Hybrid Top-Down Global Causal Discovery with Local Search for Linear and Nonlinear Additive Noise Models

Sujai Hiremath
Cornell Tech
sh2583@cornell.edu

Jacqueline Maasch
Cornell Tech
jam887@cornell.edu

Mengxiao Gao
Tsinghua University
gaomx21@mails.tsinghua.edu.cn

Promit Ghosal
University of Chicago
promit@uchicago.edu

Kyra Gan
Cornell Tech
kyragan@cornell.edu

Abstract

Learning the unique directed acyclic graph corresponding to an unknown causal model is a challenging task. Methods based on functional causal models can identify a unique graph, but either suffer from the curse of dimensionality or impose strong parametric assumptions. To address these challenges, we propose a novel hybrid approach for global causal discovery in observational data that leverages local causal substructures. We first present a topological sorting algorithm that leverages ancestral relationships in linear structural causal models to establish a compact top-down hierarchical ordering, encoding more causal information than linear orderings produced by existing methods. We demonstrate that this approach generalizes to nonlinear settings with arbitrary noise. We then introduce a nonparametric constraint-based algorithm that prunes spurious edges by searching for local conditioning sets, achieving greater accuracy than current methods. We provide theoretical guarantees for correctness and worst-case polynomial time complexities, with empirical validation on synthetic data.

1 Introduction

Causal graphical models compactly represent the *data generating processes* (DGP) of complex systems, including physical, biological, and social domains. Access to the true causal graph or its substructures can offer mechanistic insights [31, 13] and enable downstream causal inference, including effect estimation [11, 34, 2, 8, 16, 35]. In practice, the true causal graph is often unknown, and can be challenging to assume using domain knowledge. In such limited-knowledge settings, we can instead rely on causal discovery algorithms that learn the causal graph from observational data in a principled, automated manner [41, 7].

Traditional approaches to causal discovery infer causal relationships either through conditional independence relations (PC [40]) or goodness-of-fit measures (GES [3], GRaSP [12]). While these discovery methods can provide flexibility by not requiring assumptions over the functional form of the DGP, they are generally worst-case exponential in time complexity and learn Markov equivalence classes (MEC) rather than unique *directed acyclic graphs* (DAGs) [19]. Therefore, additional modeling assumptions are often necessary for time-efficient and accurate global discovery.

Certain parametric assumptions can enable recovery of the unique ground truth DAG, e.g., assuming a particular *functional causal model* (FCM) [48]. Under the *additive noise model* (ANM), we obtain unique identifiability by assuming linear causal mechanisms with non-Gaussian noise distributions

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-4231

[37, 36] or nonlinear causal functions with arbitrary noise [10]. Under the independent additive noise assumption, the causal parents of a variable are statistically independent of its noise term. For this class of models, discovery often entails regressing the variable of interest against its hypothesized parent set and testing for marginal independence between this set and the residual term [25].

Current FCM approaches to global causal discovery trade off between two main issues, suffering from either 1) strong parametric assumptions over the noise or functional form (or both) or 2) the use of high-dimensional nonparametric regressions, which require large sample sizes for reliable estimation and do not scale to large graphs. In addition, current FCM-based methods are ill-suited for causal discovery in sparse causal graphs, a setting that characterizes many high-dimensional applications (e.g., analysis of genetic data in healthcare applications) [6, 47, 46, 15].

Contributions We propose a hybrid causal discovery approach to graph learning that combines functional causal modeling with constraint-based discovery. We depart from previous methods by characterizing conditions that allow us to search for and exploit local, rather than global, causal relationships between vertices. These local relationships stem from root vertices: this motivates a top-down, rather than bottom-up, approach. Thus, we learn the topological sort and discover true edges starting from the roots rather than the leaves, as in existing methods [25, 30, 21]. This approach leverages sparsity in both the ordering phase and edge discovery phase to reduce the size of conditioning sets, as well as the number of high-dimensional regressions. We summarize our major contributions as follows:

- We introduce a topological ordering algorithm LHTS for linear non-Gaussian ANMs that exploits local ancestor-descendent relationships to obtain a compact hierarchical sort.
- We introduce a topological ordering algorithm NHTS for nonlinear ANMs that exploits local parent-child relationships to run fewer high-dimensional regressions than traditional methods, achieving lower sample complexity.
- We introduce a constraint-based algorithm ED that nonparametrically prunes spurious edges from a discovered topological ordering, leveraging local properties of causal edges to use smaller conditioning sets than traditional sparse regression techniques.
- We achieve accurate causal discovery in synthetic data, outperforming baseline methods.

