Expert-In-The-Loop Causal Discovery: Iterative Model Refinement Using Expert Knowledge

Ankur Ankan[1] **Johannes Textor**[1]

[1]Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Abstract

Many researchers construct directed acyclic graph (DAG) models manually based on domain knowledge. Although numerous causal discovery algorithms were developed to automatically learn DAGs and other causal models from data, these remain challenging to use due to their tendency to produce results that contradict domain knowledge, among other issues. Here we propose a hybrid, iterative structure learning approach that combines domain knowledge with data-driven insights to assist researchers in constructing DAGs. Our method leverages conditional independence testing to iteratively identify variable pairs where an edge is either missing or superfluous. Based on this information, we can choose to add missing edges with appropriate orientation based on domain knowledge or remove unnecessary ones. We also give a method to rank these missing edges based on their impact on the overall model fit. In a simulation study, we find that this iterative approach to leverage domain knowledge already starts outperforming purely data-driven structure learning if the orientation of new edge is correctly determined in at least two out of three cases. We present a proof-of-concept implementation using a large language model as a domain expert and a graphical user interface designed to assist human experts with DAG construction.

1 INTRODUCTION

Understanding cause-and-effect relationships between variables is a fundamental objective in many scientific fields. These relationships reveal the mechanisms behind observed phenomena and guide effective interventions or policy decisions. Causal discovery methods aim to discover such relationships among random variables using observational data. These include constraint-based methods like the PC algorithm [Spirtes et al., 2001, Kalisch and Bühlmann, 2007] and Fast Causal Inference (FCI) [Spirtes et al., 2000], score-based methods such as Hill-Climb Search and Greedy Equivalence Search [Chickering, 2002], and continuous optimization-based methods like NOTEARS [Zheng et al., 2018] and DAGMA [Bello et al., 2022]. Despite this significant body of work, the adoption of causal discovery methods in observational research has so far been limited. Challenges encountered with existing causal discovery algorithms in practice include but are not limited to:

1. **Lack of Trust:** While constraint-based algorithms are often asymptotically consistent [Kalisch and Bühlmann, 2007], they can and do make mistakes on finite samples. These mistakes can be severe and contradict obvious domain knowledge (think of edges going into unmodifiable attributes such as Age). The choice of algorithm and hyperparameters significantly affects the output, making it difficult to assess reliability. Additionally, the absence of robust performance evaluation methods for any given dataset further reduce the confidence in their outputs. A recent paper advised Epidemiologists to not attempt using structure learning algorithms without the help of an expert [Gururaghavendran and Murray, 2024].
2. **Outputs Markov Equivalence Class (MEC):** As multiple DAGs can be consistent with an observational dataset, automated algorithms can only recover the MECs. These MECs can contain a combination of directed and undirected edges. This structural uncertainty can make it difficult or impossible to apply the learned model for downstream tasks, such as identification or causal effect estimation [Maathuis et al., 2009, Perkovic et al., 2017].

Figure 1 highlights some of these issues. In practice, DAGs are still largely constructed from domain knowledge alone [Tennant et al., 2020, Petersen et al., 2021]. This can, how-

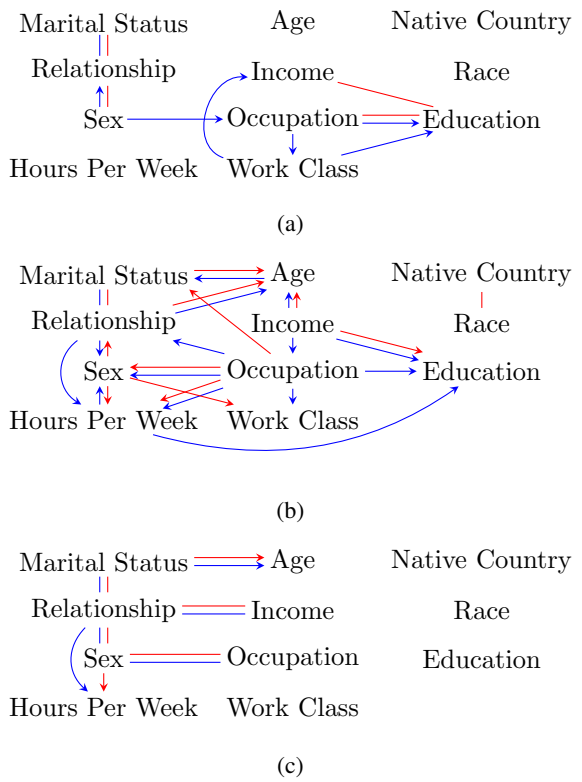


Figure 1: A comparison of Markov Equivalence Classes (MECs) learned from the Adult Income Dataset [Becker and Kohavi, 1996] using different causal discovery algorithms and sample sizes. Edge colors represent the sample size used: red for $N = 400$, and blue for $N = 800$. (a) PC algorithm with a mutual information based CI test, (b) PC algorithm with a residualization-based test [Ankan and Textor, 2023], (c) Hill Climb Search with Bayesian Information Criterion (BIC) score. The learned model structure varies significantly across different algorithms and sample sizes.

ever, be equally problematic. Constructing DAGs requires us to distinguish between different causal structures, such as direct or indirect effects or causal mediation. Specifically, there is often a lack of theoretical support for the *absence* of direct causal effects, which are the key assumptions that graphical models make to enable downstream causal inferences. Given the likelihood of making mistakes, it is therefore important to at least validate the consistency of a DAG against our dataset. One way to test this consistency is by testing whether the conditional independence (CI) statements implied by the DAG hold in the data [Ankan et al., 2021]. Specifically, each missing edge between a pair of variables in the DAG leads to one or more CI statements, which can be checked using statistical tests. Violations to CI statements can point to erroneous omission of important causal effects or to the presence of latent confounders.

In this paper, we propose a structure learning method that leverages CI testing to assist researchers during manual DAG construction, rather than merely validating the model. Specifically, our approach iteratively uses CI testing to identify pairs of variables that may lack (direct or indirect) connections and asks a domain expert to orient the potential causal relationship between them. This process can introduce superfluous edges, which are subsequently detected and removed using CI testing. Critically, our approach does not require the domain expert to distinguish between direct and indirect effects; it can be thought of as guidance during manual model construction, ensuring that the model remains consistent with the data.

Our contributions are organized as follows:

1. In Section 3.1, we present our iterative method for DAG construction based on domain knowledge and data, and prove its correctness in an "oracle setting". We show that the method remains valid even when allowing the domain expert to make certain kinds of mistakes.
2. We then present a ranking method to prioritize potential modifications to the DAG that help in fixing the most severe violations first (Section 3.2).
3. We show empirically that even experts that make mistakes can outperform purely data-driven causal discovery algorithms (Section 4).
4. We provide two proof-of-concept implementations of our approach: one geared towards LLMs as experts, and a graphical user interface designed for human experts (Section 5).

