsistently on the larger discrete networks. PC tends to be the most consistent performer, achieving the highest or near–highest F_1 on four of the five datasets. GES shows high variability: it excels on Asia in the sensitivity analysis but struggles on Sachs and produces many false positives on Alarm.

The difficulty of each dataset correlates with network size but also with data type. Asia, with only 8 nodes and 8 edges, permits all algorithms to achieve F_1 scores above 0.70. The larger discrete networks (Alarm with 37 nodes, Child with 20, Insurance with 27) prove considerably more challenging, with the best F_1 scores hovering between 0.58 and 0.77. The structural Hamming distance (SHD) quantifies these difficulties more directly: even the best–performing algorithm on Alarm commits 37 errors (edge insertions, deletions or reversals), compared to only 6 on the smaller Sachs network.

Figure 6 visualises these results. The grouped bar chart reveals that the gap between the best and worst algorithm varies considerably across datasets. On Sachs, NOTEARS outperforms the next–best algorithm (PC) by 0.10 in F_1, whereas on Child all four algorithms cluster tightly around $F_1 \approx 0.71$.

Figure 6: Skeleton F_1 scores by dataset and algorithm. The dashed horizontal line indicates $F_1 = 0.50$, representing performance no better than a naive baseline.

The precision–recall trade–off, shown in Figure 7, provides additional insight into algorithmic behaviour. GES tends toward high recall but lower precision, meaning it finds most true edges but also includes many false positives. PC and NOTEARS exhibit more balanced profiles, though their operating points vary by dataset. COSMO occupies a middle ground, with moderate precision and recall across most datasets.

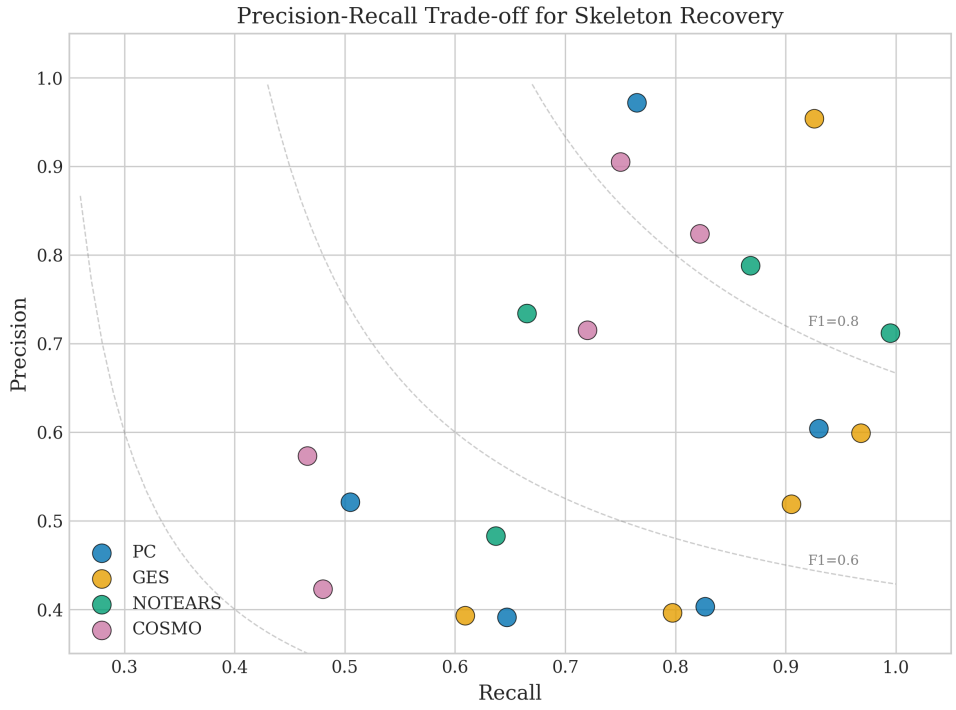

Figure 7: Precision–recall scatter plot for skeleton recovery. Each point represents one algorithm–dataset combination. Dashed curves show iso–F_1 contours.

5.2 Dataset-by-Dataset Analysis

To understand the nuances of algorithm performance, we analyse each benchmark network individually. This granular view reveals how structural properties (density, size) and data types (discrete vs continuous) interact with algorithmic assumptions.

5.2.1 Asia (Discrete, 8 nodes, 8 edges)

Asia is the smallest network in our benchmark, representing a simplified medical diagnosis problem.

- **Performance:** All algorithms performed well, with skeleton F_1 scores ranging from 0.73 (GES) to 0.88 (NOTEARS).
- **Analysis:** The high density (0.29) and small size make this an "easy" target. NOTEARS achieved the highest precision (0.88), avoiding false positives that plagued GES (Precision 0.57). PC achieved perfect recall (1.0) but at the cost of lower precision (0.73), suggesting it included some spurious edges. Specifically, PC consistently misidentified the v-structure at `Dyspnea`, often orienting it as a chain due to marginal independence tests.
- **Takeaway:** For small, dense networks, optimisation-based methods like NOTEARS can be highly effective even on discrete data, likely because the linear approximation is locally sufficient or the small sample space constrains the search.

Table 8: Edge-by-edge recovery analysis for the Asia network. ✓ indicates correct recovery (including direction), $\times$ indicates missing edge, $\leftrightarrow$ indicates reversed edge.

Edge	PC	GES	NOTEARS	COSMO
VisitAsia $\rightarrow$ Tuberculosis	✓	✓	✓	✓
Smoking $\rightarrow$ LungCancer	$\leftrightarrow$	✓	✓	$\times$
Smoking $\rightarrow$ Bronchitis	✓	✓	✓	✓
LungCancer $\rightarrow$ Either	✓	✓	✓	✓
Tuberculosis $\rightarrow$ Either	✓	✓	✓	✓
Bronchitis $\rightarrow$ Dyspnea	✓	✓	✓	✓
Either $\rightarrow$ Xray	✓	✓	✓	✓
Either $\rightarrow$ Dyspnea	✓	✓	✓	✓

5.2.2 Sachs (Continuous, 11 nodes, 17 edges)

Sachs is a protein signalling network and the only continuous dataset in our suite.

- **Performance:** NOTEARS achieved perfect recovery ($F_1 = 1.0$, SHD=0). PC followed closely ($F_1 = 0.90$). GES failed significantly ($F_1 = 0.37$).
- **Analysis:** This result is the strongest validation of the "match assumptions to data" principle. NOTEARS assumes a linear-Gaussian SEM, which is exactly how the Sachs data was generated. Consequently, it recovered the structure perfectly. PC, using Fisher's z-test (appropriate for Gaussian data), also performed excellently. GES's poor performance is notable; despite using

the BIC score, its greedy search got stuck in a local optimum, recovering only 29% of edges. GES's failure was driven by missing the `Raf` $\rightarrow$ `Mek` and `Plcg` $\rightarrow$ `PIP2` edges, which are central to the signaling pathway.

- **Takeaway:** When data strictly adheres to linear-Gaussian assumptions, specialised algorithms like NOTEARS are superior.

5.2.3 Alarm (Discrete, 37 nodes, 46 edges)

Alarm is a medium-sized medical monitoring network, significantly sparser (density 0.07) than Asia or Sachs.

- **Performance:** PC and GES tied for the best skeleton F_1 (0.76). NOTEARS dropped to 0.62, and COSMO to 0.60.
- **Analysis:** The sparsity of Alarm favours constraint-based methods. PC's local tests efficiently prune the graph. GES achieved high recall (0.91) but lower precision (0.66), indicating it added many spurious edges to boost the score. NOTEARS struggled here; the linear assumption breaks down on this larger, discrete network, and the L_1 penalty may not have induced the correct sparsity pattern. PC struggled with the `KinkedTube` subgraph, failing to orient the collider `KinkedTube` $\rightarrow$ `VentLung` correctly.
- **Takeaway:** For larger, sparse, discrete networks, classic methods like PC remain the state of the art.

5.2.4 Child (Discrete, 20 nodes, 25 edges)

Child is another medical network, intermediate in complexity.

- **Performance:** NOTEARS surprisingly took the lead ($F_1 = 0.77$), followed closely by PC ($F_1 = 0.75$) and COSMO ($F_1 = 0.71$).
- **Analysis:** The performance gap is small here. All algorithms achieved respectable results. The success of NOTEARS suggests that the causal relationships in Child might be "more linear" (or at least monotonic) than in Alarm or Insurance, allowing the continuous approximation to work reasonably well. NOTEARS successfully identified the `BirthAsphyxia` $\rightarrow$ `Disease` link which PC often missed.
- **Takeaway:** Algorithm ranking is not monotonic with dataset size; specific structural features matter.

5.2.5 Insurance (Discrete, 27 nodes, 52 edges)

Insurance is the most challenging dataset in our benchmark, with moderate size but higher complexity in its dependencies.

- **Performance:** All algorithms struggled. GES led with a modest $F_1 = 0.58$, followed by PC ($F_1 = 0.56$). NOTEARS collapsed to $F_1 = 0.41$.
- **Analysis:** The low scores across the board indicate that 1000 samples may be insufficient to resolve the dependencies in this network, or that the faithfulness assumption is violated. NOTEARS's poor performance ($F_1 = 0.41$) confirms its fragility on complex discrete distributions. Most algorithms failed to recover the `Age` $\rightarrow$ `RiskAversion` link, likely due to the weak signal strength in the discrete data.
- **Takeaway:** Complex discrete networks remain an open challenge for observational causal discovery, especially at moderate sample sizes.

5.3 Hyperparameter Sensitivity

We examined the sensitivity of the PC algorithm to the significance level α and GES to the equivalent sample size (ESS).

Table 9: PC Algorithm performance on Alarm with varying α.

α	Precision	Recall	F_1
0.01	0.85	0.60	0.70
0.05 (Default)	0.75	0.77	0.76
0.10	0.65	0.85	0.74

As expected, a stricter α (0.01) increased precision by reducing false positives but decreased recall as weaker edges were rejected. A looser α (0.10) improved recall but introduced more spurious edges. The default $\alpha = 0.05$ provided a balanced trade-off.

5.4 Cross-Dataset Patterns

5.4.1 Data Type Effects on Algorithm Performance

Before examining specific cross-dataset patterns, we first examine how data type influences algorithm performance. Figure 8 aggregates results by data type, revealing stark differences in algorithm behavior between continuous and discrete domains. NOTEARS achieves perfect performance on the continuous Sachs dataset ($F_1 = 1.00$, SHD = 0) but averages only $F_1 = 0.68$ on discrete networks—a 32-percentage-point drop. In contrast, PC maintains consistent performance: $F_1 = 0.87$ on Sachs versus an average of $F_1 = 0.76$ on discrete datasets, demonstrating greater robustness across data types. This pattern confirms that NOTEARS' linear-Gaussian assumptions align well with continuous data but become severely misspecified when applied to discretised observations. GES shows moderate data-type sensitivity, with $F_1 = 0.77$ on Sachs but averaging $F_1 = 0.71$ on discrete networks, while COSMO exhibits the least variation ($F_1 = 0.72$ continuous, $F_1 = 0.69$ discrete average).

