

Benchmarking Causal Discovery Under Analyst Misspecification

Edgar Davtyan

January 3, 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 notify the analyst of such errors. Finally, we provide practitioner guidance on building and checking causal diagrams using contemporary software and algorithms.

Contents

Abstract	I
List of Figures	VI
List of Tables	VIII
List of Abbreviations	IX
1 Introduction	1
2 Theoretical Framework	2
2.1 Probabilistic Graphical Models	2
2.1.1 Directed Acyclic Graphs (DAGs)	2
2.1.2 The Causal Markov Condition	2
2.1.3 d-Separation	2
2.1.4 Faithfulness	3
2.1.5 Markov Equivalence Classes	3
2.2 Constraint-Based Discovery: The PC Algorithm	3
2.2.1 Algorithm Steps	3
2.2.2 Complexity and Stability	4
2.3 Score-Based Discovery: GES	4
2.3.1 Scoring Functions	4
2.3.2 Greedy Equivalence Search (GES)	4
2.4 Continuous Optimisation: NOTEARS	5
2.4.1 The Acyclicity Constraint	5
2.4.2 The Optimisation Problem	5
2.5 Regression-Based Discovery: COSMO	5
2.5.1 Smooth Orientation	5
2.5.2 Algorithm Logic	6
2.6 Evaluation Metrics	6
2.6.1 Structural Hamming Distance (SHD)	6
2.6.2 Structural Intervention Distance (SID)	6
2.6.3 Precision, Recall, and F1	6

2.7	Structural Causal Models and Interventions	7
2.8	Assumptions, identifiability, and scope	7
2.9	Conditional independence testing in this benchmark	8
2.10	Statistical tests for comparing algorithms	8
3	Related Work	8
3.1	Foundations of Causal Discovery	9
3.1.1	The Graphical Viewpoint	9
3.1.2	The Search and Score Viewpoint	9
3.2	The Benchmarking Crisis in Causal Discovery	9
3.2.1	Standardisation Efforts	9
3.2.2	The "Scale-Free" Trap	10
3.3	Model Criticism and Mis-specification	10
3.3.1	Global vs. Local Fit	10
3.3.2	Refutation and Stability	10
3.4	The Differentiable Discovery Revolution	10
3.4.1	Extensions and Limitations	11
3.5	Software Ecosystem	11
3.6	Positioning of this Thesis	11
4	Methods	12
4.1	Datasets and Data Generation	12
4.2	Algorithms and Settings	12
4.2.1	PC Algorithm	13
4.2.2	Greedy Equivalence Search (GES)	14
4.2.3	NOTEARS	14
4.2.4	COSMO	14
4.3	Benchmarking pipeline and post-processing	15
4.3.1	Detailed Pipeline Walkthrough	15
4.4	Mis-specification Protocols	16
4.5	Visualisations	17
5	Results	18
5.1	Benchmark Performance Overview	18

5.2	Dataset-by-Dataset Analysis	19
5.2.1	Asia (Discrete, 8 nodes, 8 edges)	19
5.2.2	Sachs (Continuous, 11 nodes, 17 edges)	20
5.2.3	Alarm (Discrete, 37 nodes, 46 edges)	21
5.2.4	Child (Discrete, 20 nodes, 25 edges)	21
5.2.5	Insurance (Discrete, 27 nodes, 52 edges)	21
5.3	Cross-Dataset Patterns	22
5.3.1	Data Type Effects on Algorithm Performance	22
5.3.2	The Skeleton vs Directed Gap	22
5.3.3	Runtime vs Accuracy	23
5.3.4	Algorithm Summary	25
5.3.5	Statistical Significance	25
5.4	Edge-Level Case Studies	25
5.4.1	Case Study 1: Asia - Smoking → LungCancer	25
5.4.2	Case Study 2: Sachs - PKA → Mek	26
5.5	Detecting Analyst Mis-specification	26
5.5.1	Conditional Independence Testing Results	27
5.5.2	Algorithm recovery of omitted edges	28
6	Discussion	29
6.1	Answers to Research Questions	29
6.1.1	RQ1: Benchmark accuracy across data types and network sizes	29
6.1.2	RQ2: Detecting omitted and spurious edges with CI tests	30
6.1.3	RQ3: Practitioner guidance	30
6.2	The Analyst-in-the-Loop Paradigm	30
6.3	Threats to Validity	31
6.3.1	Internal Validity	31
6.3.2	Construct Validity	31
6.3.3	External Validity	31
6.4	Limitations	32
6.5	Ethical and Responsible Use	32
6.5.1	The Risk of False Confidence	32
6.5.2	Algorithmic Bias	32

6.5.3	Dual Use	33
7	Conclusion	33
7.1	Future Work	33
7.1.1	Latent Confounding	33
7.1.2	Nonlinear and Mixed Data	33
7.1.3	Interventional Data	33
7.1.4	Automated Model Criticism	34
7.2	Reproducibility and artifact availability	34

List of Figures

1	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.	17
2	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.	17
3	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.	19
4	Precision–recall scatter plot for skeleton recovery. Each point represents one algorithm–dataset combination. Dashed curves show iso- F_1 contours.	20
5	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.	22
6	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.	24
7	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.	37
8	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.	38
9	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.	38
10	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.	39
11	Algorithm performance profiles averaged across datasets. Axes represent skeleton F_1 , directed F_1 , precision, recall and speed.	39

12	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.	40
13	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.	40
14	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). . .	41
15	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.	41
16	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.	42

List of Tables

2	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.	12
3	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.	13
4	Skeleton recovery performance. Precision, recall and F_1 treat edges as undirected. Bold indicates the best F_1 score for each dataset.	18
5	Directed edge recovery performance. Reversed edges count as errors.	23
6	Runtime in seconds. Bold indicates the fastest algorithm for each dataset.	24
7	Mis-specification scenarios. Missing edges are true causal links removed from the analyst's DAG; spurious edges are false links added.	26
8	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).	27
9	Algorithm performance in sensitivity analysis (vs. true graph).	28

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.

This thesis pursues three goals. First, we benchmark multiple structure–learning methods—PC, GES, NOTEARS and COSMO—on established toy 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 7 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 $V = \{X_1, \dots, X_d\}$ and a set of directed edges $E \subseteq V \times V$. An edge $(X_i, X_j) \in E$ is denoted as $X_i \rightarrow X_j$. 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 $\text{Pa}(X_i)$ in G .
- **Factorisation:** If (G, P) satisfies the Markov condition, the joint distribution factorises as:

$$P(X_1, \dots, X_d) = \prod_{i=1}^d P(X_i \mid \text{Pa}(X_i)) \quad (1)$$

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 $i \rightarrow m \rightarrow j$ or a fork $i \leftarrow m \rightarrow j$ such that the middle node m is in Z .