2 BACKGROUND

We denote random variables using uppercase letters like X, and a set of random variables by $\boldsymbol{X} = \{X_1, \cdots, X_k\}$, where $|\boldsymbol{X}| = k$ is the number of variables in the set. We denote the standard deviation of X as σ_X, the covariance between variables X and Y as $\mathrm{cov}(X, Y)$, a covariance matrix as Σ, and the entry corresponding to the covariance between X and Y as Σ_{XY}. A DAG $G = (V, E)$ is an acyclic directed graph whose nodes V correspond to random variables and whose edges E represent direct causal relationships [Pearl, 2009]. The set of parents of X in G is denoted as $\mathrm{Pa}_G(X)$, and its ancestors and descendants as $\mathrm{An}_G(X)$ and $\mathrm{De}_G(X)$, respectively; we use the convention that $X \notin \mathrm{An}_G(X)$ and $X \notin \mathrm{De}_G(X)$. We define the transitive closure G^+ of a DAG $G = (V, E)$ such that for any edge $X \to Y \in E$, G^+ has edges $\{X_i \to Y \ \forall X_i \in \mathrm{An}_G(X)\}$. We denote a path, $\pi(X, Y) = \{X, V_0, V_1, \cdots, V_k, Y\}$ between X and Y in G where consecutive pairs of variables in π are connected by an edge. We say X and Y are d-connected in G given a conditioning set $\boldsymbol{Z} \subseteq V - \{X, Y\}$, if there exists at least one d-connecting path $\pi(X, Y)$, i.e., a path $\pi(X, Y)$ for which (i) for every collider structure $(V_1 \to V_2 \leftarrow V_3)$ on $\pi(X, Y)$, $\boldsymbol{Z} \cap \{V_2, \mathrm{De}_G(V_2)\} \neq \emptyset$,

X	Y	$\boldsymbol{Z}$	p-value
Incm	MrtS	Age, Occp, Rltn	0.29
Incm	Sex	Occp, Rltn	0.21
Incm	Wrkc	Occp	0.00015
Edct	MrtS	Incm, Occp, Rltn	0.02

Figure 2: Results of testing some of the implied CIs of the DAG in Figure 1(b) using the same residualization based CI test that was used for learning it.

and (ii) for every non-collider structure ($V_1 \to V_2 \to V_3$, $V_1 \leftarrow V_2 \leftarrow V_3$, $V_1 \leftarrow V_2 \to V_3$), $V_2 \notin \boldsymbol{Z}$. X and Y are d-separated given $\boldsymbol{Z}$ if no d-connecting path exists between them. In particular, nodes connected by an edge cannot be d-separated by any $\boldsymbol{Z}$. The *skeleton* $S = (V, E_S)$ of a DAG $G = (V, E_G)$ is the undirected graph with edges $E_S = \{X - Y \mid X \to Y \in E_G \vee Y \to X \in E_G\}$.

2.1 ASSUMPTIONS

In this paper, we consider the structure learning problem under the widely known *causal Markov*, *causal sufficiency* and *faithfulness* assumptions [Spirtes et al., 1993]. These entail that there exists a DAG G on the variables $\boldsymbol{X}$ whose implied d-separation statements coincide exactly with the conditional independence relationships among the variables in $\boldsymbol{X}$, and for which all direct common causes of variables in $\boldsymbol{X}$ are also included in $\boldsymbol{X}$. These are the same assumptions made by the structure learning algorithms we consider in this paper, such as the PC algorithm [Spirtes et al., 1993].

2.2 CONDITIONAL INDEPENDENCE TESTS

Our structure learning algorithm uses the idea of testing DAGs using CI statements like $X \perp\!\!\!\perp Y | \boldsymbol{Z}$ (with potentially $\boldsymbol{Z} = \emptyset$). For example, take the MEC learned by the PC algorithm in Figure 1(b) using $N = 800$, and suppose that we orient the undirected edge between *Marital Status* and *Relationship* as *Marital Status* $\to$ *Relationship*. We can then read implied CIs of the DAG using d-separation, and test them in our dataset. Figure 2 shows the result of some tests. Using a significance threshold $\alpha = 0.05$, the first 2 implied CIs hold in the data whereas the remaining tests fail. One of the ways to fix these failing tests is to add an edge between the variable pair X and Y. While this method helps us in finding variable pairs where we may need to add an edge, it does not give us any information about the orientation of this edge. In this paper, we use domain knowledge to determine this orientation.

In addition to determining significance, we are interested in quantifying the *strength* of any partial association between X and Y after conditioning on $\mathbf{Z}$. This metric depends on the type of CI test used. In the following, we provide a brief overview of some CI tests and effect size measures for different types of (possibly mixed) data.

Both X and Y are continuous: When both X and Y are continuous, we can use a (partial) correlation test for CI testing and Pearson's correlation coefficient can be used as the effect size. When $\boldsymbol{Z} = \emptyset$, the correlation coefficient is defined as: $r_{X,Y} = \mathrm{cov}(X, Y)/(\sigma_X \sigma_Y)$. When $\boldsymbol{Z} \neq \emptyset$, the partial correlation coefficient can be used instead. This is estimated by fitting two regression models $E_X : X \sim \boldsymbol{Z}$ and $E_Y : Y \sim \boldsymbol{Z}$, calculating the residuals $R_X = X - E_X(\boldsymbol{Z})$ and $R_Y = Y - E_Y(\boldsymbol{Y})$, and computing the Pearson's correlation coefficient between the residuals: $r_{X,Y|\boldsymbol{Z}} = r_{R_X,R_Y}$

X is ordinal, and Y and $\boldsymbol{Z}$ are continuous or ordinal: Polyserial (for continuous Y) and polychoric (ordinal Y) correlations are used to estimate correlations involving ordinal variables [Poon and Lee, 1987]. Both methods make the assumption that the observed ordinal variable is a result of thresholding a latent normally distributed continuous variable. Under this assumption, the methods then estimate the threshold values and covariance matrix using maximum likelihood. Using the estimated covariance matrix, Σ we can perform a correlation test for CI and compute the Pearson's correlation coefficient as the effect size (same as for continuous X and Y), i.e., $\boldsymbol{Z} = \emptyset$, $r_{X,Y} = \Sigma_{XY}/(\sqrt{\Sigma_{XX}\Sigma_{YY}})$, and when $\boldsymbol{Z} \neq \emptyset$, $r_{X,Y|\boldsymbol{Z}} = -\Sigma^{-1}_{XY}/(\sqrt{\Sigma^{-1}_{XX}\Sigma^{-1}_{YY}})$.