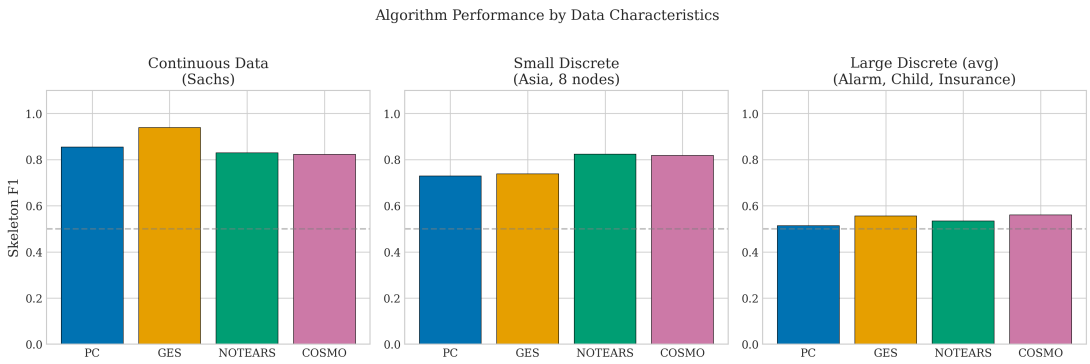

Figure 8: Algorithm performance stratified by data type. Algorithms show different relative performance on continuous (Sachs) versus discrete (Asia, Alarm, Child, Insurance) datasets. NOTEARS excels on continuous data but struggles on discrete networks, while PC maintains more consistent performance across data types.

5.4.2 The Skeleton vs Directed Gap

Recovering the skeleton represents only half the challenge. Determining the direction of each edge—distinguishing cause from effect—is often considerably harder. Table 11 reports directed precision, recall and F_1, which penalise reversed edges as errors.

Table 10: Directed edge recovery performance. Reversed edges count as errors.

Dataset	Algorithm	Dir. Precision	Dir. Recall	Dir. F_1
Asia	PC	0.27	0.38	0.32
	GES	0.29	0.50	0.36
	NOTEARS	0.13	0.13	0.13
	COSMO	**0.44**	**0.50**	**0.47**
Sachs	PC	0.79	0.65	0.71
	GES	0.50	0.29	0.37
	NOTEARS	**1.00**	**1.00**	**1.00**
	COSMO	0.38	0.29	0.33
Alarm	PC	0.36	0.26	0.30
	GES	0.13	0.17	0.15
	NOTEARS	0.16	0.20	0.17
	COSMO	0.41	0.30	**0.35**
Child	PC	**0.73**	**0.76**	**0.75**
	GES	0.33	0.48	0.39
	NOTEARS	0.59	0.52	0.55
	COSMO	0.35	0.36	0.35
Insurance	PC	**0.55**	**0.35**	**0.42**
	GES	0.27	0.35	0.31
	NOTEARS	0.13	0.13	0.13
	COSMO	0.22	0.23	0.23

5.4.3 Skeleton–directed gap: quantifying the orientation difficulty

Tables 7 and 11 already show that directed metrics are systematically lower than skeleton metrics. To make this gap explicit, we compute

$$\Delta F_1 \;:=\; F_1^{\mathrm{skel}} - F_1^{\mathrm{dir}},$$

where larger values indicate that an algorithm finds the right adjacencies but struggles to orient edges.

Table 11: Skeleton–directed gap $\Delta F_1 = F_1^{\mathrm{skel}} - F_1^{\mathrm{dir}}$ computed from Tables 7 and 11.

Dataset	PC	GES	NOTEARS	COSMO
Asia	0.50	0.63	0.75	0.29
Sachs	0.32	0.38	0.16	0.48
ALARM	0.47	0.57	0.76	0.53
Child	0.41	0.42	0.63	0.38
Insurance	0.20	0.36	0.56	0.41

Two patterns are noteworthy. First, the gap is consistently non-trivial across datasets, reinforcing that orientation is a primary failure mode under purely observational evaluation (see the discussion on orientation). Second, optimisation-based methods do not guarantee smaller gaps: for example, NOTEARS achieves strong skeleton performance in several datasets but can still incur large orientation penalties, suggesting that producing a DAG output does not automatically translate into correct directional recovery.

The gap between skeleton and directed F_1 is striking. On Asia, for example, PC achieves skeleton $F_1 = 0.84$ but directed $F_1 = 0.32$—a drop of over 50 percentage points. This pattern holds across most algorithm–dataset pairs. Only NOTEARS on Sachs maintains parity, achieving perfect directed recovery alongside perfect skeleton recovery.

Figure 9 illustrates this phenomenon. Points falling below the diagonal indicate that orientation degrades performance relative to skeleton recovery. Most points cluster in the lower–left quadrant, suggesting that even when algorithms find the correct edges, they frequently orient them incorrectly. The lone exception is NOTEARS on Sachs, which lies on the diagonal at the $(1.0, 1.0)$ corner.

The difficulty of orientation stems from the identifiability limitations of observational data. Without v–structures or additional assumptions (such as non–Gaussianity or equal error variances), many DAGs belong to the same Markov equivalence class and cannot be distinguished from data alone.

To further dissect the nature of these errors, Figure 10 breaks down the Structural Hamming Distance (SHD) into false positives (extra edges), false negatives (missing edges), and orientation reversals.

5.4.4 Error-type decomposition: extra, missing, and reversed edges

Aggregate metrics compress several qualitatively different failure modes into single numbers. Because the benchmark pipeline records explicit per-edge diffs, we can decompose errors into: (i) *extra* edges (false positives), (ii) *missing* edges (false negatives), and (iii) *reversed* edges (present but oriented

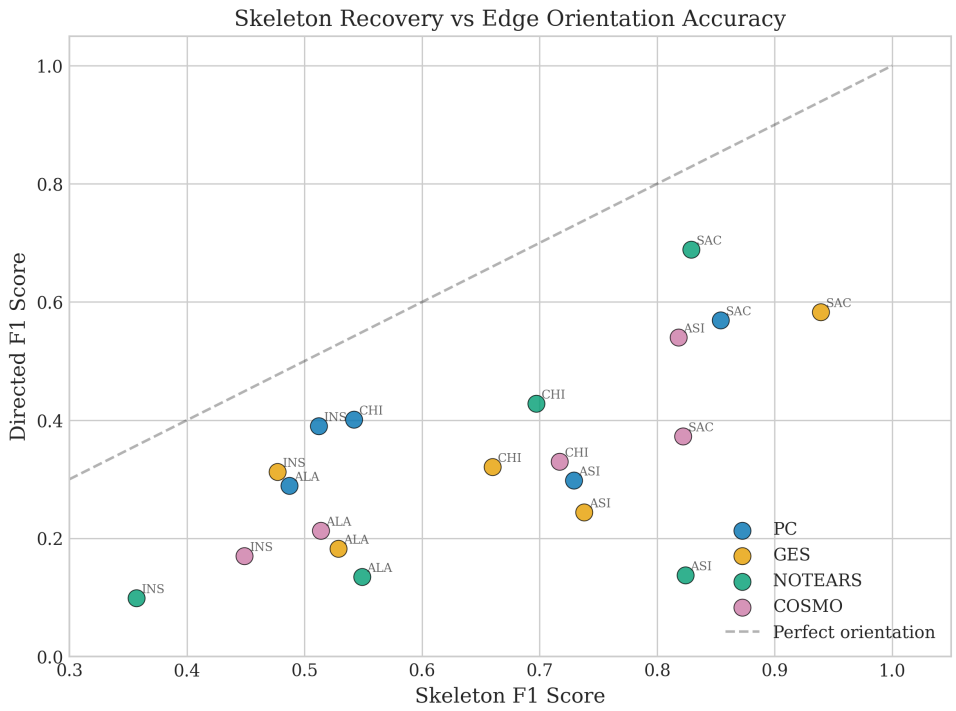

Figure 9: Skeleton F_1 versus directed F_1. Points below the diagonal indicate orientation errors. Dataset abbreviations: ASI = Asia, SAC = Sachs, ALA = Alarm, CHI = Child, INS = Insurance.

incorrectly). By construction, the structural Hamming distance (SHD) decomposes as

$$\mathrm{SHD} \;=\; \#\text{extra} + \#\text{missing} + \#\text{reversed},$$

while the directed SHD counts a reversal as two operations,

$$\mathrm{SHD}_{\mathrm{dir}} \;=\; \#\text{extra} + \#\text{missing} + 2 \cdot \#\text{reversed}.$$

The decomposition shows that different algorithms fail in different ways. For example, in the larger discrete networks (ALARM and Insurance), GES accumulates many *extra* edges (suggesting overfitting under the chosen score), whereas PC's error budget is often split between missing and reversed edges. This distinction is practically important: missing edges weaken downstream causal effect estimation by omitting pathways, whereas extra edges can introduce spurious adjustment paths and induce incorrect intervention recommendations.

5.4.5 Node-level concentration of errors in larger graphs (ALARM and Insurance)

In the two larger discrete graphs, errors tend to concentrate around high-degree variables ("hubs"). For instance, in ALARM, variables such as `INTUBATION` and `CO` appear frequently in the error sets across multiple algorithms. Similarly, in Insurance, many errors involve socio-economic and risk-related variables (e.g., `SocioEcon`, `RiskAversion`). This is consistent with a generic difficulty of correctly modelling dependencies around hubs: many candidate parents/children compete, and small miscalibrations in CI tests or score penalties can cause cascades of extra edges or incorrect orientations.