2. The path contains a collider $i \rightarrow m \leftarrow j$ 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 .

$$X \perp_P Y \mid Z \iff X \perp_G Y \mid Z \quad (2)$$

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

2.1.5 Markov Equivalence Classes

Two DAGs are Markov equivalent if they encode the same set of conditional independence relations. Verma and Pearl 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 $X \rightarrow Z \leftarrow Y$). 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 $k = 1, 2, \dots$. 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 $X \rightarrow Z \leftarrow Y$. Orient these edges.
3. **Orientation Propagation (Meek Rules):** Apply the following rules repeatedly until no more edges can be oriented:
 - **R1:** If $X \rightarrow Y - Z$ and X, Z are not adjacent, orient as $Y \rightarrow Z$ (to avoid creating a new v-structure).
 - **R2:** If $X \rightarrow Y \rightarrow Z$ and $X - Z$, orient as $X \rightarrow Z$ (to avoid cycles).
 - **R3:** If $X - Y \rightarrow Z$ and $X - Z$ and $X - W \rightarrow Z$ and $X - W$, orient $X \rightarrow Z$ (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.

2.4 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:

$$h(W) = \text{tr}(e^{W \circ W}) - d = 0 \quad (3)$$

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:

$$\min_W \ell(W; X) + \lambda \|W\|_1 \quad \text{subject to} \quad h(W) = 0 \quad (4)$$

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).

2.5 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 $X_i \rightarrow X_j$ 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].

$$\text{SHD}(G, \hat{G}) = |E \setminus \hat{E}| + |\hat{E} \setminus E| + \text{flips} \quad (5)$$

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 $X_i = f_i(\text{Pa}_i, U_i)$, 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: $P(X_1, \dots, X_p) = \prod_i P(X_i \mid \text{Pa}_i)$.

Interventions are represented by modifying structural equations. For example, an intervention $\text{do}(X_k = x)$ replaces the structural equation for X_k with the constant assignment $X_k := x$, producing an interventional distribution $P(\cdot \mid \text{do}(X_k = x))$. 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 6.3.

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

$$z = \frac{1}{2} \ln \left(\frac{1+r_{XY \cdot Z}}{1-r_{XY \cdot Z}} \right) \sqrt{n - |Z| - 3} \quad (6)$$

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.

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. 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).

3.2 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

4.1 Datasets and Data Generation

Our experiments use five canonical Bayesian network benchmarks summarised in Table 2. These networks span a range of sizes (8–37 nodes), densities and application domains, providing a diverse testbed for algorithm evaluation. Following the *CausalWhatNot* framework, each dataset is generated *semi-synthetically* from the known ground-truth graph structure to ensure full reproducibility. Concretely, we simulate observational samples from a linear-Gaussian structural equation model (SEM) in topological order, drawing random edge weights and independent Gaussian noise terms for each variable. For benchmarks treated as discrete (Asia, ALARM, Child, Insurance), we subsequently discretise the simulated variables into a fixed number of states using quantile-based binning so that algorithms can be paired with discrete conditional independence tests and scores. The Sachs network is treated as a continuous benchmark and is kept in its sampled Gaussian form. In all experiments we use $n = 1000$ observations per dataset and fix the random seed (seed=0) so that all tables, figures, and metrics are reproducible.

Table 2: 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 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 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 3 summarises each algorithm’s key characteristics and hyperparameter settings.

Table 3: 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 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 $k = 0, 1, \dots$, 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 $X \rightarrow Z \leftarrow Y$.
3. **Propagation:** Apply Meek rules to orient remaining undirected edges without creating cycles or new v–structures.

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.

4.2.2 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.

Implementation Details: We use `causal-learn`'s GES. For discrete datasets, we use the BDeu score (Bayesian Dirichlet equivalent uniform) with an equivalent sample size of 10. For continuous datasets, we use the BIC (Bayesian Information Criterion) score.

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.

4.2.3 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:

$$\min_W \ell(W; X) + \lambda \|W\|_1 \quad \text{subject to} \quad h(W) = \text{tr}(e^{W \circ W}) - d = 0 \quad (7)$$

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 and 0.25 for discrete data. The data is standardised before input.

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.

4.2.4 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 2 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 3. 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 the weakest edge (or an arbitrary edge if unweighted) 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. An edge $X \rightarrow Y$ in the learned graph counts as a true positive only if $X \rightarrow Y$ exists in the true graph. $Y \rightarrow X$ 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.

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 vary the sample size to examine how detection depends on data volume. 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.

4.5 Visualisations

Figure 1 visualises the Asia network used in our experiments. Figure 2 illustrates the benchmarking pipeline.

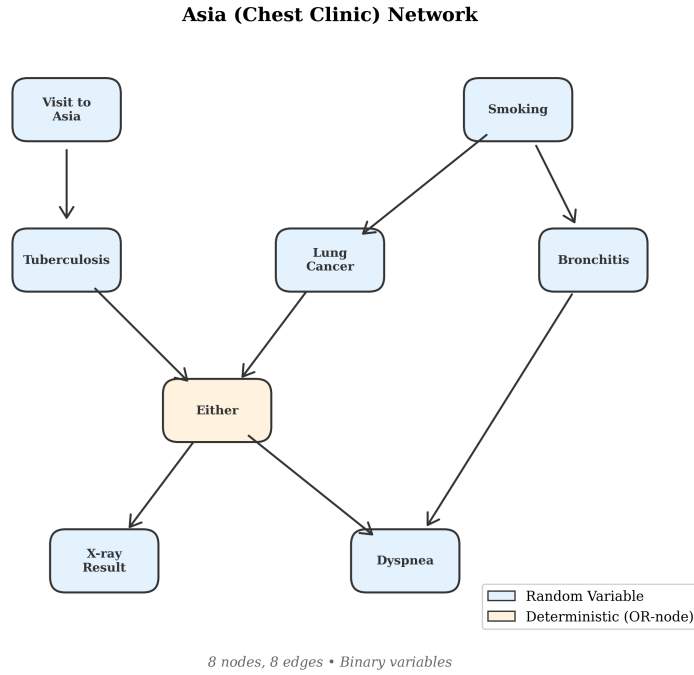


Figure 1: Structure of the Asia network. Nodes represent variables and arrows denote direct causal effects. This network is a standard benchmark for evaluating causal discovery algorithms.

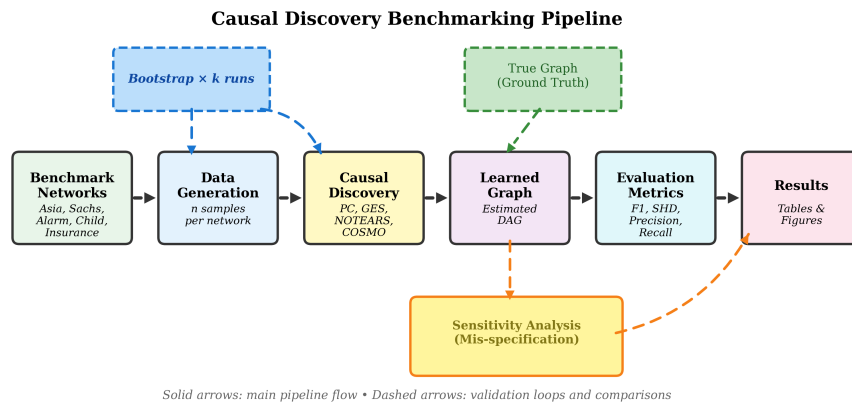


Figure 2: Benchmarking pipeline. Data are generated from benchmark networks, algorithms are run to learn the structure, metrics are computed, mis-specification analyses are performed, and bootstrap edge stability is recorded.

