Don't Test What You Can Deduce: Causal Discovery with Logical Inference

Jonghwan Kim Sanghack Lee
Graduate School of Data Science, Seoul National University, South Korea

uai2026

Seoul National University
Graduate School of Data Science

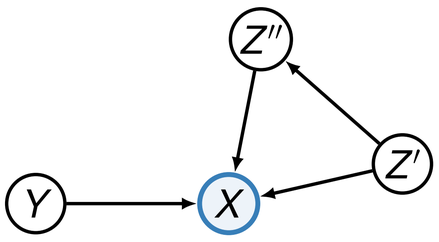
1. Background

- Constraint-based discovery removes edges using conditional independence tests (CITs):

$$\mathrm{CI}(X; Y \mid \mathbf{S}) \quad \Longrightarrow \quad X \perp\!\!\!\perp Y \mid \mathbf{S} \text{ or } X \not\perp\!\!\!\perp Y \mid \mathbf{S}.$$

- PC proceeds level-wise, increasing $|\mathbf{S}|$. With finite data, high-order CITs lose statistical power and false independencies can trigger fatal error propagation.
- Existing approaches either **restrict the number of CITs** or add **post-test checks**—leading to an under-specified graph or extra CITs.

2. A Motivating Example



Suppose we try to learn the graph on the left. At level 0, PC already finds

$$Y \perp\!\!\!\perp Z', \qquad Y \perp\!\!\!\perp Z''.$$

It also knows $Y \not\perp\!\!\!\perp X$ and $X \not\perp\!\!\!\perp Z'$.

Now PC moves to level 1 and asks for the outcome of $\mathrm{CI}(X; Y \mid Z')$. Using rules from graphoid axioms, this query can be deduced directly from the previous query history:

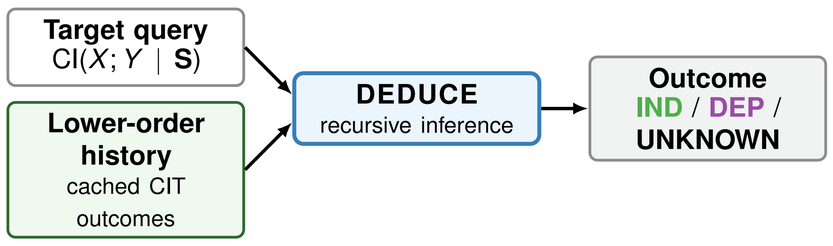
$$Y \perp\!\!\!\perp Z',\ Y \not\perp\!\!\!\perp X,\ X \not\perp\!\!\!\perp Z' \implies X \not\perp\!\!\!\perp Y \mid Z'.$$

No new CIT is needed.

Question: Why estimate a fragile high-order query if its answer is already logically entailed by lower-order outcomes in the PC algorithm's query history?

3. DEDUCE: A Logical Inference Engine

- We present DEDUCE, an engine for deducing higher-order CI outcomes from strictly lower-order CIT results.



- DEDUCE recursively applies the following sound rules derived from faithfulness and graphoid axioms.

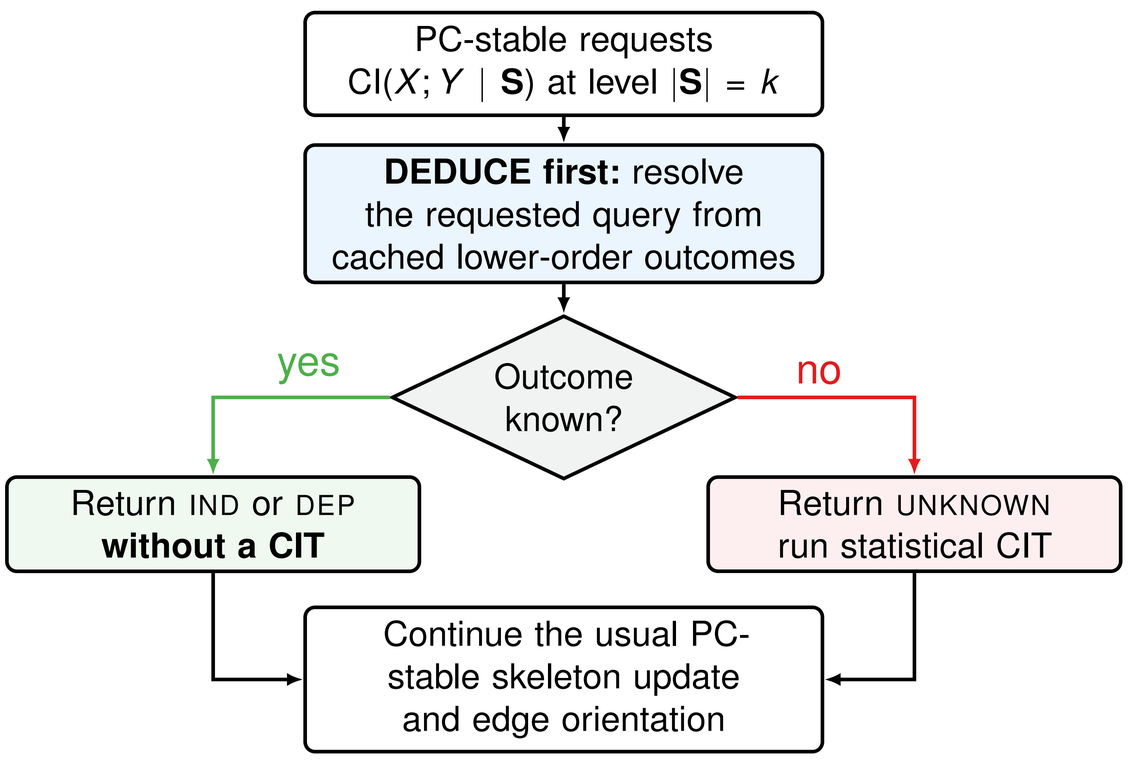
$$\Big[(X \not\perp\!\!\!\perp Y \mid \mathbf{S}') \oplus \big((X \not\perp\!\!\!\perp Z \mid \mathbf{S}') \wedge (Y \not\perp\!\!\!\perp Z \mid \mathbf{S}')\big)\Big] \implies X \not\perp\!\!\!\perp Y \mid \mathbf{S}' \cup \{Z\}$$