Organization After describing the preliminaries in Section 2, we introduce the linear problem setting in Section 3, establishing the connection between ancestral relationships and causal active paths and introducing a *linear hierarchical topological sorting* algorithm (LHTS). Next, we extend our method to the nonlinear setting in Section 4 by establishing the connection between parental relationships and active causal paths, introducing a *nonlinear hierarchical topological sorting* algorithm (NHTS). In section 5, we establish a sufficient conditioning set for determining edge relations and introduce an efficient *edge discovery* algorithm (ED). We then test LHTS, NHTS and ED in synthetic experiments in Section 6. To conclude, we discuss future work that might generalize our approach to full ANMs.

Related Work Our work is related to two kinds of discovery methods that explicitly leverage the topological structure of DAGs: 1) permutation-based approaches, and 2) FCM-based approaches.

The original permutation-based approach SP [27] searches over the space of variable orderings to find permutations that induce DAGs with minimal edge counts. Authors in [39] introduce greedy variants of SP (such as GSP) that maintain asymptotic consistency; GRaSP [12] relaxes the assumptions of prior methods to obtain improvements in accuracy. These methods highlight the importance of using permutations for efficient causal discovery, but generally suffer from the need to bound search runtime with heuristics, poor sample efficiency in high dimensional settings, and are unable to recover a unique topological ordering or DAG ([22]).

On the other hand, the recent stream of FCM-based approaches decompose graph learning into two phases: 1) learning the topological sort, i.e., inferring a causal ordering of the variables; and 2) edge discovery, i.e., identifying edges consistent with the causal ordering [38, 25, 1, 30, 21, 32, 20].

The literature on topological ordering algorithms for ANMs is organized along the types of parametric assumptions made on both the functional forms and noise distributions of the underlying DGP. Early approaches like ICA-LiNGAM [37] and DirectLiNGAM [38] focus on learning DAGs generated by linear functions and non-Gaussian noise terms. Recent work leverages score matching to obtain the causal ordering in settings with nonlinear functions and Gaussian noise: SCORE [30] and DAS [21] exploit particular variance properties, while DiffAN estimates the score function with a diffusion model [32]. NoGAM [20] generalizes the score-matching procedure of SCORE to nonlinear causal

mechanisms with arbitrary noise distributions. RESIT [25] leverages residual independence results in nonlinear ANMs to identify topological orderings when the noise distribution is arbitrary. NoGAM and RESIT both rely on high-dimensional nonparametric regression.

Once a topological ordering is obtained, spurious edges are pruned. Works that are agnostic to the distribution of noise often use a parametric approach, implementing either a form of sparse regression (e.g., Lasso regression [18]) or a version of additive hypothesis testing with generalized additive models (GAMs) [17] (e.g., CAM-pruning [1]). RESIT [25] provides another alternative edge pruning procedure for nonlinear ANMs, relying again on high-dimensional nonparametric regression.

2 Preliminaries

We focus on *structural causal model*s (SCMs) represented as DAGs. These graphs describe the causal relationships between variables, where an edge $x_i \to x_j$ implies that x_i has a direct causal influence on x_j. Let $G = (V, E)$ be a DAG on $|V| = d$ vertices, where E represents directed edges. To define pairwise relationships between vertices, we let $\text{Ch}(x_i)$ denote the children of x_i such that $x_j \in \text{Ch}(x_i)$ if and only if $x_i \to x_j$, and $\text{Pa}(x_i)$ denote the parents of x_i such that $x_j \in \text{Pa}(x_i)$ if and only if $x_j \to x_i$. Similarly, let $\text{An}(x_i)$ denote the ancestors of x_i such that $x_j \in \text{An}(x_i)$ if and only if there exists a directed path $x_j \dashrightarrow x_i$, and $\text{De}(x_i)$ denote the descendants of x_i such that $x_j \in \text{De}(x_i)$ if and only if there exists a directed path $x_i \dashrightarrow x_j$. Vertices can be classified based on the totality of their pairwise relationships: x_i is a *root* if and only if $\text{Pa}(x_i) = \emptyset$, a *leaf* if and only if $\text{Ch}(x_i) = \emptyset$, an isolated vertex if x_i is both a root and a leaf, and an intermediate vertex otherwise. See an illustrative DAG in Figure 1. Vertices can also be classified in terms of triadic relationships: x_i is a confounder of x_j, x_k if and only if $x_i \in \text{An}(x_j) \cap \text{An}(x_k)$; a mediator of x_j to x_k if and only if $x_i \in \text{De}(x_j) \cap \text{An}(x_k)$; and a collider between x_j and x_k if and only if $x_i \in \text{De}(x_j) \cap \text{De}(x_k)$.