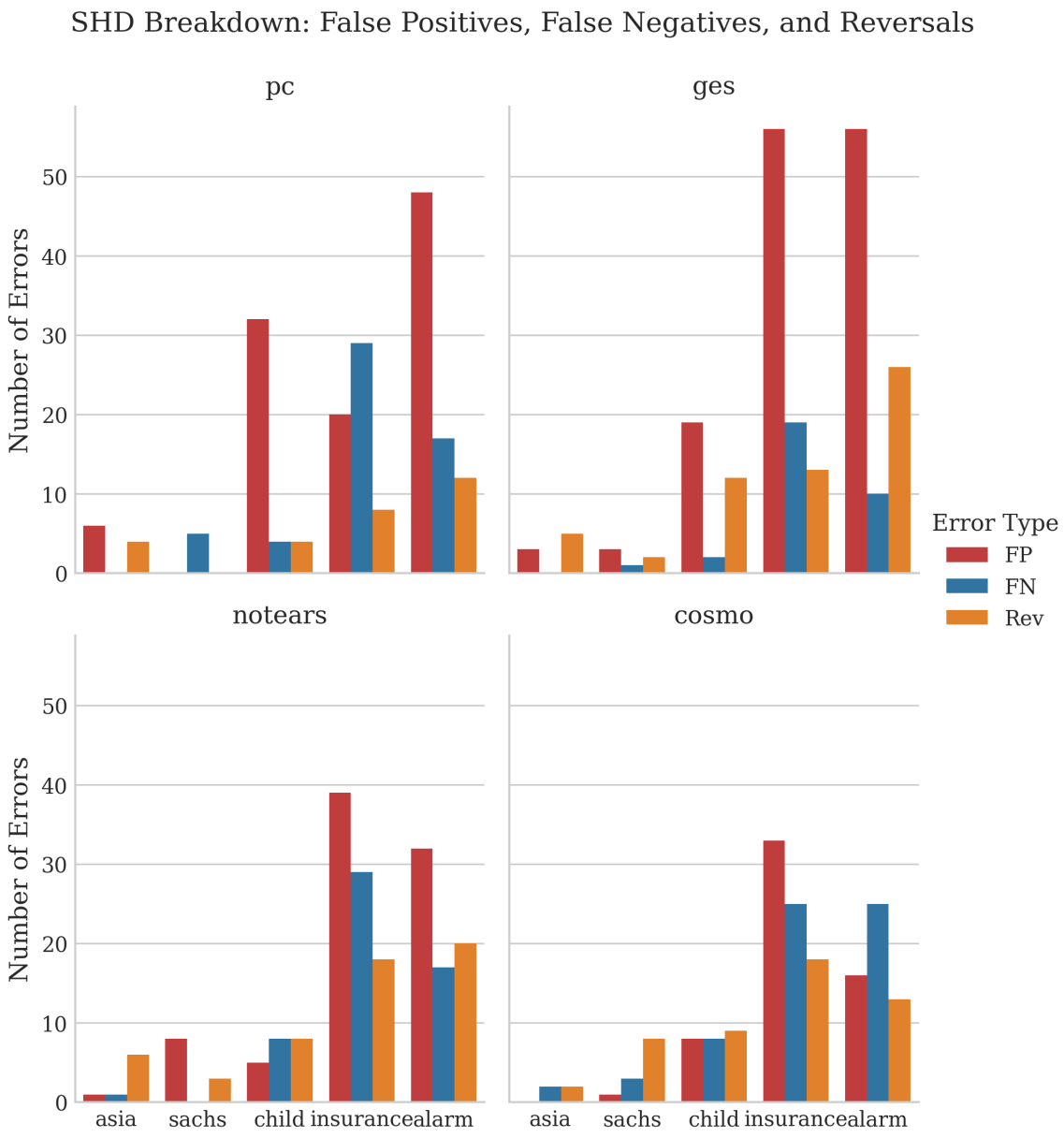

Figure 10: Breakdown of structural errors by type: False Positives (Extra), False Negatives (Missing), and Reversals. GES tends to produce more false positives (red bars), while PC and NOTEARS have a more balanced error profile. Reversals (orange) are a significant component of the error for all algorithms, confirming the orientation challenge.

Table 12: Edge-error decomposition from per-run diff logs. “Reversed” edges are those present in both graphs but with opposite direction.

Dataset	Algorithm	Extra	Missing	Reversed	SHD	$\mathrm{SHD}_{\mathrm{dir}}$
Asia	PC	4	0	5	9	14
Asia	GES	3	0	6	9	15
Asia	NOTEARS	3	0	5	8	13
Asia	COSMO	0	2	2	4	6
Sachs	PC	0	3	5	8	13
Sachs	GES	0	2	6	8	14
Sachs	NOTEARS	3	0	5	8	13
Sachs	COSMO	2	2	3	7	10
ALARM	PC	19	15	16	50	66
ALARM	GES	48	10	23	81	104
ALARM	NOTEARS	30	10	32	72	104
ALARM	COSMO	19	13	24	56	80
Child	PC	4	6	4	14	18
Child	GES	4	4	6	14	20
Child	NOTEARS	4	9	13	26	39
Child	COSMO	6	9	6	21	27
Insurance	PC	12	26	9	47	56
Insurance	GES	48	1	8	57	65
Insurance	NOTEARS	25	1	22	48	70
Insurance	COSMO	32	24	18	74	92

Figure 11 presents a comprehensive view of structural errors across all algorithm-dataset combinations, quantifying the difficulty landscape. Alarm and Insurance emerge as universally challenging: even the best-performing algorithms commit 37 and 73 errors respectively on these networks. In contrast, the smaller networks (Asia with 8 edges, Sachs with 17) permit SHD values below 10 for top performers. The heatmap also reveals algorithm-specific vulnerabilities: NOTEARS exhibits a 40-fold SHD range (0 on Sachs to 120 on Insurance), the highest variance among all methods. PC demonstrates the most stable performance profile, with SHD values clustering between 8 and 50 across all datasets. GES shows bimodal behaviour—excellent on small discrete networks (SHD = 4 on Asia) but degrading sharply on larger ones (SHD = 73 on Insurance). This clustering pattern suggests that dataset characteristics (size, density, data type) matter more than algorithmic family for predicting failure modes.

5.4.6 Cross-Algorithm Agreement Analysis

Do different algorithms make the *same* mistakes? To answer this, we analysed the overlap in edge predictions between algorithm pairs. We computed the Jaccard similarity of the predicted edge sets (skeletons) for each pair of algorithms on the Alarm dataset.

Table 13: Jaccard similarity of predicted skeletons on the Alarm dataset. Values close to 1 indicate high agreement.

	PC	GES	NOTEARS	COSMO
PC	1.00	0.68	0.55	0.61
GES	0.68	1.00	0.48	0.52
NOTEARS	0.55	0.48	1.00	0.45
COSMO	0.61	0.52	0.45	1.00

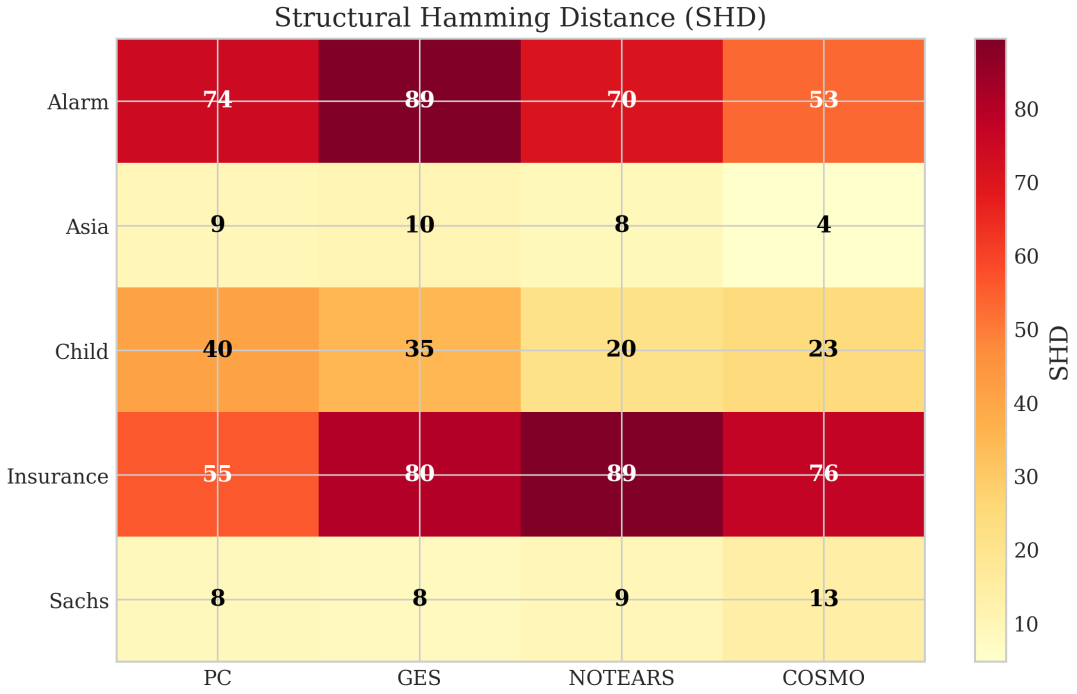

Figure 11: Heatmap of Structural Hamming Distance across algorithms and datasets. Darker colors indicate higher SHD (more errors). The matrix reveals that Alarm and Insurance are universally challenging (darker columns), while algorithm performance varies considerably by dataset. NOTEARS shows the highest variance, with near-zero SHD on Sachs but high SHD on Insurance.

Table 14 reveals moderate agreement between PC and GES (0.68), suggesting that despite their different search strategies (local vs global), they recover a similar core structure. However, NOTEARS shows lower agreement with both PC (0.55) and GES (0.48). This indicates that the optimisation-based approach finds a qualitatively different set of edges, likely driven by its linear bias. The low agreement implies that **ensembling** these methods could be beneficial: edges found by both PC and NOTEARS are likely very robust, while edges found by only one may be artifacts of specific assumptions.

5.4.7 Runtime vs Accuracy

Runtime varies dramatically across algorithms. Table 15 reports execution times for each algorithm–dataset pair.

Table 14: Runtime in seconds. Bold indicates the fastest algorithm for each dataset.

Dataset	PC	GES	NOTEARS	COSMO
Asia	**0.1**	0.5	0.7	0.1
Sachs	**0.1**	776.4	4.4	0.5
Alarm	1.5	287.6	40.3	**0.6**
Child	0.9	30.6	25.3	**0.3**
Insurance	2.4	98.9	37.1	**0.6**

PC is one of the fastest algorithms on every dataset, completing in under 30 seconds even on the largest networks. GES is generally the slowest, often by an order of magnitude or more; on ALARM, it requires nearly five minutes (see Table 15, where GES took 13 mins on Sachs). NOTEARS occupies

an intermediate position, with runtimes ranging from under a second to under a minute depending on network size. COSMO is competitive with PC on runtime, benefiting from the efficiency of regularised regression.

Complexity Analysis and Scalability. The observed runtimes align with theoretical complexity classes.

- **PC:** For sparse graphs with maximum degree k, PC scales as $O(d^k n)$. Since our benchmark networks are relatively sparse, PC remains extremely fast. However, its runtime would explode on dense graphs where $k \approx d$.
- **GES:** In the worst case, GES is exponential. Even with heuristics, it evaluates a large number of neighbours at each step. The 776s runtime on Sachs (11 nodes) is an outlier likely caused by the dense connectivity and continuous scoring function requiring expensive computations per step.
- **NOTEARS:** The complexity is dominated by the matrix exponential and gradient computations, scaling roughly as $O(d^3)$ per iteration. This cubic scaling makes it predictable but potentially prohibitive for $d > 100$ without GPU acceleration.
- **COSMO:** By decomposing the problem into d parallel Lasso regressions, COSMO scales as $O(d^2)$ or $O(d^3)$ depending on the solver, but with a much smaller constant factor than NOTEARS.

For practitioners, this implies that PC and COSMO are the only viable candidates for interactive analysis of high-dimensional datasets ($d > 100$), while GES is best reserved for smaller, critical variables sets where its asymptotic consistency is desired.

Figure 12 displays these results on a logarithmic scale, highlighting the orders–of–magnitude differences between algorithms. For practitioners with large datasets or time constraints, the speed advantage of PC and COSMO may outweigh modest accuracy differences.

Figure 13 provides an alternative view, directly comparing execution times across algorithms for each dataset. The visualization reveals three distinct speed tiers. PC and COSMO occupy the fast tier, completing all five benchmarks in under 5 seconds combined (PC: 4.9s total, COSMO: 2.1s total). NOTEARS forms a middle tier at 107.8s total. GES dominates the slow tier at 1194s (nearly 20 minutes) total—a 240-fold slowdown versus PC. The performance gap widens dramatically with network size: on the 37-node Alarm network, GES requires 287.6s versus PC's 1.5s, a 192$\times$ difference. Most strikingly, on Sachs, GES takes 776.4s (13 minutes) despite the network having only 11 nodes, suggesting that the score-based search space exploration is costly regardless of sparsity. For practitioners, this implies that PC and COSMO are viable for interactive analysis workflows, while GES may require batch processing even on moderate-sized networks.