X, Y, and $\boldsymbol{Z}$ are all discrete (ordinal or categorical): For combinations of ordinal and categorical variables, we can use a residualization-based CI test [Ankan and Textor, 2023] that returns a chi-square distributed test statistic. Given the statistic, χ^2, with df degrees of freedom, we use the Root Mean Squared Error of Approximation (RMSEA) defined as $\mathrm{RMSEA}_{X,Y|\boldsymbol{Z}} = \sqrt{\max(0, \chi^2 - \mathrm{df})/(\mathrm{df}(N-1))}$, where N is the sample size. This effect size can be used for any statistical test with a chi-square distributed test statistic.

3 EXPERT-IN-THE-LOOP CAUSAL DISCOVERY

Our approach to causal structure learning combines domain knowledge with data-driven insights in a manner that is based on the following considerations: (1) A domain expert is possibly good at determining causal directions between variables if there *is* a clear causal direction between them. (2) A domain expert may have difficulty at identifying cases where there is no causal relationship between the variables, since *potential* causal relationships can often be argued for anyway. (3) Many domain experts could struggle to distinguish direct from indirect effects, since the presence of a direct effect between two variables of interest depends on all other variables present in the graph.

We will first present theoretical results showing how a structure learning algorithm using such experts can uncover the true DAG structure. Afterwards, we will present a heuristic for deciding which changes should be prioritized.

3.1 DAG STRUCTURE LEARNING USING ANCESTRAL ORACLES

We model domain experts as procedures that take two variables X and Y that are assumed to be part of a DAG G and provide information on the ancestral relationship between them. First, a *strong ancestral oracle* $\mathcal{A}_G$ is defined as:

$$\mathcal{A}_G(X,Y) = \begin{cases} X \to Y & \text{if } X \in \text{An}_G(Y) \\ X \leftarrow Y & \text{if } Y \in \text{An}_G(X) \\ \text{None} & \text{otherwise} \end{cases}$$

Note that the ancestral oracle does not consider differences between direct and indirect relationships: for any G, H where $G^+ = H^+$, we have $\mathcal{A}_G = \mathcal{A}_H$.

Experts can make mistakes. In our analysis, we will consider experts that essentially "make up" non-existing causal relationships, but do provide correct answers on the directionality of existing ones.

Definition 1. *Let $G = (V, E_G)$ and $H = (V, E_H)$ be two DAGs. The G-*compatible ancestral oracle *$\mathcal{A}_{G|H}$ is defined by*

$$\mathcal{A}_{G|H}(X,Y) = \begin{cases} X \to Y & \textit{if } X \in An_G(Y) \\ X \leftarrow Y & \textit{if } Y \in An_G(X) \\ \mathcal{A}_H(X,Y) & \textit{otherwise} \end{cases}$$

Consider the graph $G|H$ containing the edges $\mathcal{A}_{G|H}(X,Y)$ for all pairs $(X,Y) \in V \times V$. If $G|H$ is acyclic, then $\mathcal{A}_{G|H}$ is called an acyclic G-compatible ancestral oracle.

Note that a G-compatible oracle can contradict itself. Consider $V = (X, Y, Z)$, $G = (V, \{X \to Y\})$ and $H = (V, \{Y \to Z, Z \to X\})$. This gives $\mathcal{A}_{G|H}(Y,Z) = Y \to Z$ and $\mathcal{A}_{G|H}(X,Z) = Z \to X$ which imply that Y causes X, in contradiction to $\mathcal{A}_{G|H}(X,Y) = X \to Y$. Still, it will turn out that we can recover from such errors.

Generally, using ancestral oracles for DAG construction introduces superfluous edges. Our approach uses the data to decide where to add edges and which edges are superfluous and can be removed. As a useful abstraction, let us assume that we have access to a second oracle $\mathcal{D}_G$ that can answer d-separation queries with respect to the unknown true graph G: $\mathcal{D}_G(X,Y,\mathbf{Z}) = 1$ iff X and Y are d-separated by $\mathbf{Z}$ in G, and 0 otherwise. These are the standard oracles considered in constraint-based structure learning algorithms, such as PC [Spirtes et al., 2001].

We now define two core procedures that use ancestral and d-separation queries to iteratively change a current DAG structure. The procedure EXPAND (Algorithm 1) uses CI information to search for unexplained associations in the graph and then uses domain knowledge to determine ancestry. The procedure PRUNE (Algorithm 3) uses CI information to remove superfluous edges from the graph.

Algorithm 1: Adding edges based on data and domain knowledge.

1 **Function** EXPAND($V, E, \mathcal{D}, \mathcal{A}, B, k$):
2 $\quad L \leftarrow \{\}$
3 $\quad$ **foreach** X, Y *where* $X \to Y \notin E \cup B$ *and* $Y \to X \notin E \cup B$ **do**
4 $\quad\quad$ $\mathbf{Z}$ be a set that d-separates X and Y in (V, E)
5 $\quad\quad$ **if** $\mathcal{D}(X, Y, \mathbf{Z}) = 0$ **then**
6 $\quad\quad\quad$ $L \leftarrow L \cup \mathcal{A}(X, Y)$
7 $\quad\quad$ **end**
8 $\quad\quad$ **if** $|L| \geq k$ **then**
9 $\quad\quad\quad$ **go to** 12
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 $\quad R \leftarrow$ FIXCYCLES$(V, E \cup L, \mathcal{D})$
13 $\quad B \leftarrow B \cup R$; $E \leftarrow (E \cup L) \setminus R$
14 $\quad$ **return** (V, E, B)

Algorithm 2: Fixing cycles by removing incorrect edges.

1 **Function** FIXCYCLES($V, E, \mathcal{D}$):
2 $\quad R \leftarrow \emptyset$
3 $\quad$ **foreach** $X \to Y$ *on a cycle in* (V, E) **do**
4 $\quad\quad$ **if** *there exists a set* $\mathbf{Z} \subseteq \mathbf{X}$ *where* $\mathcal{D}(X, Y, \mathbf{Z}) = 1$ **then**
5 $\quad\quad\quad$ $R \leftarrow R \cup \{X \to Y\} \cup \{Y \to X\}$
6 $\quad\quad$ **end**
7 $\quad$ **end**
8 $\quad$ **return** R

EXPAND takes an initial list of edges and searches for any unconnected vertex pairs that are not connected but where the d-separation oracle indicates a residual association not explained by other paths. The parameter B specifies a "black list" of edges that must not be added. This is important to prevent edges that were removed from cycles to be added again and will have another important role in the overall algorithm. In addition, k can be used to limit the maximum amount of edges to be added by this procedure; it will become clear soon why this is useful.

The following two propositions characterize the results of EXPAND.

Proposition 1. *For a conditional independence oracle $\mathcal{D}_G$ and a strong ancestral oracle $\mathcal{A}_G$,* EXPAND$(V, \emptyset, \mathcal{D}_G, \mathcal{A}_G, \emptyset, \infty) = G^+$.