Undirected paths that transmit causal information between vertices x_j, x_k can be differentiated into *frontdoor* and *backdoor paths* [42]. A frontdoor path is a directed path $x_j \dashrightarrow \cdots \dashrightarrow x_k$ that starts with an edge out of x_j, and ends with an edge into x_k. A backdoor path is a path $x_j \dashleftarrow \cdots \dashrightarrow x_k$ that starts with an edge into x_j, and ends with an edge into x_k. Paths that start and end with an edge out of x_j and x_k ($x_k \dashrightarrow \cdots \dashleftarrow x_k$) do not transmit causal information between x_j, x_k.

Paths between two vertices are further classified, relative to a vertex set $\mathbf{Z}$, as either *active* or *inactive* [42]. A path between vertices x_j, x_k is active relative to $\mathbf{Z}$ if every node on the path is active relative to $\mathbf{Z}$. Vertex x_i on a path is active if one of the following holds: 1) x_i is not a collider and $x_i \notin \mathbf{Z}$, 2) x_i is a collider and $x_i \in \mathbf{Z}$, 3) x_i is a collider and $x_i \notin \mathbf{Z}$, but $\text{De}(x_i) \cap \mathbf{Z} \neq \emptyset$. An inactive path is simply a path that is not active. Following convention, throughout the rest of the paper we will describe causal paths as active or inactive with respect to $\mathbf{Z} = \emptyset$ unless otherwise specified.

Definition 2.1 (Topological Orderings)**.** *Consider a given DAG $G = (V, E)$. A topological sort (linear order) is a mapping $\pi : V \to \{0, 1, \ldots, |V| - 1\}$, such that if $x_i \in Pa(x_j)$, then x_i appears before x_j in the sort π: $\pi(x_i) < \pi(x_j)$. A hierarchical sort (between a partial and linear order) is a mapping $\pi_L : V \to \{0, 1, \ldots, |V| - 1\}$, such that if $Pa(x_i) = \emptyset$, then $\pi_L(x_i) = 0$, and if $Pa(x_i) \neq \emptyset$, then $\pi_L(x_i)$ equals the maximum length of the longest directed path from each root vertex to x_i, i.e., $\pi_L(x_i) = 1 + \max\{\pi_L(x_j) : x_j \in Pa(x_i)\}$.*

We note that the hierarchical sort is unique, and that it coincides with a topological sort when the number of layers equals $|V|$, i.e., the DAG is complete.

Definition 2.2 (ANMs)**.** *ANMs [9] are a popular general class of SCMs defined over a DAG G with*

$$x_i = f_i(Pa(x_i)) + \varepsilon_i, \forall x_i \in V, \tag{1}$$

where f_is are arbitrary functions and ε_is are independent arbitrary noise distributions.

This model implicitly assumes the causal Markov condition and acyclicity; we adopt the aforementioned assumptions, as well as faithfulness [41].

3 Linear Setting

We first restrict our attention to ANMs that feature only linear causal functions f, known as Linear Non-Gaussian Acyclic causal Models (LiNGAMs). Following [38], we note that a LiNGAM can

 https://doi.org/10.52202/079017-4231

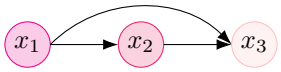


Figure 1: Illustrative DAG, where x_1 is a root, x_3 is a leaf, $x_3 \in \text{Ch}(x_2), x_3 \in \text{De}(x_1)$.

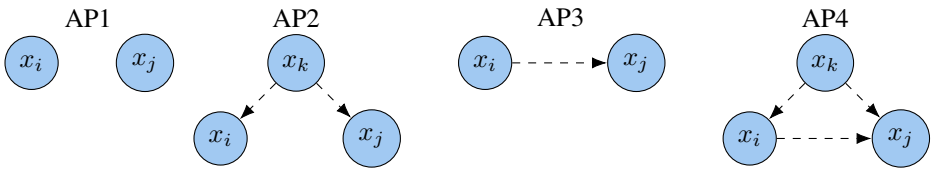


Figure 2: Enumeration of active causal path relation types between a pair of nodes x_i and x_j. Dashed arrows indicate ancestorship.

be represented as a $d \times d$ adjacency matrix $B = \{b_{ij}\}$, where b_{ij} is the coefficient from x_j to x_i. Note that, for any topological ordering π of a LiNGAM, if $\pi(x_j) > \pi(x_i)$, then $b_{ji} = 0$. Thus, each $x_i \in V$ admits the following compact representation: $x_i = \sum_{\pi(x_j)<\pi(x_i)} b_{ij} x_j + \varepsilon_i$.