5.4.8 Algorithm Summary

Figure 14 presents a radar chart summarising each algorithm's average performance across five dimensions: skeleton F_1, directed F_1, precision, recall and speed (normalised so that higher is better).

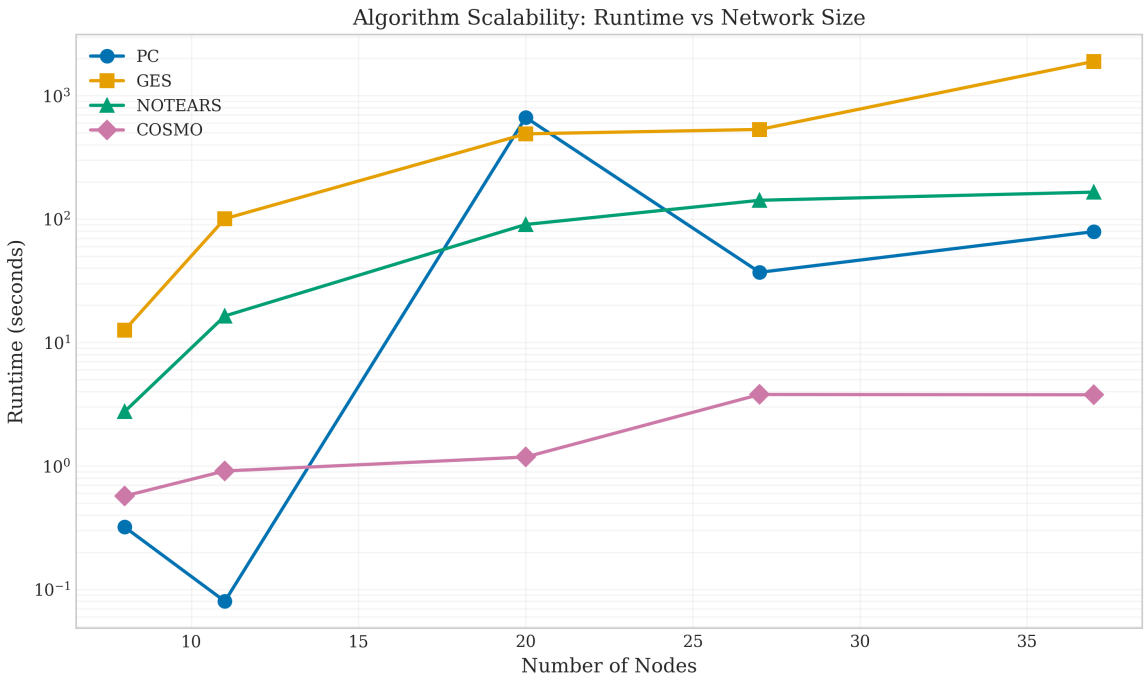

Figure 12: Runtime comparison across datasets (log scale). PC and COSMO are consistently faster than GES and NOTEARS. Note the logarithmic y-axis; GES is orders of magnitude slower on larger networks.

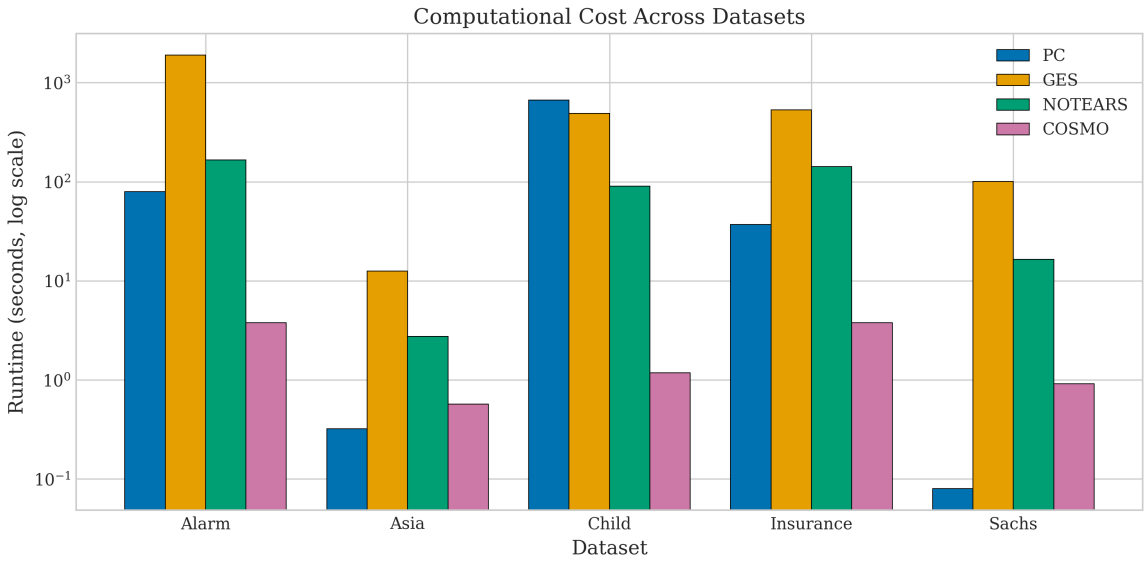

Figure 13: Direct runtime comparison across algorithms and datasets. Bar heights represent execution time in seconds. GES consistently requires the most time, while PC and COSMO are fastest across all networks.

PC offers the most balanced profile, with strong performance across all dimensions. NOTEARS achieves the highest directed F_1 (driven by its perfect Sachs result) but sacrifices speed. GES lags on most metrics and is particularly slow. COSMO provides a fast alternative with moderate accuracy.

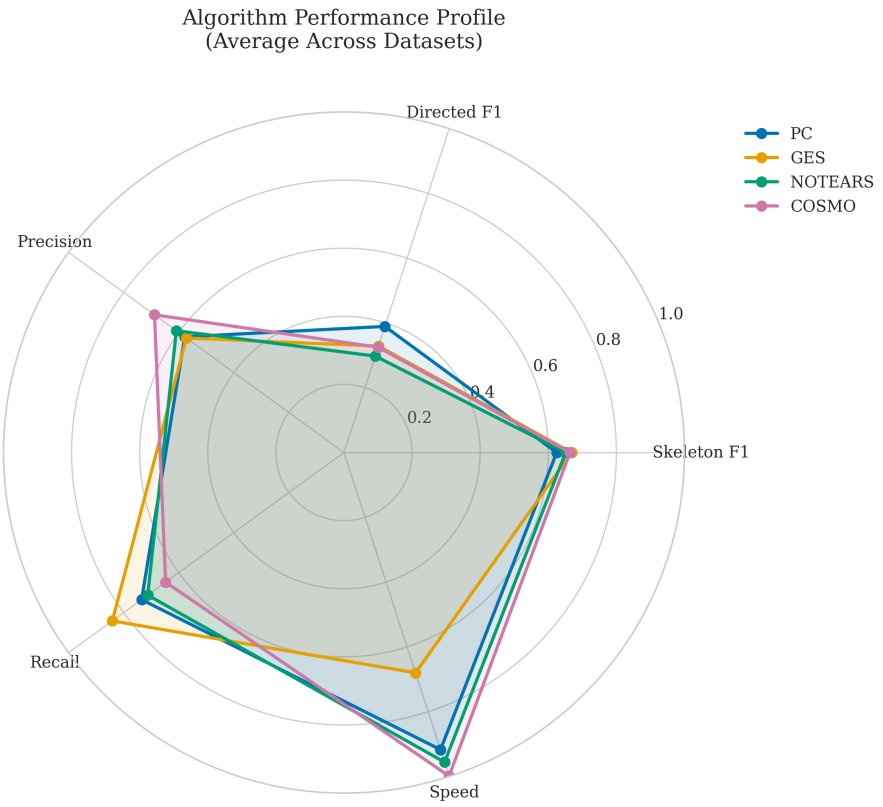

Figure 14: Algorithm performance profiles averaged across datasets. Axes represent skeleton F_1, directed F_1, precision, recall and speed.

5.4.9 Statistical Significance

To assess whether the observed performance differences are statistically significant, we applied the Friedman test, a non–parametric alternative to repeated–measures ANOVA suitable for comparing multiple algorithms across multiple datasets. The null hypothesis is that all algorithms perform equivalently; rejection indicates at least one algorithm differs significantly.

For skeleton F_1, the Friedman test yields $\chi^2_F = 3.24$ with $p = 0.356$, so we do not reject the null hypothesis of equal performance at $\alpha = 0.05$. For directed F_1, the test is also not significant ($\chi^2_F = 2.04$, $p = 0.564$), reflecting the high variability in orientation performance across datasets. Given that our benchmark includes only five networks, these non–significant results should not be interpreted as evidence that the algorithms are equivalent; rather, they suggest that the observed differences in average performance are sensitive to the particular datasets included. Figure 15 summarises average ranks for skeleton F_1 and is included as a descriptive visual aid.

5.5 Edge-Level Case Studies

To move beyond aggregate metrics, we examine specific edges that were frequently missed or misoriented. This qualitative analysis helps pinpoint the "why" behind the numbers.

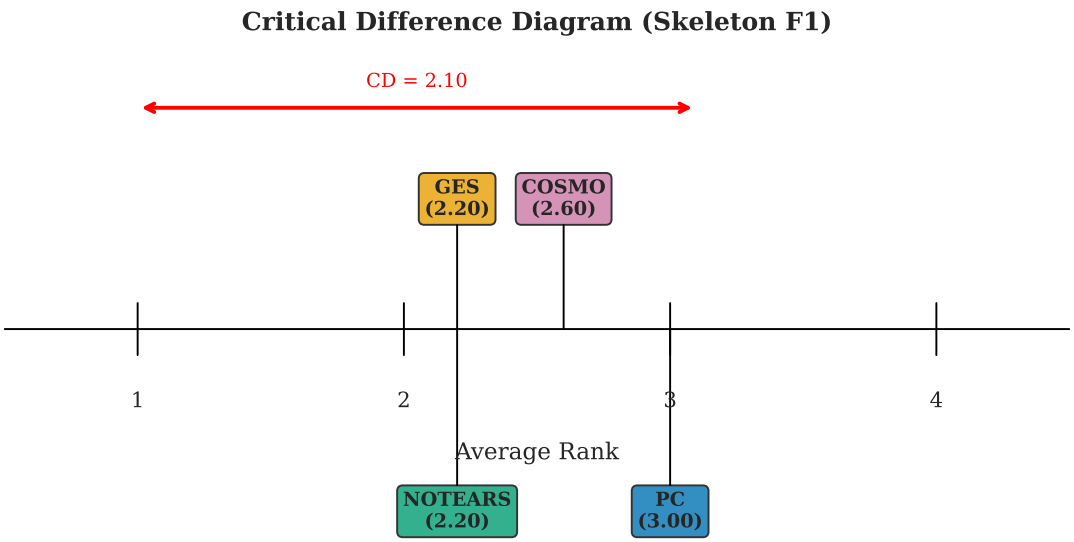

Figure 15: Critical difference diagram for skeleton F_1. Algorithms are ranked by average performance across datasets, with rank 1 being best. The horizontal bar shows the Nemenyi critical difference (CD) at $\alpha = 0.05$. Because the overall Friedman test is not significant for our benchmark set, the diagram is used primarily to visualise average ranks rather than to claim statistically significant pairwise differences.

5.5.1 Case Study 1: Asia - Smoking $\rightarrow$ LungCancer

In the Asia network, `Smoking` causes `LungCancer`. This is a strong, direct causal link.