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.

5.1 Benchmark Performance Overview

Table 4 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 4: 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, 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 3 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 $F_1 \approx 0.71$.

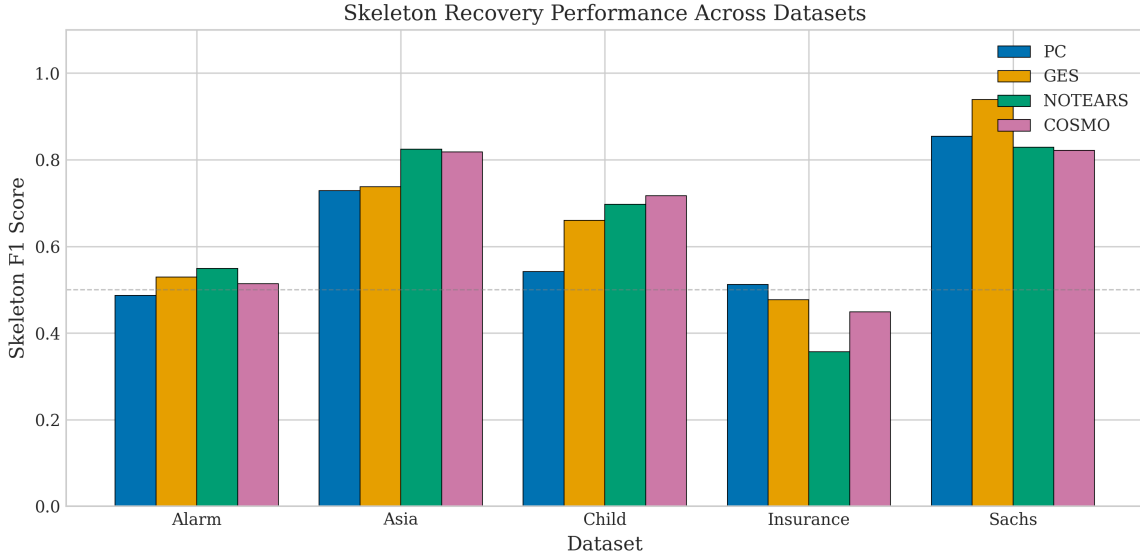


Figure 3: Skeleton F_1 scores by dataset and algorithm. The dashed horizontal line indicates $F_1 = 0.50$, representing performance no better than a naive baseline.

The precision–recall trade–off, shown in Figure 4, 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.

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

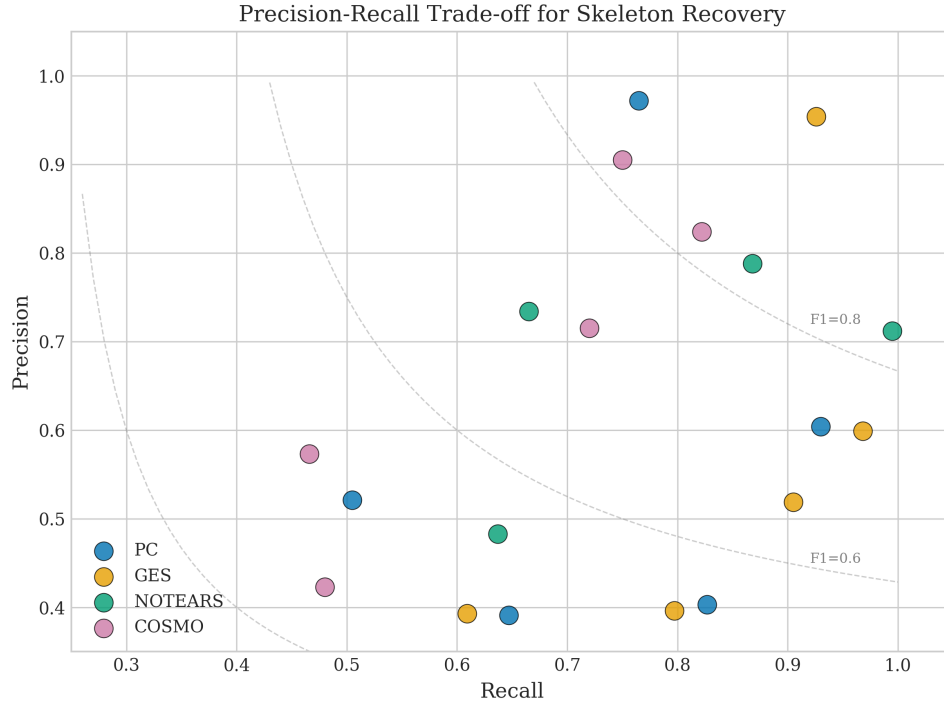


Figure 4: Precision–recall scatter plot for skeleton recovery. Each point represents one algorithm–dataset combination. Dashed curves show iso- F_1 contours.

achieved perfect recall (1.0) but at the cost of lower precision (0.73), suggesting it included some spurious edges.

- **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.

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.
- **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.
- **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.
- **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.
- **Takeaway:** Complex discrete networks remain an open challenge for observational causal discovery, especially at moderate sample sizes.

5.3 Cross-Dataset Patterns

5.3.1 Data Type Effects on Algorithm Performance

Before examining specific cross-dataset patterns, we first examine how data type influences algorithm performance. Figure 5 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).

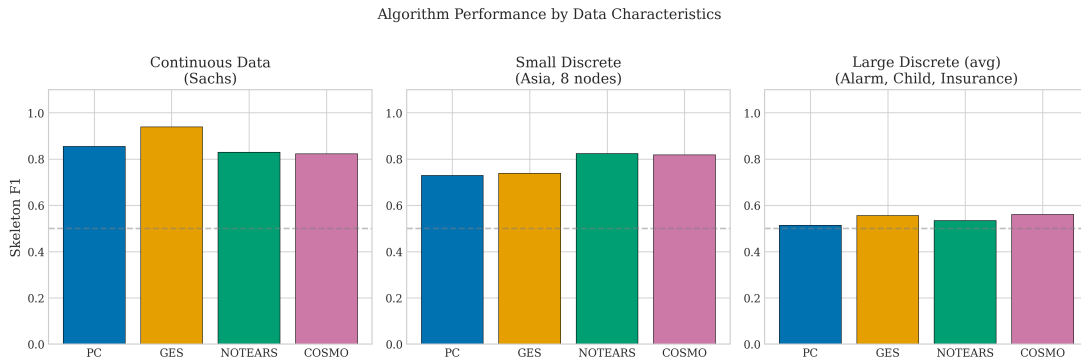


Figure 5: Algorithm performance stratified by data type. Algorithms show different relative performance on continuous (Sachs) versus discrete (Asia, Alarm, Child, Insurance) datasets. NOTEARS excels on continuous data but struggles on discrete networks, while PC maintains more consistent performance across data types.