Identifiability Identifiability conditions for LiNGAMs [37] primarily concern the distribution of errors ε_i: under Gaussianity, distinct linear DGPs can admit the same joint distribution, making them impossible to distinguish. Shimizu et al. [37] generalize this intuition with *independent component analysis* (ICA) [4] to provide a multivariate identifiability condition for LiNGAMs (see Appendix A.1). In this section, we adopt the aforementioned condition.

Ancestral Relations and Active Causal Paths We first establish the connection between ancestral relationships and active causal paths. We exhaustively enumerate and define the potential pairwise causal ancestral path relations in Figure 2 and Lemma 3.1 (proof in Appendix A.2):

Lemma 3.1 (Active Causal Ancestral Path Relation Enumeration)**.** *Each pair of distinct nodes $x_i, x_j \in V$ can be in one of four possible active causal ancestral path relations: AP1) no active path exists between x_i, x_j; AP2) there exists an active backdoor path between x_i, x_j, but there is* no *active frontdoor path between them; AP3) there exists an active frontdoor path between x_i, x_j, but there is* no *active backdoor path between them; AP4) there exists an active backdoor path between x_i, x_j,* and *there exists an active frontdoor path between them.*

Next, in Lemma 3.2, we summarize the connection between causal paths and ancestral relationships (proof in Appendix A.3):

Lemma 3.2. *The ancestral relationship between a pair of distinct nodes $x_i, x_j \in V$ can be expressed using active causal path relations: x_i, x_j are not ancestrally related if and only if they are in AP1 or AP2 relation; and x_i, x_j are ancestrally related if and only if they are in AP3 or in AP4 relation.*

The active causal ancestral path relation of a pair of nodes x_i, x_j that are not ancestrally related can be determined through marginal independence testing and sequential univariate regressions as illustrated in Lemmas 3.3 and 3.4 (proofs in Appendices A.4, A.5):

Lemma 3.3 (AP1)**.** *Vertices x_i, x_j are in AP1 relation if and only if $x_i \perp\!\!\!\perp x_j$.*

Lemma 3.4 (AP2)**.** *Let M be the set of mutual ancestors between a pair of vertices x_i and x_j, i.e., $M = An(x_i) \cap An(x_j)$. Let x_i^M, x_j^M be the result of sequentially regressing all mutual ancestors in M out of x_i, x_j with univariate regressions, in any order. Then, let r_i^j be the residual of x_j^M regressed on x_i^M, and r_j^i be the residual of x_i^M regressed on x_j^M. Suppose $x_i \not\perp\!\!\!\perp x_j$. Then, x_i, x_j are in AP2 relation if and only if $r_i^j \perp\!\!\!\perp x_i^M$ and $r_j^i \perp\!\!\!\perp x_j^M$.*

If a pair of nodes x_i, x_j is ancestrally related, fully ascertaining their ancestral relation involves discerning between the ancestor and descendent. As illustrated in Lemmas 3.5 and 3.6 (proofs in Appendices A.6, A.7), this can be determined through marginal independence testing after sequential univariate regressions with respect to the mutual ancestor set.

Lemma 3.5 (AP3)**.** *Let r_i^j be the residual of the x_j regressed on x_i, and r_j^i be the residual of x_i regressed on x_j. Vertices x_i, x_j are in AP3 relation if and only if $x_i \not\perp\!\!\!\perp x_j$ and one of the following*