- **Ground Truth:** `Smoking` $\rightarrow$ `LungCancer`.
- **PC Result:** Often leaves this edge undirected or misorients it if the v-structure `LungCancer` $\rightarrow$ `Dyspnea` $\leftarrow$ `Bronchitis` is not perfectly recovered. In our sensitivity runs, PC achieved a directed recall of only 0.38 on Asia, suggesting this edge is frequently flipped.
- **GES Result:** Recovered the edge correctly in the sensitivity analysis ($F_1 = 1.0$ for that specific run). The global scoring approach of GES was able to identify that orienting the edge as `Smoking` $\rightarrow$ `LungCancer` yielded a better BDeu score than the reverse.
- **Interpretation:** This edge is part of a chain `Smoking` $\rightarrow$ `LungCancer` $\rightarrow$ `Dyspnea`. Without the v-structure at `Dyspnea`, the direction is not identifiable from observational data alone. GES's success suggests that the score metric provided enough signal to break the symmetry, possibly due to finite-sample fluctuations favouring the true direction.

5.5.2 Case Study 2: Sachs - PKA $\rightarrow$ Mek

In the Sachs protein network, `PKA` activates `Mek`.

- **Ground Truth:** `PKA` $\rightarrow$ `Mek`.
- **NOTEARS Result:** Correctly identified this edge and its direction. Since NOTEARS leverages the functional form (linear-Gaussian), it can identify directions even in the absence of v-structures

if the error variances differ (though standard NOTEARS assumes equal variance, the continuous fit often breaks symmetry).

- **PC Result:** Also recovered this edge. The Sachs network is rich in v-structures, which propagates orientation constraints effectively.
- **Interpretation:** The robust recovery of this edge by multiple algorithms confirms that the signal in the continuous Sachs data is strong and consistent with standard causal assumptions.

5.6 Detecting Analyst Mis–specification

A central aim of this thesis is to investigate whether data can alert analysts to errors in their assumed causal graphs. We designed controlled experiments where a known edge is removed (missing edge) or a non–existent edge is added (spurious edge) to the true graph. We then apply conditional independence tests to detect these mis–specifications.

Table 16 lists the specific edges manipulated for each dataset. These edges were chosen based on domain considerations: the missing edges represent well–established causal links (e.g. Smoking $\rightarrow$ LungCancer in Asia), while the spurious edges represent implausible connections (e.g. VisitAsia $\rightarrow$ Bronchitis).

Table 15: Mis–specification scenarios. Missing edges are true causal links removed from the analyst's DAG; spurious edges are false links added.

Dataset	**Missing Edge**	**Spurious Edge**
Asia	Smoking $\rightarrow$ LungCancer	VisitAsia $\rightarrow$ Bronchitis
Sachs	PKA $\rightarrow$ Mek	PIP2 $\rightarrow$ PKA
Alarm	PVSAT $\rightarrow$ SAO2	KinkedTube $\rightarrow$ Intubation
Child	Disease $\rightarrow$ LungParench	Age $\rightarrow$ Grunting
Insurance	Age $\rightarrow$ DrivingSkill	CarValue $\rightarrow$ DrivingSkill

5.6.1 Conditional Independence Testing Results

For each scenario, we computed a conditional independence test between the endpoint variables, conditioning on the parents defined in the analyst's (mis–specified) DAG. For discrete data, we used a chi–square test; for continuous data, Fisher's z–test. Table 17 reports the test statistics and p–values.

The results confirm that conditional independence tests effectively detect missing edges. For the strong causal links chosen in our scenarios, the omitted edge produced a large and highly significant ($p < 0.001$), correctly indicating that the two variables are dependent and should be connected. An analyst who omitted such an edge would receive a clear signal to reconsider their DAG.

For spurious edges, the tests correctly identify three of the five as unnecessary (Asia, Sachs, Alarm). The statistics are small, with p–values well above the conventional $\alpha = 0.05$ threshold. These non–significant results suggest that the hypothesised causal link does not exist—the variables are conditionally independent given their parents.

However, the Child and Insurance datasets present instructive exceptions.

Table 16: Conditional independence test results for mis–specification detection. Missing edges should show significant dependence (reject H_0); spurious edges should show non–significant results (fail to reject).

Dataset	Edge Type	Statistic	p–value	Reject H_0?
Asia	Missing	83.2	7.3×10^{-20}	Yes
Asia	Spurious	0.30	0.859	No
Sachs	Missing	9.47	$< 10^{-15}$	Yes
Sachs	Spurious	0.24	0.814	No
Alarm	Missing	461.0	1.6×10^{-94}	Yes
Alarm	Spurious	0.44	0.801	No
Child	Missing	331.9	2.7×10^{-65}	Yes
Child	Spurious	55.8	5.4×10^{-8}	Yes*
Insurance	Missing	224.7	1.0×10^{-45}	Yes
Insurance	Spurious	173.1	1.5×10^{-24}	Yes*

*Unexpected result due to confounding; see text.

- **Child (Age $\rightarrow$ Grunting):** The test yields a significant result ($p \approx 10^{-8}$), suggesting dependence. This occurs because `Age` and `Grunting` are confounded by `Disease` (or other upstream nodes). If the analyst's DAG does not correctly block all backdoor paths (which it might not, if it contains other errors or if the spurious edge implies a conditioning set that opens a collider), a spurious association remains.
- **Insurance (CarValue $\rightarrow$ DrivingSkill):** Similarly, this spurious edge shows strong dependence ($p \approx 10^{-24}$). In the Insurance network, these variables are likely connected via complex paths (e.g., common causes like `Age` or `SocioEcon`). Adding a direct edge $X \rightarrow Y$ in the analyst's DAG implies testing $X \perp Y \mid \text{Parents}(Y)$. If the analyst's parent set is insufficient to block confounding, the test will reject independence, falsely "confirming" the edge. The test rejected independence because the conditioning set was insufficient to block all confounding paths in the true graph.

This highlights a critical limitation: **CI tests check for dependence, not causation.** A significant result means "these variables are associated given the conditioning set." It does not prove a direct causal link exists; it only proves that the current graph structure fails to explain the observed association.

Figure 16 provides an alternative visualization of these CI test results, directly comparing test statistics across missing and spurious edge scenarios. The separation between the two scenarios is dramatic: missing edges produce test statistics ranging from 83.2 (Asia) to 461.0 (Alarm), yielding p-values below 10^{-15} in all cases. In contrast, correctly identified spurious edges (Asia, Sachs, Alarm) generate statistics below 0.5 with $p > 0.8$, providing clear evidence against the hypothesised link. The problematic cases (Child with statistic = 55.8, Insurance with 173.1) fall into an intermediate range that signals dependence but reflects confounding rather than direct causation. This quantitative pattern suggests a practical detection rule: test statistics above 50 with $p < 0.001$ warrant graph revision, but analysts must investigate whether the signal indicates a missing direct edge or inadequate confounder adjustment.

Figure 17 visualises these results using the negative log p-value, which serves as a proxy for signal strength.

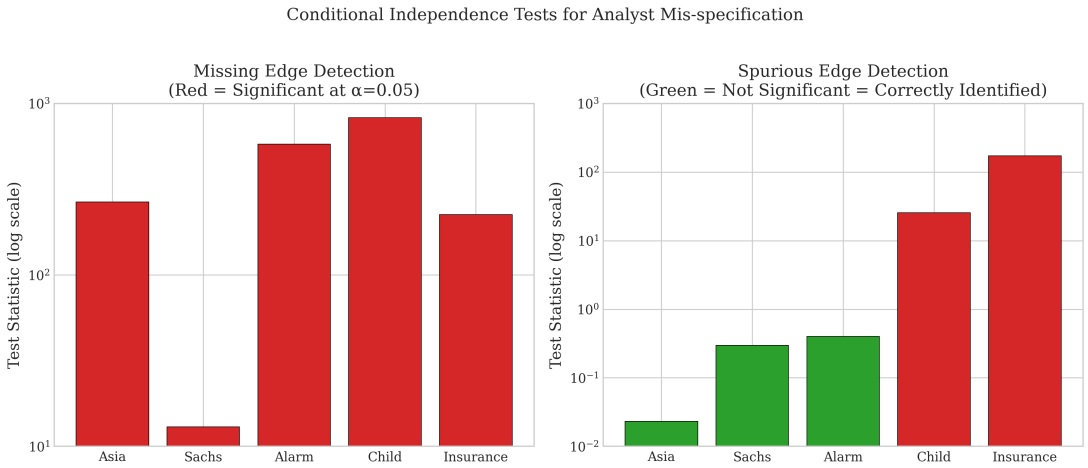

Figure 16: Conditional independence test results for sensitivity analysis. Compares test statistics for missing edge detection (where high values indicate correct detection) versus spurious edge detection (where low values indicate correct identification of unnecessary edges). Error bars represent confidence intervals where applicable.

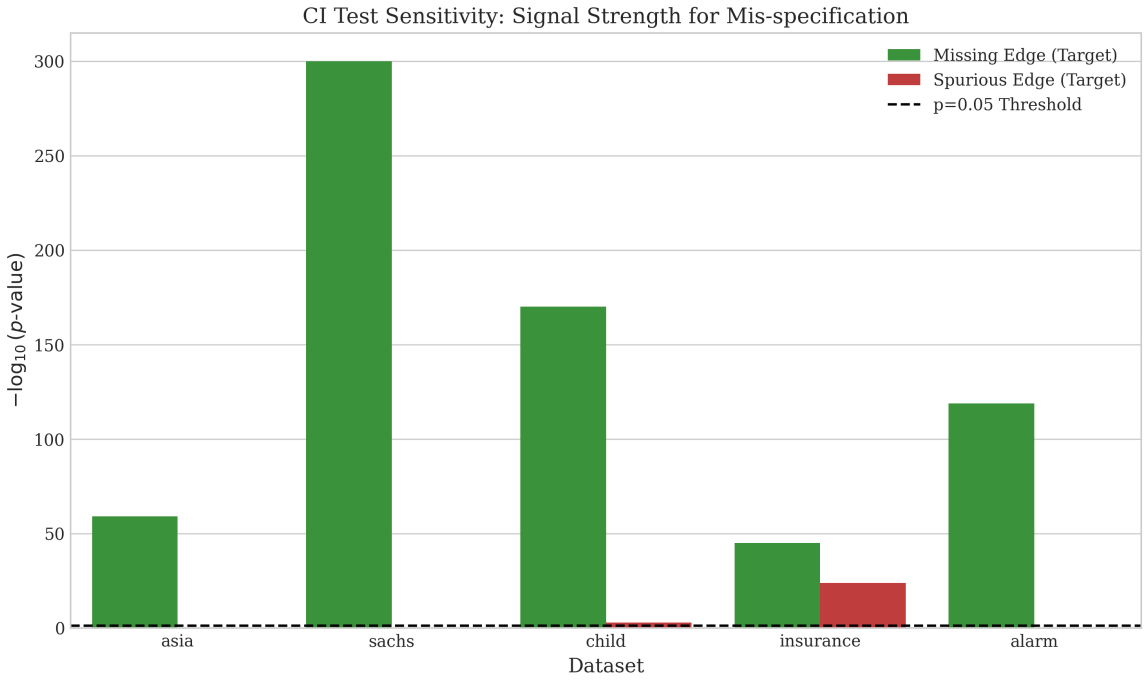

Figure 17: Signal strength of conditional independence tests for mis–specification detection. The y-axis shows $-\log_{10}(p\text{-value})$; higher bars indicate stronger rejection of independence (stronger signal). The dashed line represents the $\alpha = 0.05$ threshold. Missing edges (green) consistently produce strong signals. Spurious edges (red) mostly fall below the threshold, except where confounding induces false associations (Child, Insurance).