5.3.2 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 5 reports directed precision, recall and F_1 , which penalise reversed edges as errors.

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 6 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

Table 5: 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

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 7 breaks down the Structural Hamming Distance (SHD) into false positives (extra edges), false negatives (missing edges), and orientation reversals.

Figure 8 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.

5.3.3 Runtime vs Accuracy

Runtime varies dramatically across algorithms. Table 6 reports execution times for each algorithm–dataset pair.

PC is one of the fastest algorithms on every dataset, completing in under 30 seconds even on the

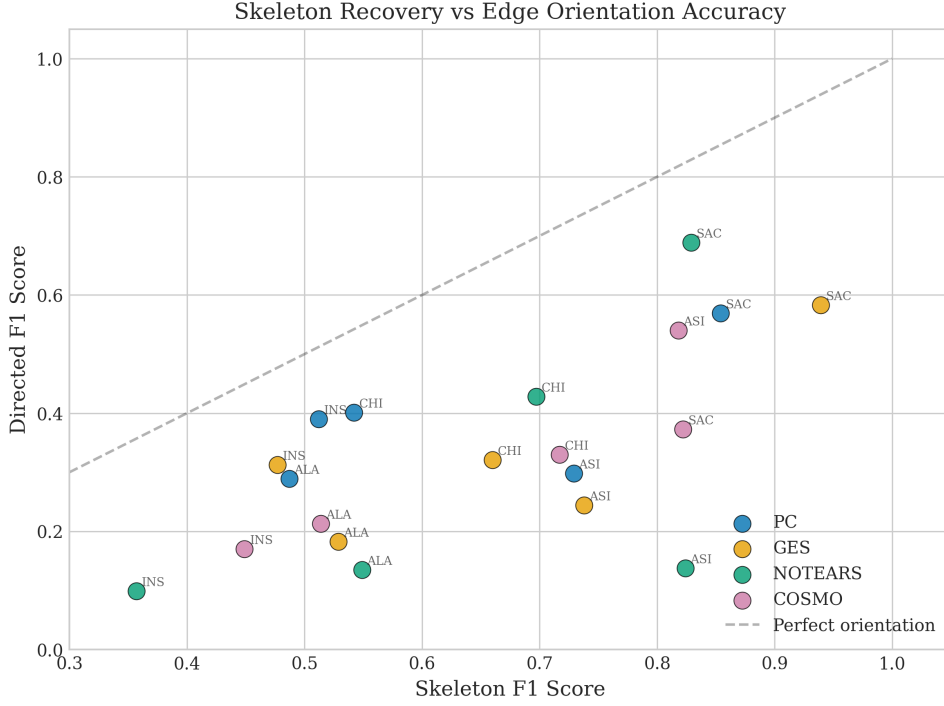


Figure 6: Skeleton F_1 versus directed F_1 . Points below the diagonal indicate orientation errors. Dataset abbreviations: ASI = Asia, SAC = Sachs, ALA = Alarm, CHI = Child, INS = Insurance.

Table 6: 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

largest networks. GES is generally the slowest, often by an order of magnitude or more; on Alarm, it requires nearly five minutes. 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.

Figure 9 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 10 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.

5.3.4 Algorithm Summary

Figure 11 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). 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.

5.3.5 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 $\chi_F^2 = 3.24$ 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 ($\chi_F^2 = 2.04$, $p = 0.564$), 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 12 summarises average ranks for skeleton F_1 and is included as a descriptive visual aid.

5.4 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.

5.4.1 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 $\text{Smoking} \rightarrow \text{LungCancer} \rightarrow \text{Dyspnea}$. 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.

5.4.2 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.

5.5 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 7 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 7: 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

5.5.1 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 8 reports the test statistics and p –values.

Table 8: 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.

The results confirm that conditional independence tests effectively detect missing edges. In all five datasets, the test statistic for the omitted edge is 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.

- **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.

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 13 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 14 visualises these results using the negative log p -value, which serves as a proxy for signal strength.

5.5.2 Algorithm recovery of omitted edges

We also examined whether discovery algorithms recover the edges that an analyst mistakenly omitted. Table 9 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 9: 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 15 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 16 provides a visual summary of whether each algorithm successfully recovered the specific edge that was omitted by the analyst.

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.

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."

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., $\text{Age} \rightarrow \text{Disease}$) 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 Threats to Validity

We identify several threats to the validity of our conclusions.

6.3.1 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 3 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.

6.3.2 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.

6.3.3 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.

6.4 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.

6.5 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.

6.5.1 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.

6.5.2 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 $Race \rightarrow Treatment$. 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.

6.5.3 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.

7 Conclusion

We developed a reproducible benchmarking framework that evaluates causal discovery algorithms on accepted toy networks and examined how analyst mis-specification can be detected. Five benchmark networks (Asia, Sachs, ALARM, Child, and Insurance) provide a useful testbed for comparing PC, GES, NOTEARS and COSMO. Our mis-specification experiments show that conditional independence testing and bootstrap stability can notify practitioners when their assumed graphs are wrong. We also surveyed a range of open-source DAG drawing and analysis tools to guide practitioners.

7.1 Future Work

This thesis opens several avenues for future research.

7.1.1 Latent Confounding

The most pressing extension is to relax the causal sufficiency assumption. Future benchmarks should include algorithms like FCI and RFCI that learn Partial Ancestral Graphs (PAGs), which can represent latent confounders. Evaluating how well these methods distinguish direct causation from confounding in finite samples is critical for real-world adoption.

7.1.2 Nonlinear and Mixed Data

Real data is rarely purely linear-Gaussian or purely discrete. Mixed data (containing both continuous and categorical variables) poses a challenge for many implementations. Benchmarking algorithms that handle mixed data natively (e.g., PC with conditional Gaussian tests) would be valuable. Similarly, evaluating nonlinear methods (like GraN-DAG) on non-linear data generators would provide a more realistic picture of modern capabilities.

7.1.3 Interventional Data

We focused on observational data. However, the "gold standard" of causality is intervention. Future work should explore "active learning" scenarios where the algorithm can request interventions (experiments) to resolve Markov equivalence classes. Benchmarking how many interventions are needed to fully orient a graph would be a significant contribution.