$$\Big[(X \perp\!\!\!\perp Y \mid \mathbf{S}') \wedge \big((X \perp\!\!\!\perp Z \mid \mathbf{S}') \vee (Y \perp\!\!\!\perp Z \mid \mathbf{S}')\big)\Big] \implies X \perp\!\!\!\perp Y \mid \mathbf{S}' \cup \{Z\}$$

$\oplus$ denotes exclusive OR.

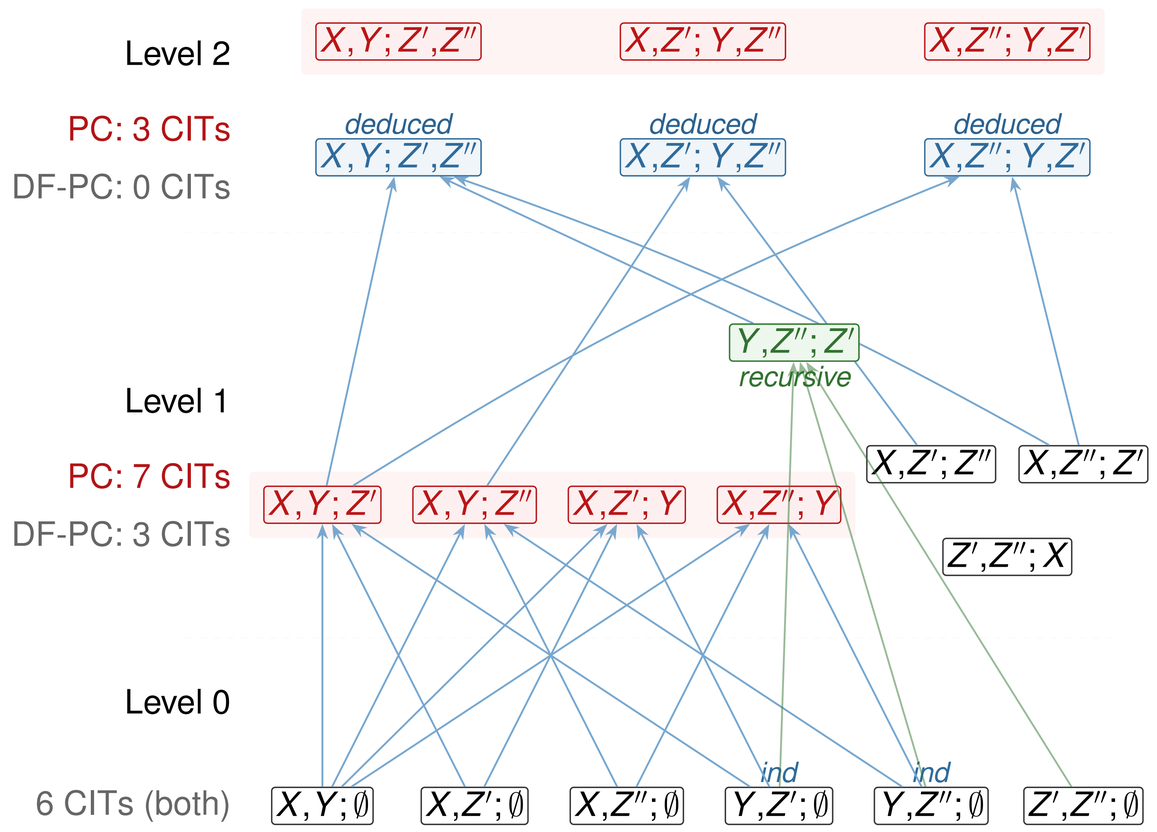
4. DF-PC: Deduce First, Test Only as Fallback

- We present DF-PC, a structure learning algorithm that integrates DEDUCE into PC-stable to skip unnecessary CITs.