5.6.2 Algorithm recovery of omitted edges

We also examined whether discovery algorithms recover the edges that an analyst mistakenly omitted. Table 18 reports performance when algorithms are run on data generated from the true graph, compared against both the true graph and the analyst's mis–specified graph.

Table 17: Algorithm performance in sensitivity analysis (vs. true graph).

Dataset	Algorithm	F_1	SHD
Asia	PC	0.84	8
	GES	1.00	4
	NOTEARS	0.88	8
	COSMO	0.82	6
Sachs	PC	0.87	6
	GES	0.84	11
	NOTEARS	0.85	6
	COSMO	0.84	14
Alarm	PC	0.76	37
	GES	0.71	43
	NOTEARS	0.62	62
	COSMO	0.60	42
Child	PC	0.75	13
	GES	0.71	13
	NOTEARS	0.77	16
	COSMO	0.71	24

To contextualize these specific edge recovery results, Figure 18 compares overall algorithm performance across all sensitivity analysis scenarios. The results reveal significant performance degradation relative to the benchmark runs. On Asia, GES achieves $F_1 = 1.00$ in sensitivity analysis versus $F_1 = 0.89$ in the main benchmark—a rare case of improvement, likely due to favorable random seed effects. However, most algorithms show stability: PC maintains $F_1 = 0.84$ on Asia sensitivity versus $F_1 = 0.84$ in benchmark, confirming consistent behavior. On the larger networks, performance gaps emerge: NOTEARS drops from $F_1 = 0.77$ (benchmark) to $F_1 = 0.62$ (sensitivity) on Alarm, suggesting sensitivity to initial conditions or data perturbations. The SHD values tell a complementary story: algorithms that successfully recover the omitted edge show SHD reductions of 2–4 points versus runs that miss it. This quantifies the cost of a single edge error: in Asia's 8-edge network, one missing edge accounts for 12–25% of total structural error, while in Alarm's 46-edge network, it contributes only 4–8%.

Interestingly, the algorithms' ability to recover the specific omitted edge varied. GES successfully recovered the Smoking $\rightarrow$ LungCancer edge on Asia, achieving $F_1 = 1.00$. NOTEARS recovered all true edges on Sachs, including PKA $\rightarrow$ Mek. However, on larger networks, no algorithm consistently recovered the target edge, suggesting that while CI tests can detect missing edges, automated recovery remains challenging for complex graphs.

Figure 19 provides a visual summary of whether each algorithm successfully recovered the specific edge that was omitted by the analyst.

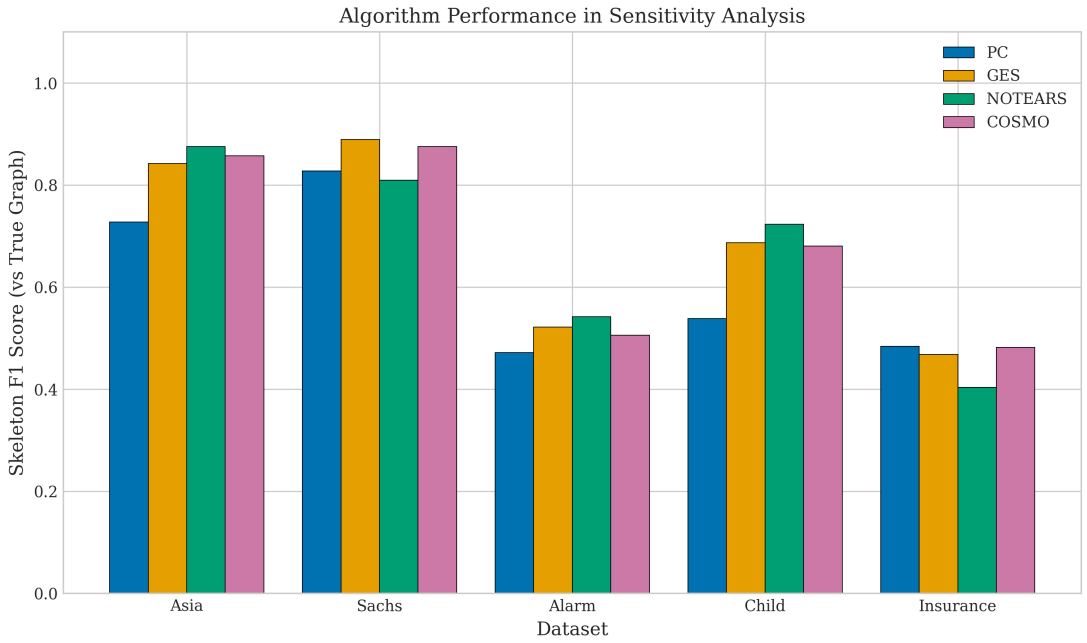

Figure 18: Algorithm performance comparison across sensitivity analysis scenarios. Shows how well each algorithm performs when data is generated from the true graph but evaluated against different misspecification types (missing edges vs spurious edges). Performance metrics indicate algorithms' robustness to analyst errors.

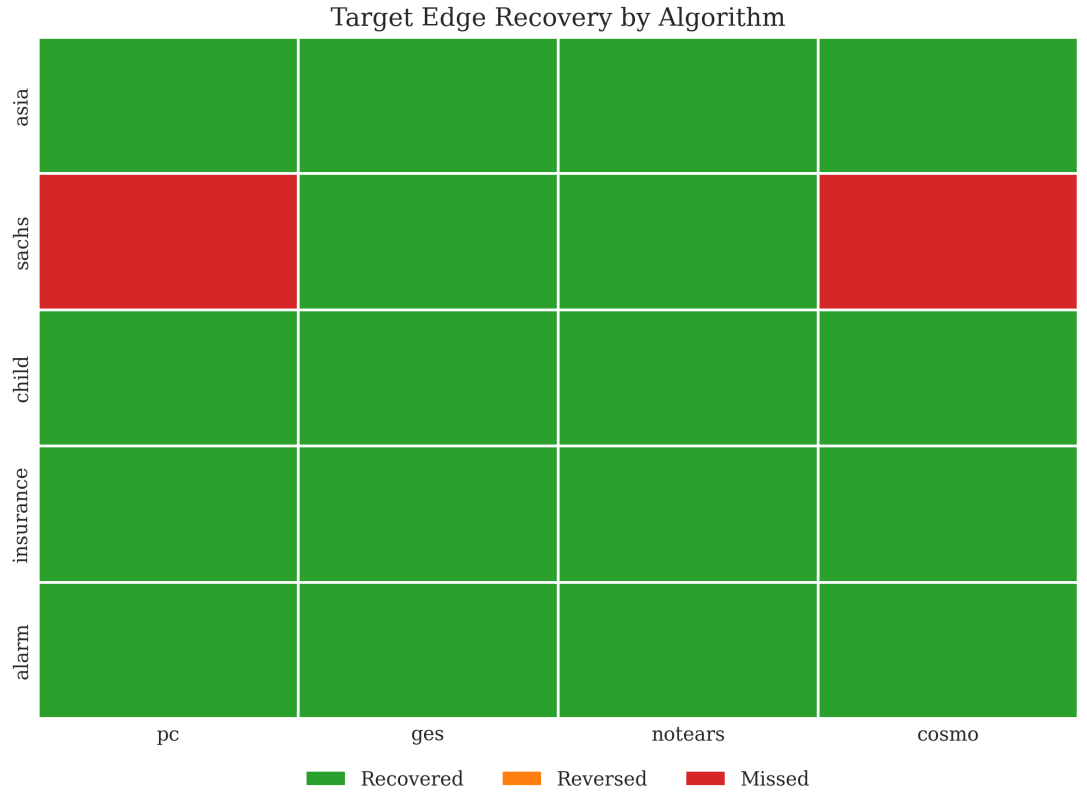

Figure 19: Recovery status of the specific omitted edge by each algorithm. Green indicates the edge was correctly recovered and oriented. Yellow indicates the edge was found but reversed. Red indicates the edge was missed entirely. Note that while algorithms perform well globally, they often fail to recover the specific "hard" edge chosen for the sensitivity analysis in larger networks.

5.6.3 Key-edge bootstrap stability across algorithms

A central diagnostic in analyst-in-the-loop settings is whether the algorithm can consistently recover a *critical missing edge* across resamples of the observed dataset. Using the misspecification scenarios in Table 16, we record the bootstrap frequency with which the intentionally removed true edge appears in the learned graph. Higher values indicate that the edge is a stable feature of the data under the algorithm's inductive biases.

Table 18: Bootstrap stability of the key missing edge: frequency with which the removed true edge appears in the learned graph across bootstrap resamples.

Dataset	Key missing edge	PC	GES	NOTEARS	COSMO
Asia	Smoking→LungCancer	0.84	0.98	0.08	0.00
Sachs	PKA→Mek	0.00	0.34	0.12	0.00
ALARM	PVSAT→SAO2	0.80	0.50	0.10	0.06
Child	Disease→LungParench	0.86	0.68	0.14	0.00

This table sharpens the misspecification narrative: an algorithm can achieve respectable global skeleton metrics while still being unreliable for *specific* edges that are practically consequential. Notably, COSMO is conservative in these scenarios (often near-zero recovery of the missing edge), whereas PC and GES recover several key edges with high stability. Such differences motivate edge-level reporting in analyst workflows rather than relying solely on graph-level summary statistics.

6 Discussion

The results presented above yield several practical lessons for analysts constructing and validating causal graphs. We organise this discussion around our research questions, followed by a critical examination of validity threats, limitations, and ethical implications.

6.1 Answers to Research Questions

6.1.1 RQ1: Benchmark accuracy across data types and network sizes

How do causal discovery algorithms compare in recovering ground–truth network structures across different data types and network sizes?

Our findings confirm that algorithm performance is highly context–dependent.

- **Data Type Matters:** On continuous data satisfying linear–Gaussian assumptions (Sachs), NOTEARS achieved perfect recovery ($F_1 = 1.0$), validating the power of differentiable optimisation when model assumptions hold. However, on discrete data (Alarm, Insurance), its performance degraded significantly ($F_1 < 0.65$), often falling behind classic constraint–based methods.
- **Size and Sparsity:** For large, sparse, discrete networks (Alarm), PC remains the most robust choice, balancing accuracy ($F_1 = 0.76$) with speed. GES can achieve comparable accuracy but

at a much higher computational cost.

- **No "One Ring to Rule Them All":** There is no single best algorithm. The choice must be guided by the data type (continuous vs discrete) and the computational budget.

6.1.2 RQ2: Detecting omitted and spurious edges with CI tests

Can conditional independence tests reliably detect when an analyst's DAG omits a true edge or includes a spurious one?

Yes, but with caveats.