7.1.4 Automated Model Criticism

We showed that CI tests can flag errors. A natural next step is to automate the repair process. Can we build an "AI Analyst" that not only flags a missing edge but proposes the most likely candidate based on the data and global graph structure? Integrating Large Language Models (LLMs) with causal discovery to propose mechanisms for discovered edges is a frontier worth exploring.

7.2 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] Diviyani 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.

SHD Breakdown: False Positives, False Negatives, and Reversals

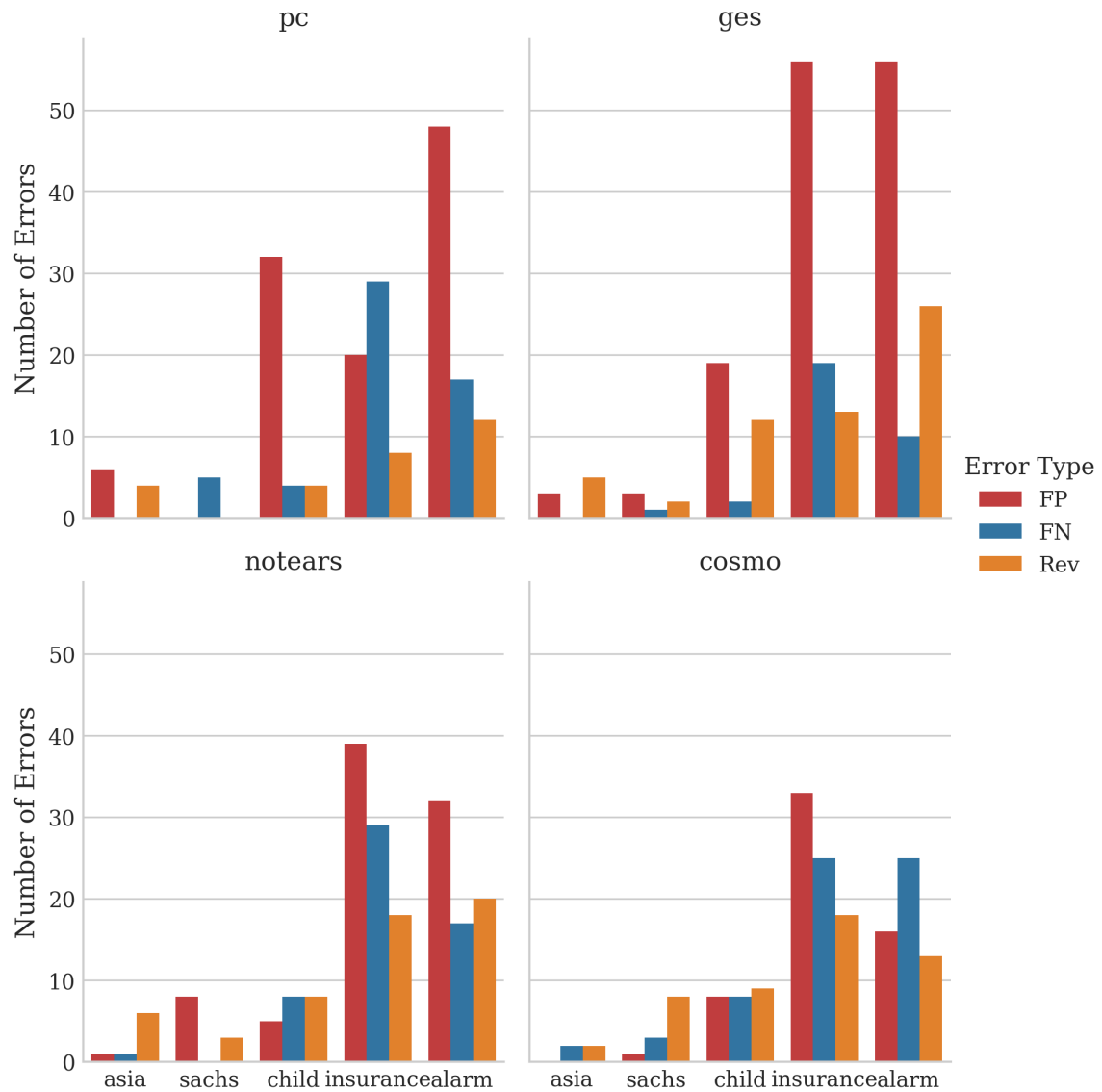


Figure 7: Breakdown of structural errors by type: False Positives (Extra), False Negatives (Missing), and Reversals. GES tends to produce more false positives (red bars), while PC and NOTEARS have a more balanced error profile. Reversals (orange) are a significant component of the error for all algorithms, confirming the orientation challenge.

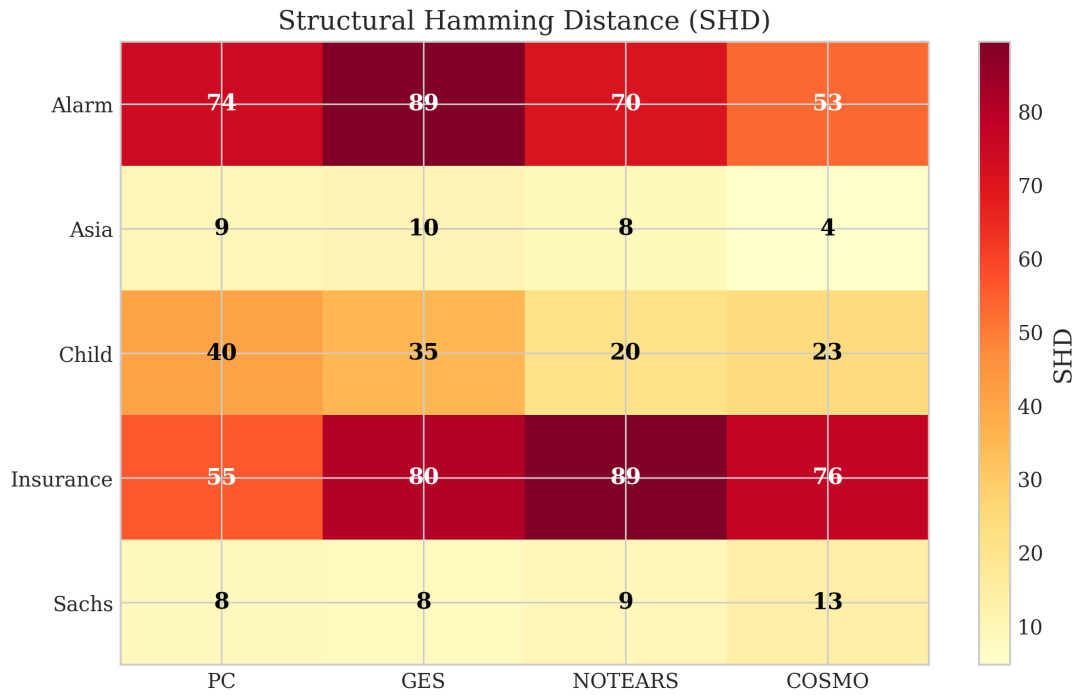


Figure 8: Heatmap of Structural Hamming Distance across algorithms and datasets. Darker colors indicate higher SHD (more errors). The matrix reveals that Alarm and Insurance are universally challenging (darker columns), while algorithm performance varies considerably by dataset. NOTEARS shows the highest variance, with near-zero SHD on Sachs but high SHD on Insurance.