- DF-PC keeps the PC-stable schedule unchanged and modifies only the query mechanism: DEDUCE(X, Y, $\mathbf{S}$) intercepts each requested CIT.

5. What Gets Skipped during Structure Learning?



Black: CITs used by both methods; **Red:** PC-only CITs;
Blue: DF-PC deductions; **Green:** recursive internal deduction.

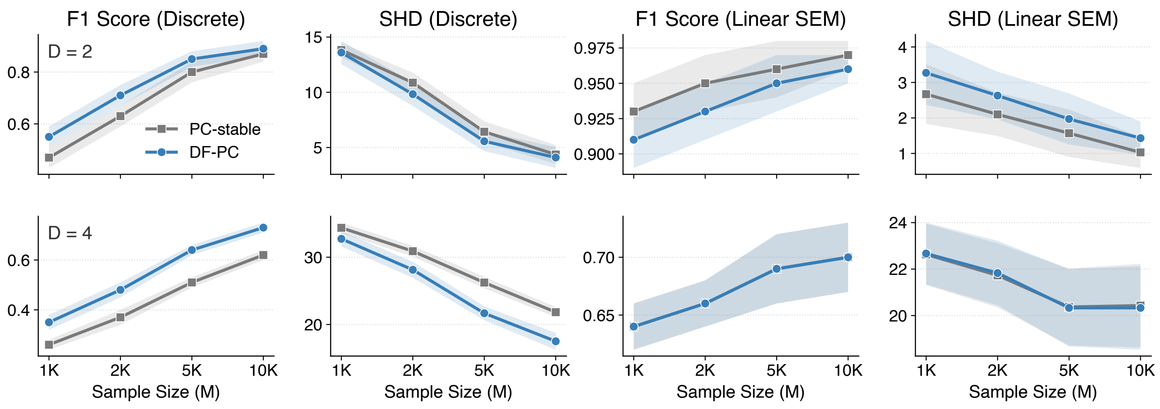
Total: PC-STABLE 16 CITs $\longrightarrow$ DF-PC 9 CITs.

6. Theoretical Discussions

- **Soundness of DEDUCE:** with a faithful Bayesian network and oracle tester, every deduced CI outcome is correct.
- **Soundness and completeness of DF-PC:** under an oracle tester, DF-PC recovers the same pattern as PC-stable.
- **Never more CITs than PC-stable:** each requested query is either deduced or sent to the same tester. For some nontrivial graphs, DF-PC can recover the exact structure using **only marginal CITs**—which PC-stable cannot do.

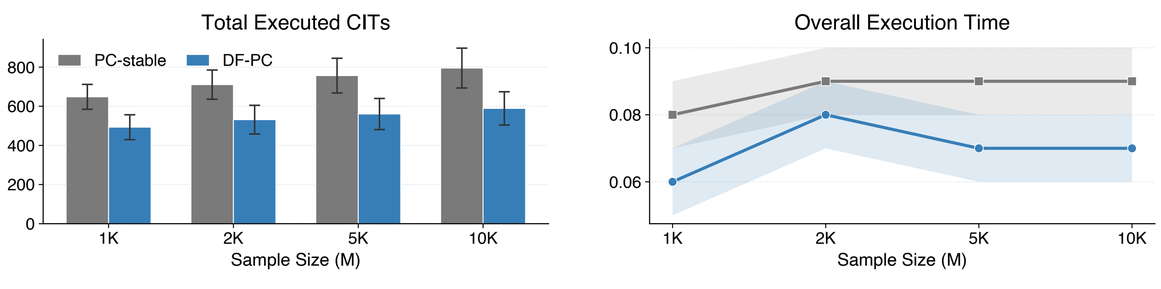
7. Empirical Results

Accuracy: DF-PC preserves or improves structure learning



Takeaway. DF-PC is consistently competitive and often better than PC-stable, with the clearest gains in **discrete data** and at **higher density**.

Efficiency: DF-PC executes fewer CITs and runs faster



Takeaway. By skipping many higher-order tests, DF-PC reduces the number of executed CITs and improves runtime.

8. Conclusion

- We present DF-PC, a variant of PC that not only uses independence findings as edge-deletion signals but also as ingredients of logical deduction.
- This approach effectively reduces higher-order CITs, lowering computational cost while preserving or improving learning accuracy.