Proof. Since we start from an empty graph, every pair of vertices X, Y where $X \in \mathrm{An}_G(Y)$ is d-separated by the empty set and is connected by the edge $X \to Y$. No other edges are added. Therefore, the result is G^+. □

When using experts that do not always correctly detect the absence of causal relationships, the resulting graph can get larger but also smaller, if this leads to the occurrence of cycles that need to be broken.

Proposition 2. *For a conditional independence oracle $\mathcal{D}_G$ and a G-compatible ancestral oracle $\mathcal{A}_{G|H}$, let $\tilde{G} = \textsc{Expand}(V, \emptyset, \mathcal{D}_G, \mathcal{A}_{G|H}, \emptyset, \infty)$ Then $\tilde{G}$ is acyclic, and $G \subseteq \tilde{G}$.*

Proof. Let S be the skeleton of G. All edges in S are added during EXPAND with correct orientation and cannot be removed by FIXCYCLES. Therefore, the result is a supergraph of G. Each cycle that is possibly created during EXPAND contains at least one edge that is not in S, otherwise G itself would be cyclic. At least one edge of every cycle is therefore removed by FIXCYCLES, making the result acyclic again. The removed edge cannot be in S, so $G \subseteq \tilde{G}$. □

Algorithm 3: Pruning superfluous edges

1 **Function** PRUNE*(V,E,$\mathcal{D}$)*:
2 $R \leftarrow \{\}$
3 **foreach** $X \to Y \in E$ **do**
4 let $\mathbf{Z}$ be a set that d-separates X and Y in $(V, E \setminus \{X \to Y\})$
5 **if** $\mathcal{D}(X, Y, \mathbf{Z}) = 1$ **then**
6 $R \leftarrow R \cup \{X \to Y\}$
7 **end**
8 **end**
9 $E \leftarrow E \setminus R$
10 **return** (V, E)

Unlike our ancestral oracles, the pruning operation is quite effective at distinguishing direct and indirect effects, as shown by the following result.

Proposition 3. *Consider two DAGs $G = (V, E)$ and $G' = (V, E')$ where $E \subseteq E'$. Then $\textsc{Prune}(V, E', \mathcal{D}_G) = (V, E)$.*

Proof. Let S be the skeleton of G. For an edge $X - Y$ in S, $\mathcal{D}_G(X, Y, \mathbf{Z}) = 0$ regardless of $\mathbf{Z}$, so all edges in the skeleton are retained after PRUNE. Conversely, if $X - Y$ is not in S, let $\mathbf{Z}$ be the d-separating set chosen in line 4 of PRUNE (there is at least one such $\mathbf{Z}$, the union of the parents of X and Y). Since $\mathbf{Z}$ d-separates all paths from X to Y in G', it does the same in G which contains a subset of these paths. Therefore, all edges not in G are removed. □

By combining Propositions 1, 2 and 3, we immediately obtain the following:

Theorem 1. *Given a d-separation oracle $\mathcal{D}_G$ and a G-compatible ancestral oracle $\mathcal{A}_{G|H}$, let $(V', E', B) = \textsc{Expand}(V, \emptyset, \mathcal{D}_G, \mathcal{A}_{G|H}, \emptyset, \infty)$. Then $\textsc{Prune}(V', E', \mathcal{D}_G) = G$.*

Since we require expert knowledge only in the EXPAND operation, we may try to be more economical by asking fewer questions at a time and interleaving expansion and pruning steps. This leads us to the following, more iterative DAG construction algorithm.

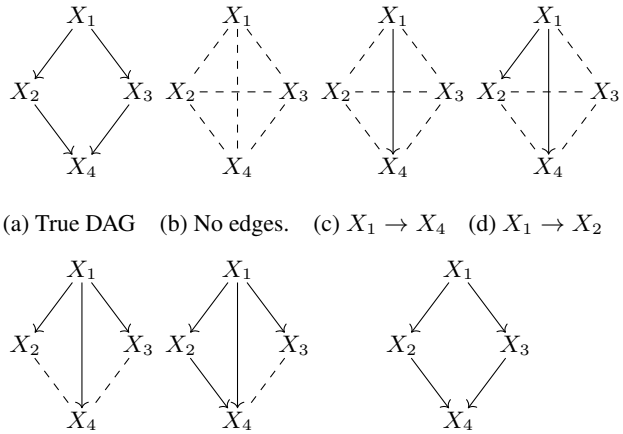
Algorithm 4: Iterative structure learning with expert in the loop

1 **Function** EXPERTINLOOP*($V, \mathcal{D}, \mathcal{A}$)*:
2 $E_p \leftarrow \emptyset$ /* Current edges */
3 $B \leftarrow \emptyset$ /* Edges that were pruned or removed from cycle */
4 **repeat**
5 $E \leftarrow E_p$
6 $(V, E, B) \leftarrow \textsc{Expand}(V, E, \mathcal{D}, \mathcal{A}, B, 1)$
7 $(V, E_p) \leftarrow \textsc{Prune}(V, E, \mathcal{D})$
8 $B \leftarrow B \cup \{E \setminus E_p\}$
9 **until** $E = E_p$
10 **return** *(V,E)*

Theorem 2. *Let $G = (V, E^*)$ be a DAG, $\mathcal{D}_G$ a d-separation oracle for G, and $\mathcal{A}_{G|H}$ a G-compatible ancestral oracle for G. Then $\textsc{ExpertInLoop}(V, \mathcal{D}_G, \mathcal{A}_{G|H}) = G$.*

Proof. The loop in Algorithm 4 terminates if and only if $E = E^*$. If the loop does not terminate, a new edge has been added in line 6, and/or one or more edges were pruned in line 7. Every edge can be added at most once and pruned at most once. Therefore, the algorithm always terminates after at most $|V|(|V| - 1) + 1$ iterations of the loop. For every edge $e = X \to Y$ in G, $\mathcal{D}_G(X, Y, \mathbf{Z}) = 0$ irrespective of $\mathbf{Z}$, so e must be added to E in some iteration, and can never be pruned afterwards. Therefore, after some iteration, (V, E) must be a supergraph of G after executing line 6, and this will be pruned to the real graph G in line 7 (Proposition 3). In the next iteration, no further changes are made, and the loop terminates. □

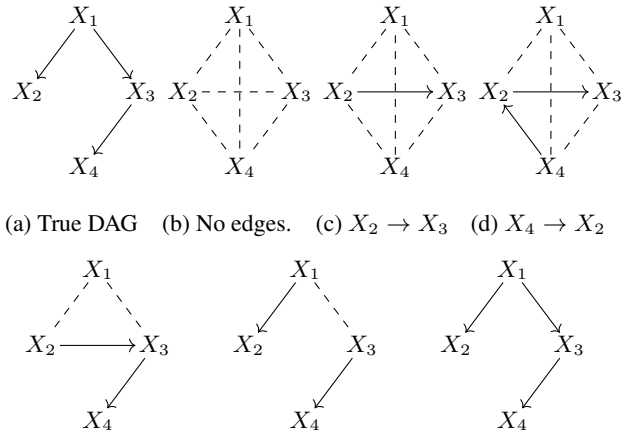
Figures 3 and 4 show possible runs of EXPERTINLOOP. We note that the general property of Algorithm 4 that each possible edge can be added and removed at most once, which guarantees that the algorithm eventually terminates, remains valid even if the oracles make arbitrary mistakes. This is crucial when using the algorithm in practice.