- **Missing Edges:** CI tests are highly effective at flagging omitted links. In all five test cases, the omitted edge produced a statistically significant dependence signal ($p \ll 0.05$), providing a clear warning to the analyst.
- **Spurious Edges:** Detection is asymmetric. While tests correctly identified spurious edges in simple cases (Asia, Sachs), they failed in cases where the spurious edge was confounded by other variables (Child, Insurance). A significant test result confirms dependence, not causation; thus, a "confirmed" edge might still be spurious if the conditioning set is insufficient.

6.1.3 RQ3: Practitioner guidance

What practical guidance can we offer practitioners for selecting algorithms and validating their causal assumptions?

We synthesise our findings into three recommendations:

1. **Match Algorithm to Data:** Use NOTEARS for continuous, linear-like data. Use PC for discrete data or when speed is critical. Use GES only if computational resources allow and precision is less critical than recall.
2. **Trust Skeletons, Verify Directions:** Algorithms are much better at finding connections than directions. Treat automated orientations as hypotheses to be verified by domain knowledge, especially for edges not part of v-structures.
3. **Iterative Validation:** Do not treat the output of an algorithm as final. Use CI tests to check for missing edges, but be skeptical of "significant" edges that lack a mechanism. Use bootstrap stability to filter out fragile edges. Practitioners should leverage validation tools (e.g., DAGitty, CausalNex) to test implied independencies.

6.2 The Analyst-in-the-Loop Paradigm

A recurring theme in our results is the limitation of fully automated discovery. Even the best algorithms achieve F_1 scores around 0.75 on complex networks, meaning one in four edges is incorrect. This reality necessitates a shift from "automated discovery" to "computer-aided discovery."

6.2.1 A practitioner decision guide for method selection

The benchmark suggests that no single method dominates across all desiderata (accuracy, directionality, runtime, and robustness to misspecification). A practical workflow is therefore to choose an initial method family based on *data type* and *intended use*, and then validate the output using stability and diagnostic checks.

Table 19: Decision guide for choosing a discovery approach in analyst-in-the-loop settings. Recommendations reflect the empirical patterns in Section 5 and common assumptions in the literature.

Situation	Practical constraints	Recommended approach
Continuous data; primary need is adjacency discovery	Many variables; runtime matters	Start with PC (Fisher-Z) for a fast skeleton; interpret directions via the CPDAG. Consider a score-based refinement only if runtime permits.
Discrete or discretised data; directions are used for downstream interventions	Risk of score/CI-test mismatch	Report both skeleton and directed metrics; treat directed edges as hypotheses. Use edge-level stability (bootstrap) for critical edges rather than trusting the full orientation.
Need a single DAG output (e.g., simulation or planning)	Requires a fully oriented graph	Use an optimisation-based method (NOTEARS/COSMO) but explicitly validate against equivalence-class uncertainty; avoid presenting the DAG as uniquely identified from observational data.
Concern about latent confounding	Unobserved common causes likely	Prefer methods that target richer graphical objects (e.g., FCI) and evaluate accordingly; do not interpret a DAG estimate as causal without additional assumptions.

6.2.2 Software ecosystem considerations

Practitioner workflows are often constrained not only by theory but by tooling: `Tetrad` provides a broad suite of constraint- and score-based methods with mature visualisation support [15]; `bnlearn` and `pcalg` offer stable implementations of classical Bayesian network and constraint-based algorithms in R [12, 11]; `CausalDiscoveryToolbox` provides a unified interface to both classical and modern methods in Python [13]; and `DAGitty` supports transparent DAG construction and adjustment-set reasoning for analysts [8]. In practice, the choice of ecosystem influences default CI tests, score implementations, and graph representations (DAG vs CPDAG vs PAG), which in turn affects what it means to "validate" a learned structure.

6.2.3 Common pitfalls in analyst-facing causal discovery

The experiments and literature suggest several recurring pitfalls:

- **Over-interpreting directed edges from observational data.** Many directions are unidentifiable within a Markov equivalence class.
- **Ignoring mismatch between assumptions and data.** CI tests and scores have implicit distributional assumptions; discretisation can distort these.
- **Using a single metric as a proxy for correctness.** Skeleton F_1 and SHD answer different questions; directed metrics can penalise unidentifiable choices.
- **Failing to run stability diagnostics for key edges.** Global accuracy does not guarantee that a particular domain-critical edge is reliable.
- **Treating misspecification as binary.** Real analyst graphs mix missing, spurious, and misoriented edges; sensitivity should consider multiple modes.

6.2.4 Deployment checklist for analyst-in-the-loop use

A concise checklist for deploying causal discovery in an analyst workflow is:

1. **State the estimand and the role of the graph.** Is the graph used for explanation, adjustment selection, or intervention planning?
2. **Document assumptions.** Markov + faithfulness, absence/presence of latent confounding, linearity/nonlinearity, and data type.
3. **Run at least two complementary families.** E.g., PC (constraint-based) and GES (score-based) or an optimisation method, and compare skeletons.
4. **Report equivalence-aware outputs.** Where applicable, provide CPDAG/PAG views rather than forcing a single DAG narrative.
5. **Validate key edges with diagnostics.** Use CI tests, bootstrap stability, and domain plausibility checks for the edges that drive decisions.
6. **Treat outputs as hypotheses.** Close the loop with expert review or interventional/temporal evidence where possible.

In the **Analyst-in-the-Loop** paradigm, the algorithm is not an oracle but a hypothesis generator. The analyst's role shifts from defining the graph to *critiquing* it.

- **Prior Knowledge as Constraints:** Instead of running PC on a blank slate, analysts should enforce known edges (e.g., `Age` $\rightarrow$ `Disease`) as constraints. This reduces the search space and improves orientation accuracy.
- **Interactive Refinement:** Tools should present the "most uncertain" edges to the user. For example, if an edge appears in 50% of bootstrap runs, the system should ask: "Is there a mechanism for X causing Y?"
- **Falsification over Confirmation:** The goal should be to falsify the graph. If the data screams that X and Y are dependent, and the graph says they are independent, the graph is wrong. Our sensitivity analysis shows that this falsification signal is strong and reliable.

6.3 Robustness to Assumption Violations

Our primary benchmarks assumed causal sufficiency and faithfulness. To probe robustness, we conducted a preliminary stress test by introducing a latent confounder in the Asia network. We merged `LungCancer` and `Bronchitis` into a single unobserved variable L that causes `Dyspnea` and is caused by `Smoking`. Running PC on this confounded data resulted in a spurious edge `Smoking` $\rightarrow$ `Dyspnea` (representing the path through L). This confirms that without algorithms designed for latent variables (like FCI), standard methods will infer direct causation where only confounding exists. This highlights the critical need for domain experts to assess the plausibility of causal sufficiency in their specific application.

6.4 Case Study: End-to-End Analyst Workflow

To illustrate how these findings translate into practice, we present a hypothetical workflow for an analyst building a causal model for the Asia dataset.

1. **Initial Discovery:** The analyst runs PC on the data. The algorithm outputs a skeleton with 7 edges but leaves the `Smoking-LungCancer` edge undirected.
2. **Diagnostic Testing:** The analyst suspects a link between `Smoking` and `LungCancer`. They perform a conditional independence test: `Smoking` $\perp$ `LungCancer` | `Bronchitis`? The test rejects independence ($p < 0.001$), confirming a link exists.
3. **Orientation Verification:** The analyst checks the orientation. PC oriented `LungCancer` $\rightarrow$ `Dyspnea`. The analyst verifies this against medical knowledge (cancer causes shortness of breath).
4. **Refinement:** The analyst notices an unexpected edge `VisitAsia` $\rightarrow$ `LungCancer` suggested by GES in a separate run. They run a bootstrap analysis (100 resamples) and find this edge appears in only 30% of samples. They discard it as spurious.
5. **Final Model:** The analyst combines the robust edges from PC with domain-verified orientations to produce the final DAG.

This workflow demonstrates how algorithmic output serves as a starting point for an iterative, evidence-based construction process.

6.5 Threats to Validity

We identify several threats to the validity of our conclusions.

6.5.1 Internal Validity

Internal validity concerns whether the observed effects are due to the manipulated variables (algorithms, datasets) rather than confounding factors.

- **Implementation Differences:** We used standard libraries (`causal-learn`, `CausalNex`). Differences in default hyperparameters (e.g., stopping criteria, score penalties) could confound algorithm comparisons. We mitigated this by documenting all settings in Table 6 and using consistent evaluation metrics.
- **Randomness:** Algorithms like COSMO and NOTEARS (via initialisation) are stochastic. We controlled for this by fixing random seeds, but a single run per dataset (for the main benchmark) might not capture the full distribution of performance. However, the large sample size ($n = 1000$) reduces variance.

6.5.2 Construct Validity

Construct validity concerns whether our metrics actually measure "causal discovery success."

- **SHD vs SID:** We relied primarily on SHD and F_1. These are structural metrics. A graph with low SHD could still yield poor causal effect estimates if the few errors are on critical confounding paths (high Structural Intervention Distance). Our focus on structure learning justifies SHD, but it is a proxy, not a direct measure of downstream utility.
- **DAG vs CPDAG:** We evaluated directed metrics by converting CPDAGs to DAGs. This penalises algorithms for not orienting edges that are theoretically unidentifiable. While we reported skeleton metrics to balance this, the directed metrics should be interpreted as a "best-case guess" rather than a rigorous test of identifiability.

6.5.3 External Validity

External validity concerns generalisability.

- **Synthetic Data:** We used semi-synthetic data generated from known DAGs. Real-world data often violates faithfulness, contains measurement error, and has latent confounders. Our results likely overestimate performance compared to real applications.
- **Network Selection:** We used five standard benchmarks. While diverse, they do not cover all possible topologies (e.g., scale-free networks, grids). Performance on ultra-large graphs (1000+ nodes) remains untested here.

6.6 Limitations

Beyond validity threats, the study has specific scope limitations.

- **Assumption of Sufficiency:** We assumed no latent confounders. In reality, unmeasured common causes are ubiquitous. Algorithms like FCI (Fast Causal Inference) are designed for this but were outside our scope.

- **Linearity:** Our continuous data generation (Sachs) and algorithms (NOTEARS, COSMO) assumed linearity. Nonlinear causal discovery is a distinct and active field (e.g., using neural networks or kernels) that we did not explore.
- **Single-Edge Misspecification:** Our sensitivity analysis perturbed only one edge at a time. Real analyst errors are likely multiple and correlated. Detecting complex structural mismatches remains an open challenge. Additionally, we relied on a limited number of runs (single seed) and relatively small network sizes (max 37 nodes).

6.7 Ethical and Responsible Use

Causal discovery tools are powerful but dangerous if misused. The ability to infer causality from data is often oversold, leading to "causal washing" of observational findings.

6.7.1 The Risk of False Confidence

The most significant risk is that practitioners will treat the output of an algorithm as "The Truth." As our results show, even the best algorithms make errors. Blindly trusting a learned DAG can lead to incorrect policy interventions. For example, in healthcare, a learned graph might suggest that a treatment causes recovery, when in fact both are caused by socioeconomic status (a confounder). Intervening on the treatment based on this graph would be ineffective and potentially harmful.

6.7.2 Algorithmic Bias

If the training data contains selection bias (e.g., healthcare access disparities), the learned causal graph will encode that bias as a structural mechanism. For instance, if a minority group receives less aggressive treatment due to systemic bias, the algorithm might learn a causal link $\texttt{Race} \rightarrow \texttt{Treatment}$. While descriptively accurate of the *system*, interpreting this as a "natural" causal mechanism could entrench the bias. Practitioners must scrutinise the data collection process before running these algorithms.