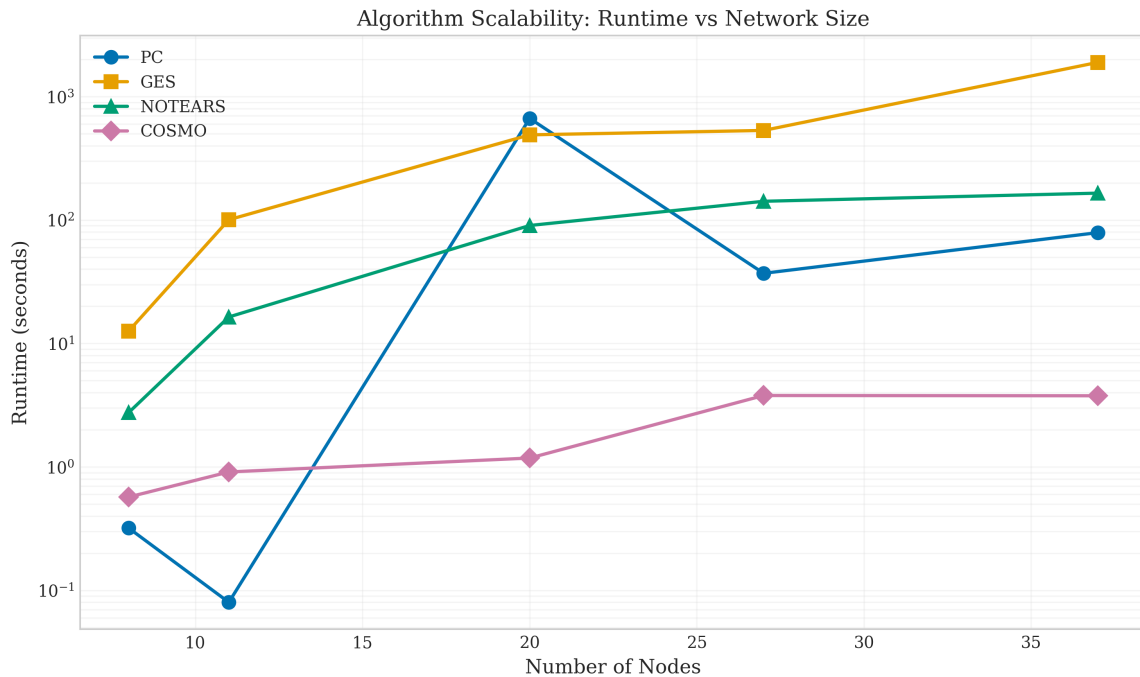


Figure 9: Runtime comparison across datasets (log scale). PC and COSMO are consistently faster than GES and NOTEARS. Note the logarithmic y-axis; GES is orders of magnitude slower on larger networks.

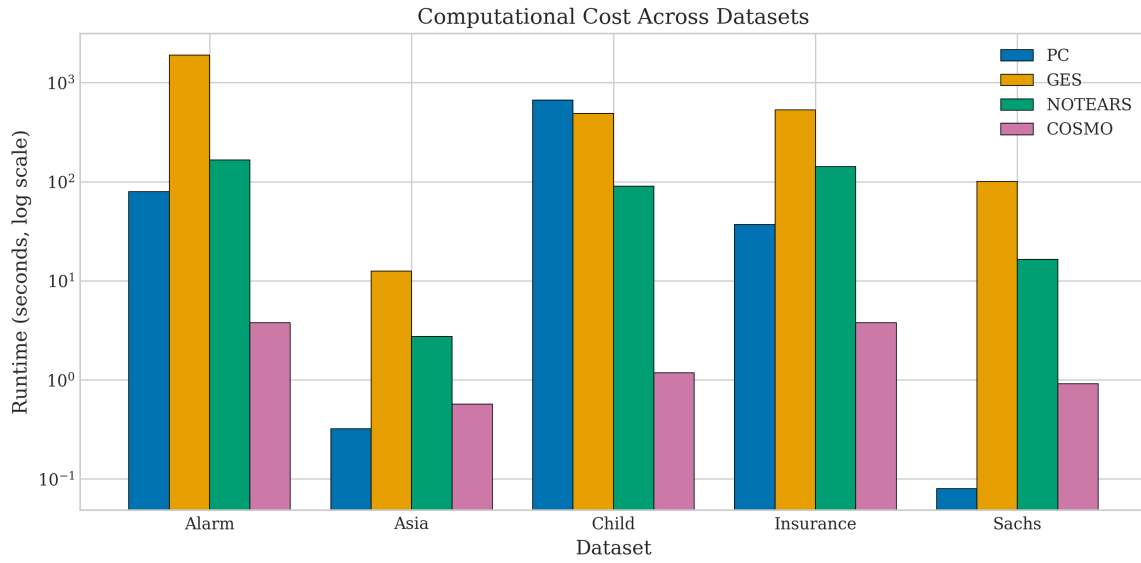


Figure 10: Direct runtime comparison across algorithms and datasets. Bar heights represent execution time in seconds. GES consistently requires the most time, while PC and COSMO are fastest across all networks.

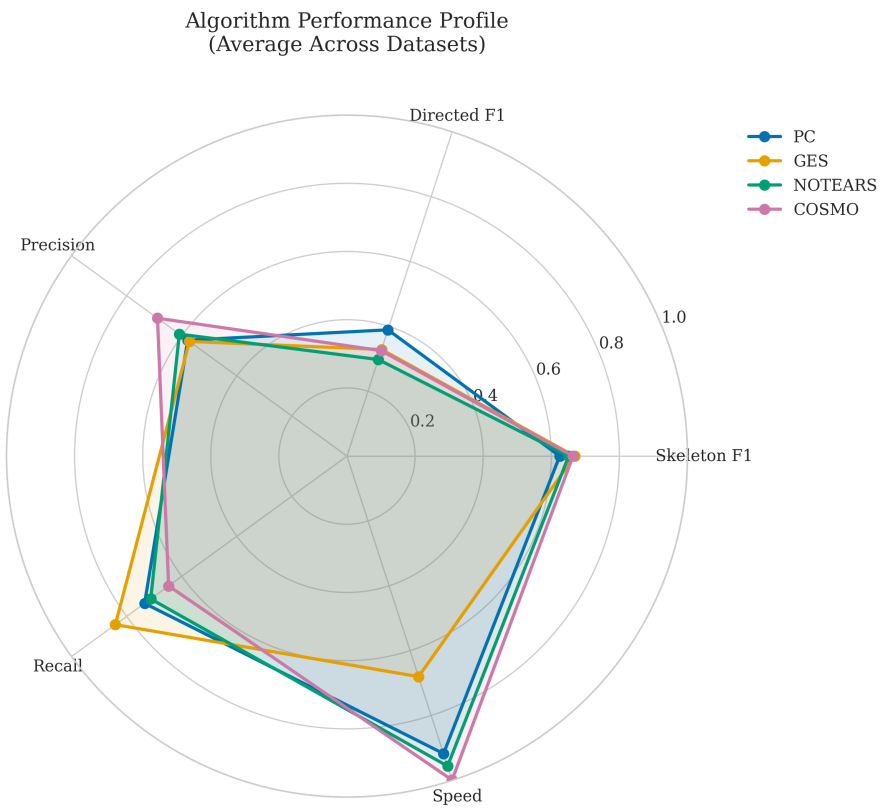


Figure 11: Algorithm performance profiles averaged across datasets. Axes represent skeleton F_1 , directed F_1 , precision, recall and speed.