(a) True DAG (b) No edges. (c) $X_1 \to X_4$ (d) $X_1 \to X_2$

(e) $X_1 \to X_3$ (f) $X_2 \to X_4$ (g) $X_3 \to X_4$, $X_1 \not\to X_4$

Figure 3: Results of multiple iterations of EXPERTINLOOP with $k = 1$ using a strong ancestral oracle. Dashed edges are candidates for addition in each iteration of EXPAND.



(a) True DAG (b) No edges. (c) $X_2 \to X_3$ (d) $X_4 \to X_2$

(e) $X_3 \to X_4$, $X_4 \not\to X_2$ (f) $X_1 \to X_2$, $X_2 \not\to X_3$ (g) $X_1 \to X_3$

Figure 4: Similar example as in Figure 3 but using a G-compatible ancestral oracle that adds the edges $X_2 \to X_3$ and $X_4 \to X_2$, leading to a cycle that is broken in (e).

3.2 RANKING POTENTIAL NEW EDGES

EXPERTINLOOP adds a new edge in each iteration. The edge to be added is selected using the EXPAND algorithm, which returns a potential edge at random based on the iteration order. However, if the residual association between the selected variables is low, adding the edge may result in only a marginal improvement in the overall model fit. Given that the algorithm requires expert intervention in each iteration to specify the edge orientation, it can be beneficial to prioritize edges that contribute the most to improving the model.

To achieve this, instead of selecting edges randomly, we propose ranking potential edges based on their residual association given the current DAG. This residual association can be quantified using the effect sizes from the CI tests used for deciding d-separation in Algorithm 1. Specifically, for a d-separation query $D_G(X, Y, \mathbf{Z})$, we quantify the residual association, $\phi(X, Y, \boldsymbol{Z})$, as the effect size of the CI test $X \perp\!\!\!\perp Y | \boldsymbol{Z}$.

Algorithm 5 shows a version of EXPAND where we use the largest unexplained association to determine where to next add an edge. To implement this, we need an effect size metric by which we can rank effects. Depending on the type of variables, we can use the effect sizes shown in Section 2.2. The algorithm selects the edge that has the highest absolute residual association, resulting in prioritization of edges that contribute the most to improving the model.

In addition to prioritizing edges to add or remove, the ranking could also be used to break cycles when CI tests fail to identify the edges that are not in the skeleton during the procedure FIXCYCLE, which could happen due to finite sample effects. In that case, removing edges that correspond to weak associations may be preferable to keeping the cycles in the model.

3.3 COMPARISON TO SCORE-BASED METHODS

Superficially, the data-driven aspects of our procedure appear similar to score-based automated causal discovery methods like Greedy Equivalence Search (GES), which iteratively add or remove edges that maximize improvements in a specified scoring metric. Indeed, we can define the *total residual association* τ for a DAG $G = (V, E)$ as:

$$\tau = \sum_{\substack{X,Y \in V \\ X \to Y, Y \to X \notin E}} \phi(X, Y, \mathrm{pa}_G(X) \cup \mathrm{pa}_G(Y)) \quad (1)$$

Our RANKEDEXPAND approach behaves similarly to GES in that it tries to prioritize modifications that lead to the largest improvements in τ. However, adding an edge between the two vertices that contribute the most to τ does not necessarily lead to the largest possible decrease in τ because the new edge can also affect other residual associations. In other words, τ is not a *decomposable* fit measure like the ones normally used in score-based structure learning.

Another key difference lies in the interpretability of the evaluation metric. Scoring metrics are usually based on the log-likelihood with a penalty for model complexity. They can be used to make relative comparisons between models but their values do not have any interpretation in an absolute sense. That is, they indicate which model is better for a given dataset but do not quantify how well the model explains the data. In contrast, τ could be seen as an absolute measure

Algorithm 5: Adding an edge between variables with the highest correlation

Function RANKEDEXPAND($V, E, \mathcal{D}, \mathcal{A}, \phi, B$):
  $\phi_{\max} \leftarrow 0$; $L \leftarrow \emptyset$
  foreach X, Y *where* $X \to Y \notin E \cup B$ *and* $Y \to X \notin E \cup B$ **do**
    let $\mathbf{Z}$ be a set that d-separates X and Y in (V, E)
    if $\mathcal{D}(X, Y, \mathbf{Z}) = 0$ **then**
      if $|\phi(X, Y, \mathbf{Z})| > \phi_{\max}$ **then**
        $\phi_{\max} \leftarrow |\phi(X, Y, \mathbf{Z})|$
        $L \leftarrow \{\mathcal{A}(X, Y)\}$
      end
    end
  end
  $R \leftarrow$ FIXCYCLES$(V, E \cup L, \mathcal{D})$
  $B \leftarrow B \cup R$; $E \leftarrow (E \cup L) \setminus R$
  return (V, E, B)

of model fit – its value approaches 0 as the model perfectly explains the observed data.

It is possible to integrate domain knowledge into score-based methods in a similar way as we do here for constraint-based structure learning. Kitson and Constantinou [2025] recently extended the Tabu search algorithm by a procedure that asks a domain expert for advice before making changes that would improve the score only marginally. In contrast, our approach places greater emphasis on expert input as no edges are oriented without consulting the expert. Further, the approach by Kitson and Constantinou [2025] requires experts that are able to distinguish between direct and indirect effects.

4 EMPIRICAL ANALYSIS

In this section, we compare our EXPERTINLOOP algorithm with automated causal discovery algorithms. Our goal with the empirical analysis is to understand the behaviour of our method when both experts and d-separation oracles are not perfect. Specifically, how good does the domain expert have to be so we actually benefit from their expertise, compared to a purely data-driven approach?