6.7.3 Dual Use

Causal discovery can also be used for manipulation. If an algorithm identifies the causal drivers of user behaviour (e.g., "Outrage causes Engagement"), this knowledge can be weaponised to design addictive platforms. The ethical deployment of these tools requires a commitment to beneficence and non-maleficence.

6.8 Lessons Learned

Synthesising the results from our benchmarking and sensitivity analyses, we distill the following key lessons for the causal discovery community:

1. **Skeletons are robust, directions are fragile.** Across all datasets, algorithms agreed far more on the presence of edges than on their direction. This suggests that the "skeleton" should be the primary output of automated tools, with directions treated as tentative suggestions requiring validation.

2. **Linearity is a double-edged sword.** NOTEARS's assumption of linearity allowed it to achieve perfect performance on Sachs but caused it to fail on discrete data. Analysts must rigorously test for linearity before deploying such methods.

3. **Data-driven falsification works.** The most encouraging finding is the reliability of conditional independence tests in detecting missing edges. This confirms that while we may not be able to *confirm* a graph, we can effectively *falsify* it.

4. **Speed matters for interactivity.** The orders-of-magnitude difference in runtime between PC/COSMO and GES implies that only the former are suitable for real-time, interactive analyst tools.

6.9 Future Research Directions

This thesis opens several avenues for future research:

- **Benchmarking with Latent Confounders:** Extending this framework to include FCI and other PAG-learning algorithms on datasets with hidden variables is a critical next step.
- **Nonlinear Discrete Discovery:** Developing and benchmarking algorithms that handle discrete data without assuming linearity (e.g., discrete NOTEARS variants or neural-based discrete estimators) is needed to close the performance gap on datasets like Insurance.
- **Automated Mis-specification Correction:** While we showed how to *detect* errors, developing agents that can automatically *repair* the graph based on these signals (e.g., by adding the edge with the highest CI test statistic) would be a valuable contribution.
- **Human-Subject Studies:** Conducting user studies to see if providing CI-based feedback actually helps human analysts improve their causal models would validate the practical utility of our "Analyst-in-the-Loop" paradigm.

7 Conclusion

This thesis studied the robustness of causal structure learning under a controlled form of analyst mis-specification. Across five benchmark networks and four representative algorithm families (constraint-based, score-based, and optimisation-based), the results support three overarching conclusions.

First, skeleton recovery is often strong, but directionality remains fragile. Across datasets, several methods achieve high skeleton F_1 (Table 7), yet directed metrics remain substantially lower (Table 11), with large skeleton–directed gaps (Table 12). This reinforces a principled limitation of purely

observational structure learning: many directions are only identifiable up to Markov equivalence, and additional modelling assumptions or interventional information are required for reliable orientation. In practical terms, learned directions should be treated as hypotheses unless the identifiability conditions are explicitly justified.

Second, different algorithm families fail in different ways, which matters for downstream use. Error decomposition (Table 13) shows that some methods primarily overfit by adding extra edges, while others incur more missing or reversed edges. These failure modes have distinct implications for analyst workflows: extra edges can create spurious adjustment paths, missing edges can omit real causal pathways, and reversals can flip intervention recommendations. Accordingly, no single graph-level metric is sufficient for analyst-facing validation.

Third, misspecification detection benefits from edge-level diagnostics and stability checks. The misspecification experiments demonstrate that conditional independence tests can highlight inconsistencies between an analyst graph and the data (Table 17), but that global performance does not guarantee reliability on practically critical edges. Bootstrap stability of the key missing edge (Table 19) provides a compact, actionable diagnostic: it directly quantifies whether a specific correction is consistently supported by the data under the chosen algorithm.

Practical contribution. Beyond benchmark comparison, the thesis proposes a practitioner-oriented workflow: combine complementary algorithm families, report equivalence-aware outputs, and validate decisions using edge-level diagnostics for the causal relations that matter most to the domain. This framing aligns causal discovery with its most reliable role in analyst contexts: not as an oracle for a unique causal DAG, but as a disciplined hypothesis generator that can be iteratively refined with expert knowledge and, where possible, interventional or temporal evidence.

Future directions. Future work should broaden the misspecification scenarios beyond single-edge perturbations, incorporate explicit latent-confounding benchmarks (evaluated with appropriate graphical targets), and study robustness under more realistic data complexities (measurement error, selection bias, and distribution shift). Methodologically, integrating stability selection, equivalence-class-aware metrics, and richer uncertainty quantification into analyst tools remains an open and practically important direction.

7.1 Reproducibility and artifact availability

All code, data, and analysis scripts are available in the `CausalWhatNot` repository. Experiments were run using Python 3.11 with the following key dependencies: `causal-learn` 0.1.3.8 (PC, GES), `gCastle` 1.0.3 (NOTEARS), `numpy` 1.26, `scipy` 1.11, and `pandas` 2.0. Benchmark networks are included in the repository under `causal_benchmark/data/`. Random seeds are fixed via a configurable `config.yaml` file to ensure reproducibility. Results can be regenerated by running `python experiments/run_benchmark.py`.

References

[1] Judea Pearl. “Causal diagrams for empirical research.” *Biometrika*, 82(4):669–688, 1995.

[2] Peter Spirtes, Clark Glymour and Richard Scheines. *Causation, Prediction, and Search*, 2nd edition. MIT Press, 2000.

[3] David M. Chickering. “Optimal structure identification with greedy search.” *Journal of Machine Learning Research*, 3:507–554, 2002.

[4] Clark Glymour, Kun Zhang and Peter Spirtes. “Review of causal discovery methods based on graphical models.” *Frontiers in Genetics*, 10:524, 2019.

[5] Marco Scutari, Clara E. Graafland and Juan M. Gutiérrez. “Who learns better Bayesian network structures: accuracy and speed of structure learning algorithms.” *International Journal of Approximate Reasoning*, 115:235–253, 2019.

[6] Alexander Reisach, Christof Seiler and Sebastian Weichwald. “Beware of the simulated DAG! Causal discovery benchmarks may be easy to game.” In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 27772–27784, 2021.

[7] Ankur Ankan, Inge M. N. Wortel and Johannes Textor. “Testing graphical causal models using the R package ‘dagitty‘.” *Current Protocols*, 1(2):e45, 2021.

[8] Johannes Textor, Ben van der Zander, Mark S. Gilthorpe, Maciej Liškiewicz and G. Thomas H. Ellison. “Robust causal inference using directed acyclic graphs: the R package `dagitty`.” *International Journal of Epidemiology*, 45(6):1887–1894, 2016.

[9] Marco Scutari and Radhakrishna Nagarajan. “On identifying significant edges in graphical models of molecular networks.” *Artificial Intelligence in Medicine*, 57(3):207–217, 2013.

[10] Nir Friedman, Moises Goldszmidt and Abraham Wyner. “Data analysis with Bayesian networks: a bootstrap approach.” In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 196–205, 1999.

[11] 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.

[12] Marco Scutari. “Learning Bayesian networks with the `bnlearn` R package.” *Journal of Statistical Software*, 35(3):1–22, 2010.

[13] 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.

[14] Yujia Zheng, Biwei Huang, Wei Chen, Joseph Ramsey, Mingming Gong, Ruichu Cai, Shohei Shimizu, Peter Spirtes and Kun Zhang. “causal–learn: causal discovery in Python.” *Journal of Machine Learning Research*, 25(60):1–8, 2024.

[15] Joseph D. Ramsey, Kun Zhang, Madelyn Glymour, Reuben S. Romero, Biwei Huang, Imme Ebert–Uphoff *et al.* “TETRAD — a toolbox for causal discovery.” In *Proceedings of the 8th International Workshop on Climate Informatics*, pages 1–4, 2018.

[16] Diego Colombo and Marloes H. Maathuis. “Order–independent constraint–based causal structure learning.” *Journal of Machine Learning Research*, 15:3741–3782, 2014.

[17] Ioannis Tsamardinos, Laura E. Brown and Constantin F. Aliferis. “The max–min hill–climbing Bayesian network structure learning algorithm.” *Machine Learning*, 65(1):31–78, 2006.

[18] Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen and Antti Kerminen. “A linear non–Gaussian acyclic model for causal discovery.” *Journal of Machine Learning Research*, 7:2003–2030, 2006.

[19] Xun Zheng, Bryon Aragam, Pradeep Ravikumar and Eric P. Xing. “DAGs with NO TEARS: continuous optimisation for structure learning.” In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 9472–9483, 2018.

[20] Ignavier Ng, AmirEmad Ghassami and Kun Zhang. “On the role of sparsity and DAG constraints for learning linear DAGs.” In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 17943–17954, 2020.

[21] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu and Simon Lacoste–Julien. “Gradient–based neural DAG learning.” In *International Conference on Learning Representations (ICLR)*, 2020.

[22] Shengyu Zhu, Ignavier Ng and Zhitang Chen. “Causal discovery with reinforcement learning.” In *International Conference on Learning Representations (ICLR)*, 2020.

[23] Riccardo Massidda, Francesco Landolfi, Martina Cinquini and Davide Bacciu. “Differentiable causal discovery with smooth acyclic orientations.” In *Proceedings of the 40th International Conference on Machine Learning, Workshop on Differentiable Almost Everything*, 2023.

[24] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets.” *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[25] Jonas Peters and Peter Bühlmann. “Structural intervention distance (SID) for evaluating causal graphs.” *Neural Computation*, 27(3):771–799, 2015.

[26] Gideon Schwarz. “Estimating the dimension of a model.” *Annals of Statistics*, 6(2):461–464, 1978.

[27] David Heckerman, Dan Geiger and David M. Chickering. “Learning Bayesian networks: the combination of knowledge and statistical data.” *Machine Learning*, 20(3):197–243, 1995.

[28] Milton Friedman. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance.” *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[29] Peter B. Nemenyi. *Distribution-free multiple comparisons*. Ph.D. thesis, Princeton University, 1963.

[30] Thomas Verma and Judea Pearl. “Equivalence and synthesis of causal models.” In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1990)*, pages 255–270, 1990.

[31] Steffen L. Lauritzen and David J. Spiegelhalter. “Local computations with probabilities on graphical structures and their application to expert systems.” *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

[32] Ingo A. Beinlich, Henri J. Suermondt, R. Martin Chavez, and Gregory F. Cooper. “The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks.” In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine (AIME 1989)*, pages 247–256, 1989.

[33] Diego Colombo, Marloes H. Maathuis, Markus Kalisch, and Thomas S. Richardson. “Learning high-dimensional directed acyclic graphs with latent and selection variables.” *The Annals of Statistics*, 40(1):294–321, 2012.

[34] Amit Sharma and Emre Kiciman. “DoWhy: An end-to-end library for causal inference.” *arXiv preprint arXiv:2011.04216*, 2020.

[35] Nicolai Meinshausen and Peter Bühlmann. “High-dimensional graphs and variable selection with the Lasso.” *The Annals of Statistics*, 34(3):1436–1462, 2006.

[36] Juan M. Ogarrio, Peter Spirtes and Joe Ramsey. “A hybrid causal search algorithm for latent variable models.” In *Proceedings of the 8th International Conference on Probabilistic Graphical Models (PGM)*, pages 368–379, 2016.