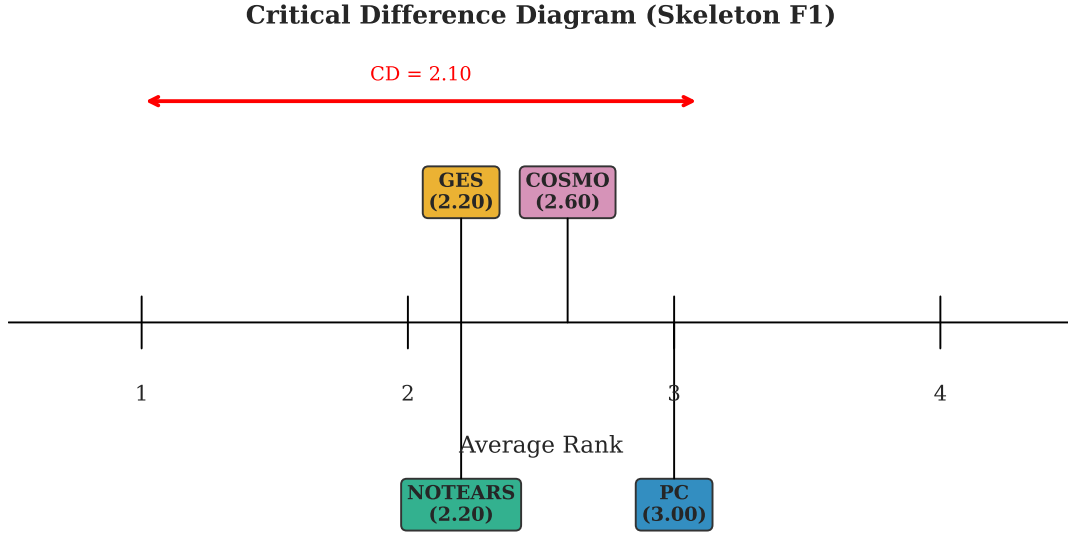


Figure 12: Critical difference diagram for skeleton F_1 . Algorithms are ranked by average performance across datasets, with rank 1 being best. The horizontal bar shows the Nemenyi critical difference (CD) at $\alpha = 0.05$. Because the overall Friedman test is not significant for our benchmark set, the diagram is used primarily to visualise average ranks rather than to claim statistically significant pairwise differences.

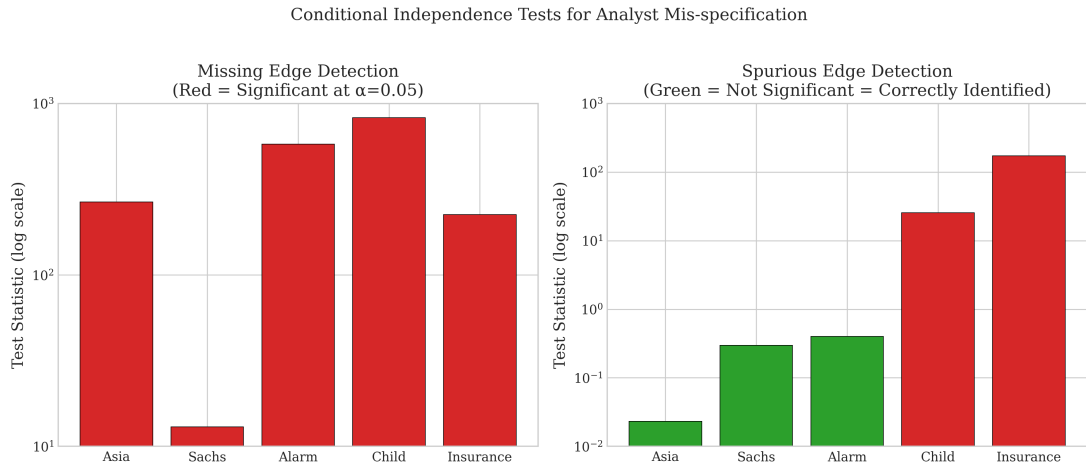


Figure 13: Conditional independence test results for sensitivity analysis. Compares test statistics for missing edge detection (where high values indicate correct detection) versus spurious edge detection (where low values indicate correct identification of unnecessary edges). Error bars represent confidence intervals where applicable.