In our analysis, we simulate data from a "true" DAG G and use the EXPERTINLOOP algorithm with the RANKEDEXPAND heuristic to recover G. We implement the d-separation oracle $\mathcal{D}_G(X, Y, \mathbf{Z})$ by conducting conditional independence tests of $X \perp\!\!\!\perp Y \mid \mathbf{Z}$, which on finite data inevitably make type I and type II errors. To simulate an imperfect domain expert, we use a version of a strong ancestral oracle $\mathcal{A}_G$ (Section 3.1) that with probability α knows the correct

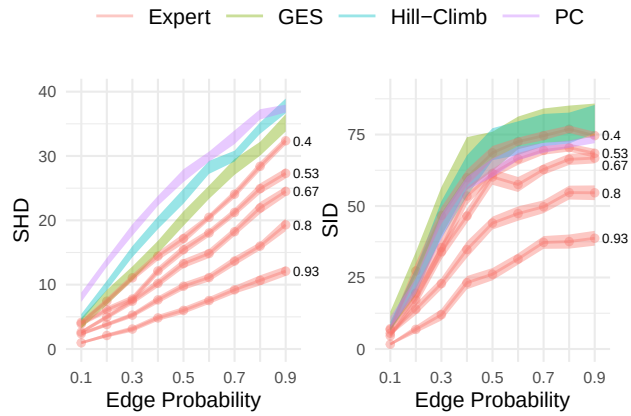


Figure 5: Comparison of PC, Hill-Climb Search, and GES algorithms against EXPERTINLOOP algorithm. As automated algorithms only recover the CPDAG, we use the best and worst scoring orientation of the CPDAG to get the range. We test EXPERTINLOOP with varying values of expert accuracy, $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The corresponding α_{eff} is shown in the plot. Expert shading: mean $\pm$ standard error. Others shading: [mean min. - standard error min., mean max. + standard error max.]

answer and randomly guesses otherwise, i.e.,

$$x = \text{rand}([0, 1])$$
$$\text{Expert}(\alpha) = \begin{cases} \mathcal{A}_G(X, Y), & \text{if } x <= \alpha \\ \text{rand}(X \to Y, Y \leftarrow X, \text{None}) & \text{otherwise} \end{cases}$$

Additionally, instead of performing an exhaustive search for separating sets in FIXCYCLES, we used a heuristic: for each edge $X \to Y$ on the cycle, we ran a CI test between X and Y, using as the conditioning set the parents of Y combined with one variable from the cycle. This approach was able to successfully break all cycles in our empirical analysis.

It is important to note that the effective accuracy of Expert is higher than α as even when $x > \alpha$, there is a $1/3$ chance that edge orientation is correct. Therefore, an Expert with accuracy α has an effective accuracy, $\alpha_{\text{eff}} = \alpha + (1 - \alpha)/3$.

We compare the performance of the EXPERTINLOOP algorithm to three other automated algorithms: PC [Spirtes et al., 2001, Kalisch and Bühlmann, 2007], Hill-Climb Search [Scutari, 2010], and Greedy Equivalence Search (GES) [Chickering, 2002] on linear-Gaussian data. For EXPERTINLOOP, we use the RANKEDEXPAND method with Pearson's correlation coefficient as the effect size (Section 2.2) and significance threshold (and type I error rate) $\alpha = 0.05$. We start by generating a random DAG on 10 nodes and use linear models with random effects to simulate 500 samples

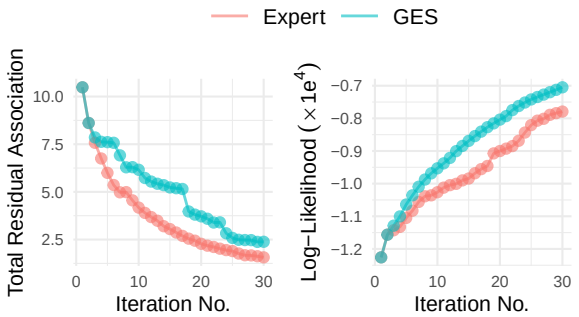


Figure 6: LLM-in-the-loop causal discovery on the Adult Income datases. As the ground truth is unknown, two measures of fit are shown over 30 iterative modifications: total residual association τ (left, see Equation 1; lower is bettter); log-likelihood (right, higher is better).

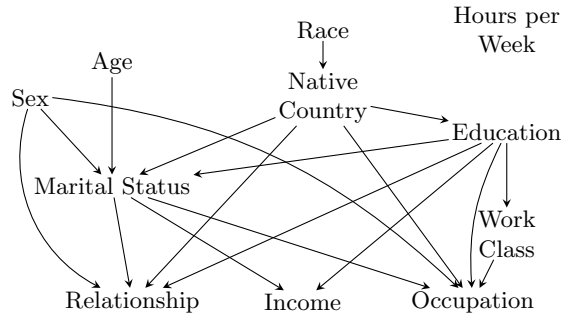


Figure 7: DAG learned from the Adult Income dataset using an LLM (Gemini 1.5 Flash) as the expert. The p-value threshold used is 0.05 and the measure of association threshold is 0.1, meaning that associations below 0.1 are not considered by RANKEDEXPAND.

from it. We simulate each variable V_i in the DAG as

$$V_i = \boldsymbol{\beta} \cdot \mathrm{Pa}_G(V_i) + \mathcal{N}(0, 1)$$

where each coefficient β_i of the vector $\boldsymbol{\beta} \in \mathbb{R}^{|\mathrm{Pa}_G(V_i)|}$ is randomly drawn as

$$\beta_i \sim \mathrm{Uniform}([-0.6, 0.6])$$

We perform the experiment for varying densities of the DAG and accuracies of Expert. We repeat each experiment 30 times and report the mean and standard error of the results.

To compare the learned DAG to the original DAG, we use two metrics: the Structural Hamming Distance (SHD) and the Structural Intervention Distance (SID) [Peters and Bühlmann, 2015]. Unlike EXPERTINLOOP, the automated algorithms can only recover the MEC, therefore, we considered all possible orientations of the MEC to compute a range of SHD and SID values representing the best and worst case scenarios. The results of this analysis are shown in Figure 5. For SHD, the performance of EXPERTINLOOP is comparable to the automated algorithms for $\alpha = 0.3$ ($\alpha_{\text{eff}} = 0.53$), implying that if an expert is able to get the edge orientation correctly in more than 1 out of 2 cases, they can outperform the automated algorithms. Similarly for SID, an expert can outperform the automated algorithms if they are able to get the orientation correct in 2 out of 3 cases. Another thing to note is that for denser DAGs, the EXPERTINLOOP is able to perform better for lower α values.

5 PRACTICAL IMPLEMENTATION

We now present two proof-of-concept implementations of our approach, geared towards using large language models (LLMs) and humans as experts to decide on edge directions.

We use the Adult Income dataset from the introduction as an example. There is no known ground truth for this dataset, and it contains mixed types of data; here, we use a version that contains ordinal and nominal variables and use the RM-SEA of a residualization-based conditional independence test [Ankan and Textor, 2023] as a measure of association to prioritize modifications (Section 2).