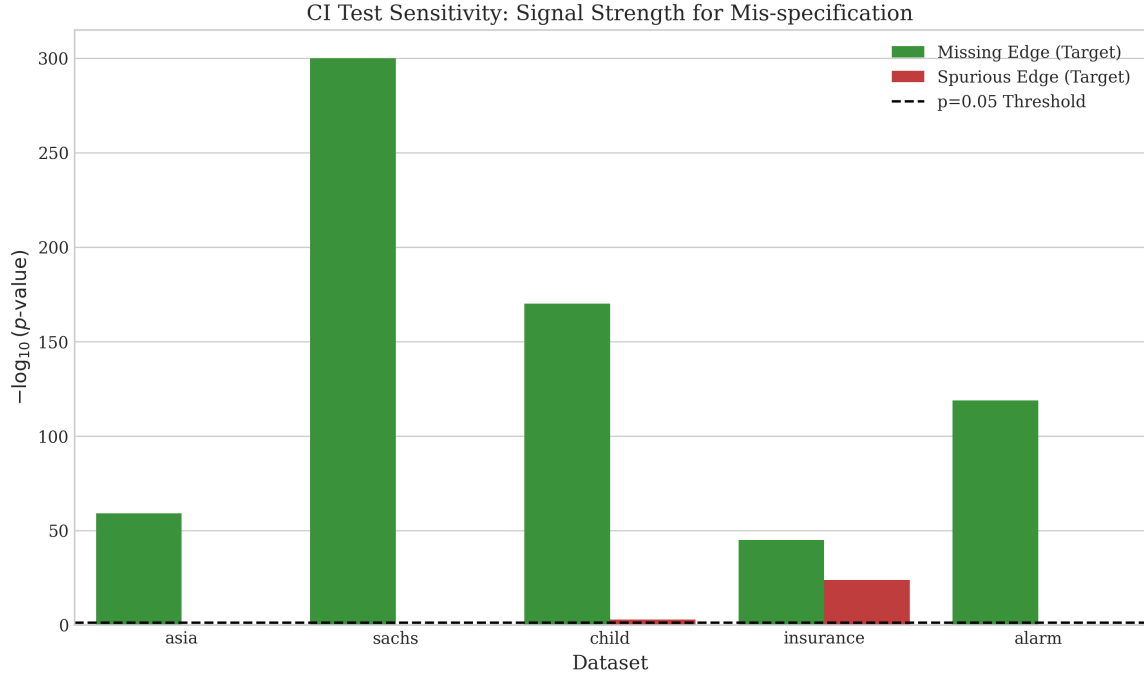


Figure 14: Signal strength of conditional independence tests for mis-specification detection. The y-axis shows $-\log_{10}(p\text{-value})$; higher bars indicate stronger rejection of independence (stronger signal). The dashed line represents the $\alpha = 0.05$ threshold. Missing edges (green) consistently produce strong signals. Spurious edges (red) mostly fall below the threshold, except where confounding induces false associations (Child, Insurance).

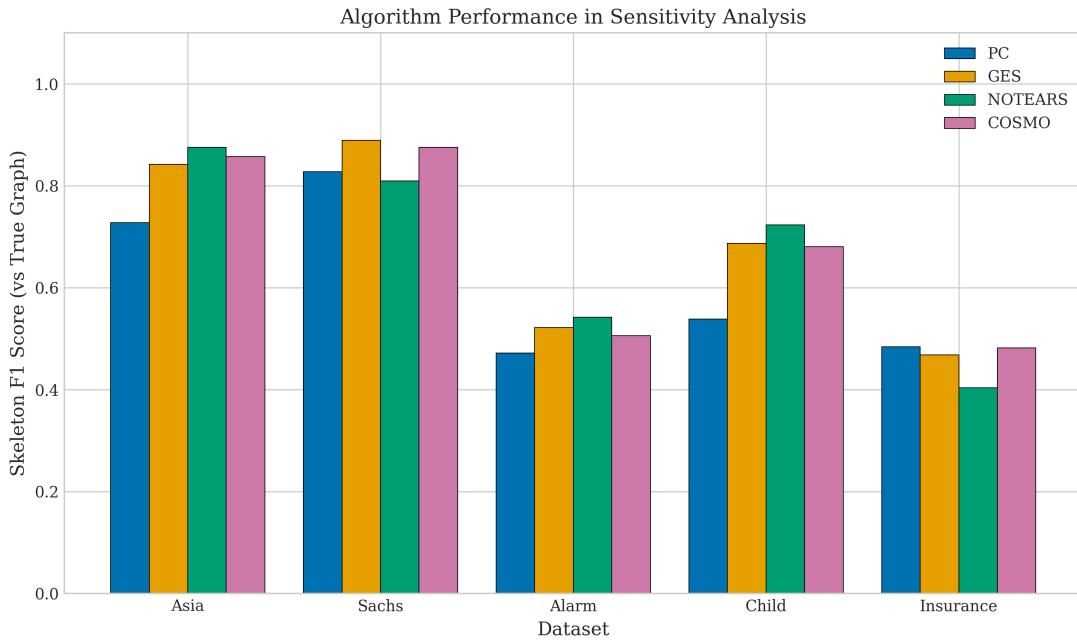


Figure 15: Algorithm performance comparison across sensitivity analysis scenarios. Shows how well each algorithm performs when data is generated from the true graph but evaluated against different misspecification types (missing edges vs spurious edges). Performance metrics indicate algorithms' robustness to analyst errors.

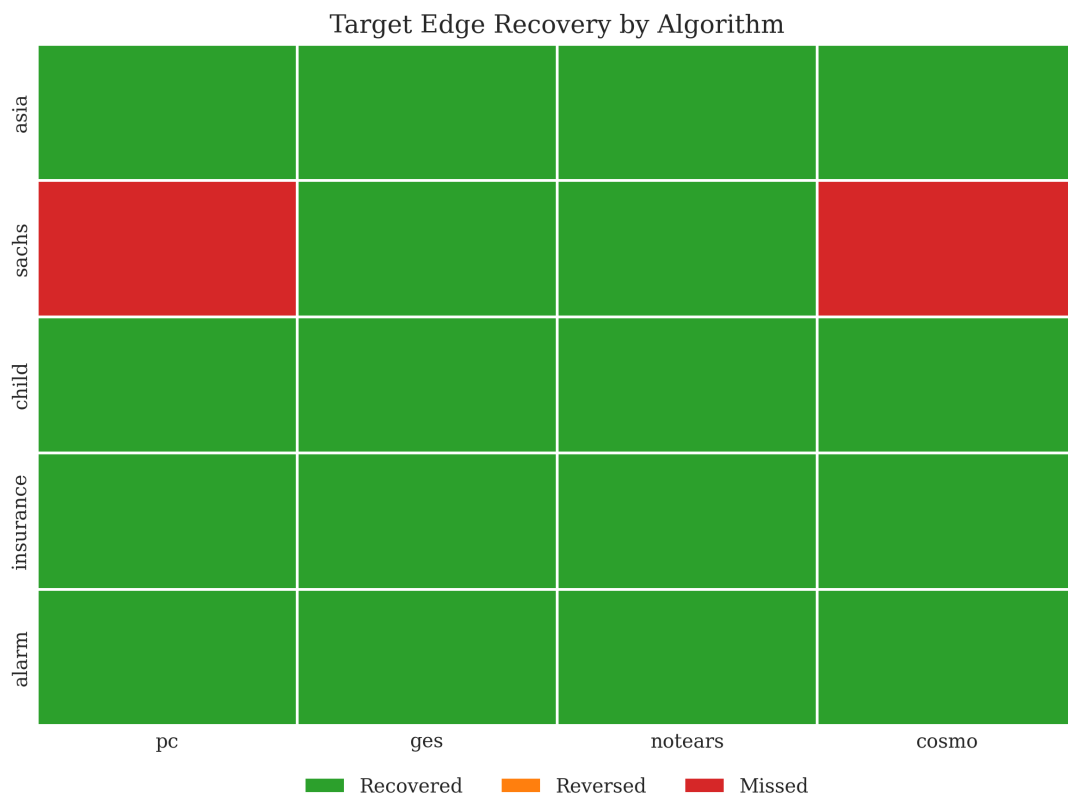


Figure 16: Recovery status of the specific omitted edge by each algorithm. Green indicates the edge was correctly recovered and oriented. Yellow indicates the edge was found but reversed. Red indicates the edge was missed entirely. Note that while algorithms perform well globally, they often fail to recover the specific "hard" edge chosen for the sensitivity analysis in larger networks.