5.1 USING LLMS AS EXPERTS

Recently, there has been significant interest in leveraging Large Language Models (LLMs) for causal discovery. These applications range from determining pairwise edge orientations [Kıcıman et al., 2023, Jin et al., 2024] to full causal structure learning [Naik et al., 2023, Vashishtha et al., 2023] and counterfactual reasoning [Kıcıman et al., 2023] (see Liu et al. [2024] for a comprehensive overview).

Since our approach relies on expert knowledge to determine ancestral relationships, we explored the potential of using an LLM for this task. We applied our causal discovery procedure using the Gemini 1.5 Flash model as the expert. We simulate the *ancestral oracle* using the LLM by asking it to choose the causal direction between a pair of variables. The LLM is provided a description of each of the variables and given the prompt shown in Appendix A. Figure 6 shows how the model fit to the data improves across 30 iterations compared to GES for a run of the algorithm. Figure 7 shows the DAG learned on the Adult Income dataset using the LLM as the expert. We can see that the model fit improves comparably to a greedy approach and the resulting DAG appears to contain largely sensible edge directions. We have provided an implementation of this method in the pgmpy package [Ankan and Textor, 2024].

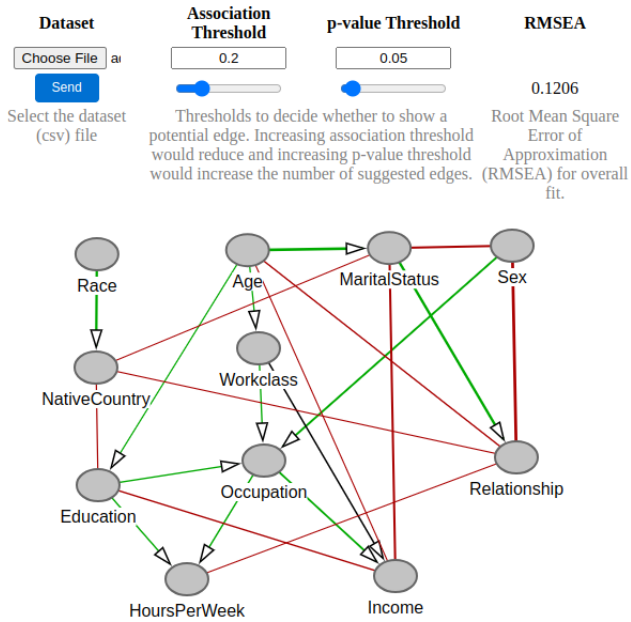


Figure 8: Tool for guided model construction. Upon dataset upload, the tool creates an empty graph and shows all pairs of variables with unexplained residual associations using undirected red edges (edge width is proportional to association strength). Users iteratively add suggested edges (shown in green), relying on their knowledge when specifying the orientation. Superfluous edges are shown in black.

5.2 USING HUMANS AS EXPERTS

To enable researchers to easily apply our approach to their own datasets, we developed an interactive web tool (Figure 8) for constructing DAGs. Users can upload their dataset, which initializes an empty DAG with nodes corresponding to the dataset's variables. They can then specify a p-value threshold and a threshold for a minimal association strength. The tool then visually highlights variable pairs with a residual association greater than the threshold, marking them with red edges; to avoid cluttering, no lines are drawn for residual associations that are statistically insignificant or below the specified threshold. The thickness of these edges represents the strength of association, helping users prioritize which variable pairs to address first; however, note that users are free to choose which variables to connect. Similarly, if an existing edge is found to be potentially superfluous (statistically insignificant), it is highlighted in black. Using this information, users can iteratively modify the model by adding or removing edges. The tool also computes the Root Mean Square Error of Approximation (RMSEA) based on Shipley's C test, a global test of model fit based on the implied CIs [Shipley, 2000], to provide an estimate of the overall fit of the model. Once satisfied with the constructed DAG, users can export the model for further analysis. This web tool can be accessed at: `https://ankurankan.github.io/2025-causal-discovery-webapp/`

6 CONCLUSIONS

Researchers often prefer to construct DAGs manually rather than using automated algorithms, which could be partly due to practical challenges with automated methods. To assist this process, we developed an iterative structure learning method that integrates manual construction with data-driven feedback, bridging the gap between fully manual and fully automated methods. The idea of augmenting structure learning by domain knowledge is of course not new. Common ways to provide such background knowledge are "whitelists" or "blacklists" of edges [Scutari, 2010], or specification of a "tiered" (e.g., temporal) structure between the variables [Bang and Didelez, 2023]. Compared to such approaches, where domain knowledge is specified and supplied beforehand, the novelty of our method lies in the interactive back-and-forth between domain knowledge and data, which generates the correct result in a polynomial amount of steps in the oracle settings we considered.

Nevertheless, the algorithm presented in this paper has significant limitations. First of all, the causal Markov and causal sufficiency assumptions are widely seen as too restrictive, since they rule out latent confounders. Further, recovering from mistakes that introduce cycles requires searching for a separating set for each edge on each cycle, which in the worst case can take exponential time and would then essentially amount to running significant parts of the PC algorithm. In practice, we may be better off breaking cycles by other means – such as the heuristic implemented in our experiments – even if the theoretical correctness guarantee is lost. Lastly, any iterative model improvement procedure runs the risk of overfitting to the given dataset. While this is also the case for purely data-driven structure learning (which may even require a vastly larger amount of model improvement steps), the inclusion of human judgement in this process means that it will be seen as less reliable. Models generated by this approach should certainly be seen as preliminary and in need of further validation using independent data, and a log of all human decisions made during the process should be kept. In this respect, it will be interesting to see how the research community will feel about the use of LLMs instead of humans for model refinement.

Avenues for future work include extending this approach to less restrictive assumptions, such as those used by FCI [Spirtes et al., 2000] that allow for latent confounding. Further, if experts make mistakes that lead to inconsistencies during model construction (such as cycles), this information could be leveraged in a more systematic way to allow backtracking and recovery.

Acknowledgements

We acknowledge the use of ChatGPT-4o (`https://chatgpt.com/`) for proofreading parts the manuscript.

References

Ankur Ankan and Johannes Textor. A simple unified approach to testing high-dimensional conditional independences for categorical and ordinal data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37 (10):12180–12188, June 2023. ISSN 2159-5399. doi: 10.1609/aaai.v37i10.26436. URL http://dx.doi.org/10.1609/aaai.v37i10.26436.

Ankur Ankan and Johannes Textor. pgmpy: A Python Toolkit for Bayesian Networks. *Journal Of Machine Learning Research*, (25):1–8, 2024.

Ankur Ankan, Inge M. N. Wortel, and Johannes Textor. Testing graphical causal models using the r package "dagitty". *Current Protocols*, 1(2), February 2021. ISSN 2691-1299. doi: 10.1002/cpz1.45. URL http://dx.doi.org/10.1002/cpz1.45.

Christine W. Bang and Vanessa Didelez. Do we become wiser with time? on causal equivalence with tiered background knowledge. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI '23. JMLR.org, 2023.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. Dagma: Learning DAGs via m-matrices and a log-determinant acyclicity characterization. *Advances in Neural Information Processing Systems*, 35:8226–8239, 2022.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Rajesh Gururaghavendran and Eleanor J Murray. Can algorithms replace expert knowledge for causal inference? a case study on novice use of causal discovery. *American Journal of Epidemiology*, August 2024. ISSN 1476-6256. doi: 10.1093/aje/kwae338. URL http://dx.doi.org/10.1093/aje/kwae338.

Zhijing Jin, Jiarui Liu, Zhiheng LYU, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona T. Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vqIH0ObdqL.

Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Mach. Learn. Res.*, 8:613–636, 2007. doi: 10.5555/1314498.1314520. URL https://dl.acm.org/doi/10.5555/1314498.1314520.

Neville K. Kitson and Anthony C. Constantinou. Causal discovery using dynamically requested knowledge. *Knowledge-Based Systems*, 314:113185, April 2025. ISSN 0950-7051. doi: 10.1016/j.knosys.2025.113185. URL http://dx.doi.org/10.1016/j.knosys.2025.113185.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2023. URL https://arxiv.org/abs/2305.00050.

Xiaoyu Liu, Paiheng Xu, Junda Wu, Jiaxin Yuan, Yifan Yang, Yuhang Zhou, Fuxiao Liu, Tianrui Guan, Haoliang Wang, Tong Yu, Julian McAuley, Wei Ai, and Furong Huang. Large language models and causal inference in collaboration: A comprehensive survey, 2024. URL https://arxiv.org/abs/2403.09606.

Marloes H. Maathuis, Markus Kalisch, and Peter Bühlmann. Estimating high-dimensional intervention effects from observational data. *The Annals of Statistics*, 37(6A), December 2009. ISSN 0090-5364. doi: 10.1214/09-aos685. URL http://dx.doi.org/10.1214/09-AOS685.

Narmada Naik, Ayush Khandelwal, Mohit Joshi, Madhusudan Atre, Hollis Wright, Kavya Kannan, Scott Hill, Giridhar Mamidipudi, Ganapati Srinivasa, Carlo Bifulco, Brian Piening, and Kevin Matlock. Applying large language models for causal structure learning in non small cell lung cancer, 2023. URL https://arxiv.org/abs/2311.07191.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Emilija Perkovic, Johannes Textor, Markus Kalisch, and Marloes H. Maathuis. Complete graphical characterization and construction of adjustment sets in markov equivalence classes of ancestral graphs. *J. Mach. Learn. Res.*, 18:220:1–220:62, 2017. URL https://jmlr.org/papers/v18/16-319.html.

J. Peters and Peter Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural Computation*, 27:771–799, 2015. URL https://api.semanticscholar.org/CorpusID:13055772.

Anne H Petersen, Merete Osler, and Claus T Ekstrøm. Data-driven model building for life-course epidemiology. *American Journal of Epidemiology*, 190(9):1898–1907, March 2021. ISSN 1476-6256. doi: 10.1093/aje/kwab087. URL http://dx.doi.org/10.1093/aje/kwab087.

Wai-Yin Poon and Sik-Yum Lee. Maximum likelihood estimation of multivariate polyserial and polychoric correlation coefficients. *Psychometrika*, 52(3): 409–430, September 1987. ISSN 1860-0980. doi: 10.1007/bf02294364. URL http://dx.doi.org/10.1007/BF02294364.

Marco Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35 (3), 2010. doi: 10.18637/jss.v035.i03. URL https://doi.org/10.18637/jss.v035.i03.

Bill Shipley. A new inferential test for path models based on directed acyclic graphs. *Structural Equation Modeling: A Multidisciplinary Journal*, 7(2):206–218, 2000. doi: 10.1207/S15328007SEM0702_4.

Pater Spirtes, Clark Glymour, Richard Scheines, Stuart Kauffman, Valerio Aimale, and Frank Wimberly. Constructing Bayesian network models of gene expression networks from microarray data. 2000.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer New York, 1993. ISBN 9781461227489. doi: 10.1007/978-1-4612-2748-9. URL http://dx.doi.org/10.1007/978-1-4612-2748-9.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, 2001. ISBN 9780262284158. doi: 10.7551/mitpress/1754.001.0001. URL http://dx.doi.org/10.7551/mitpress/1754.001.0001.

Peter W G Tennant, Eleanor J Murray, Kellyn F Arnold, Laurie Berrie, Matthew P Fox, Sarah C Gadd, Wendy J Harrison, Claire Keeble, Lynsie R Ranker, Johannes Textor, Georgia D Tomova, Mark S Gilthorpe, and George T H Ellison. Use of directed acyclic graphs (DAGs) to identify confounders in applied health research: review and recommendations. *International Journal of Epidemiology*, 50(2):620–632, December 2020. ISSN 1464-3685. doi: 10.1093/ije/dyaa213. URL http://dx.doi.org/10.1093/ije/dyaa213.

Aniket Vashishtha, Abbavaram Gowtham Reddy, Abhinav Kumar, Saketh Bachu, Vineeth N Balasubramanian, and Amit Sharma. Causal inference using LLM-guided discovery, 2023. URL https://arxiv.org/abs/2310.15117.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e347c51419ffb23ca3fd5050202f9c3d-Paper.pdf.

Expert-In-The-Loop Causal Discovery: Iterative Model Refinement Using Expert Knowledge (Supplementary Material)

Ankur Ankan[1] **Johannes Textor**[1]

[1]Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

A PROMPTS USED FOR LLM

```
You are an expert in Social Science. Following are the descriptions of two variables:

<A>: {description of variable A}
<B>: {description of variable B}

Which of the following two options is the most likely causal direction between these
variables:

1. <A> causes <B>
2. <B> causes <A>

Return a single letter answer between the choices above; Do not provide any reasoning
in the answer; Do not add any text formatting to the answer.
```

Figure 9: Prompt used for the LLM. Here the variable descriptions are replaced with description provided in Fig. 10

```
          Age: The age of a person
    Workclass: The workplace where the person is employed such as Private industry,
               or self employed
    Education: The highest level of education the person has finished.
MaritalStatus: The marital status of the person
   Occupation: The kind of job the person does. For example, sales, craft repair,
               clerical.
 Relationship: The relationship status of the person.
         Race: The ethnicity of the person.
          Sex: The sex or gender of the person.
 HoursPerWeek: The number of hours per week the person works.
NativeCountry: The native country of the person.
       Income: The income i.e. amount of money the person makes.
```

Figure 10: Variable descriptions used for prompting the LLM