Algorithm 1 LHTS: Linear Hierarchical Topological Sort

input: features $x_1, \ldots, x_d \in V$.
output: hierarchical topological sort $\pi_L(V)$.
initialize: ancestral relations set ARS.
Stage 1: AP1 Relations
for all pairs x_i, x_j **do**
  if $x_i \perp\!\!\!\perp x_j$ **then**
    Store x_i, x_j are not related in ARS.
Stage 2: AP3 Relations
for all x_i, x_j with unknown relations **do**
  Set r_i^j as residual of x_j regressed on x_i
  Set r_j^i as residual of x_i regressed on x_j.
  if $r_i^j \perp\!\!\!\perp x_i$ **and** $r_j^i \not\perp\!\!\!\perp x_j$ **then**
    Store $x_i \in An(x_j)$ in ARS.
  if $r_i^j \not\perp\!\!\!\perp x_i$ **and** $r_j^i \perp\!\!\!\perp x_j$ **then**
    Store $x_j \in An(x_i)$ in ARS.
Stage 3: AP2 and AP4 Relations
while $\exists\, x_i, x_j$ with unknown relations **do**
  for x_i, x_j with unknown relations **do**
    Sequentially regress x_i, x_j on mutual ancestors, store final residuals r_i^j, r_j^i.
    if $r_i^j \perp\!\!\!\perp x_i$ **and** ${}_I r_j^i \perp\!\!\!\perp x_j$ **then**
      Store x_i, x_j are not related in ARS.
    else if $r_i^j \perp\!\!\!\perp x_i$ **and** $r_j^i \not\perp\!\!\!\perp x_j$ **then**
      Store $x_i \in An(x_j)$ in ARS.
    else if $r_i^j \not\perp\!\!\!\perp x_i$ **and** $r_j^i \perp\!\!\!\perp x_j$ **then**
      Store $x_j \in An(x_i)$ in ARS.
Stage 4: Obtain sort by subroutine AS
return: $\pi_L(V) \leftarrow$ **AS**(ARS)

holds: 1) $x_i \perp\!\!\!\perp r_i^j$ and $x_j \not\perp\!\!\!\perp r_j^i$, corresponding to $x_i \in An(x_j)$, or 2) $x_i \not\perp\!\!\!\perp r_i^j$ and $x_j \perp\!\!\!\perp r_j^i$, corresponding to $x_j \in An(x_i)$.

Lemma 3.6 (AP4)**.** *Let M, $r_i^j, r_j^i, x_i^M, x_j^M$ be as defined in Lemma 3.4. Suppose x_i, x_j not in AP3 relation. Then, x_i, x_j are in AP4 relation if and only if $x_i \not\perp\!\!\!\perp x_j$ and one of the following holds: $r_i^j \perp\!\!\!\perp x_i^M$ and $r_j^i \not\perp\!\!\!\perp x_j^M$ corresponding to $x_i \in An(x_j)$, or 2) $r_i^j \not\perp\!\!\!\perp x_i^M$ and $r_j^i \perp\!\!\!\perp x_j^M$ corresponding to $x_j \in An(x_i)$.*

Linear Hierarchical Topological Sort We propose LHTS in Algorithm 1 to leverage the above results to discover a hierarchical topological ordering. Marginal independence tests and pairwise regressions are used to identify active causal ancestral path relations between pairs of vertices.

LHTS discovers all pairwise active causal ancestral path relations: Stage 1 discovers all AP1 relations using marginal independence tests; Stage 2 discovers all AP3 relations by detecting when pairwise regressions yield an independent residual in only one direction; Stage 3 iteratively discovers all AP2 and AP4 relations through marginal independence tests and pairwise sequential regressions involving mutual ancestors. Note that Stage 2 can be viewed as a special case of Stage 3, where the mutual ancestor set is empty. The process utilizes proofs provided in Appendices A.4, A.6, A.5, and A.7, respectively. Stage 4 uses the complete set of ancestral relations to build the final hierarchical topological sort via the subroutine **AS** (Algorithm 4 in Appendix A.8) by recursively peeling off vertices with no unsorted ancestors. We show the correctness of Algorithm 1 in Theorem 3.7 (proof in Appendix A.9) and subroutine AS (proof in Appendix A.8), as well as the worst case time complexity in Theorem 3.8 (proof in Appendix A.10). We provide a walk-through of LHTS in Appendix A.11.

Theorem 3.7. *Given a graph G, Algorithm 1 asymptotically finds a correct hierarchical sort of G.*

Theorem 3.8. *Given n samples of d vertices generated by a LiNGAM, the worst case runtime complexity of Algorithm 1 is upper bounded by $O(d^3 n^2)$.*

LHTS can be seen as a generalization of DirectLiNGAM [38], where they recursively identify root nodes as vertices that have either AP1 or AP3 relations with every remaining vertex at each step. The next lemma, Lemma 3.9 (proof in Appendix A.12) implies that LHTS discovers all root vertices in Stage 2, obtaining the first layer in the hierarchical sort. LHTS extends DirectLiNGAM by discovering AP2 and AP4 relations, allowing us to recover a compact hierarchical sort.

Lemma 3.9. *A vertex is a root vertex if and only if it has AP1 or AP3 relations with all other vertices.*

4 Nonlinear Setting

In this section, we develop a version of Algorithm 1 for ANMs that feature only nonlinear causal functions f. Rather than detecting ancestor-descendant relationships, we outline the connection

https://doi.org/10.52202/079017-4231

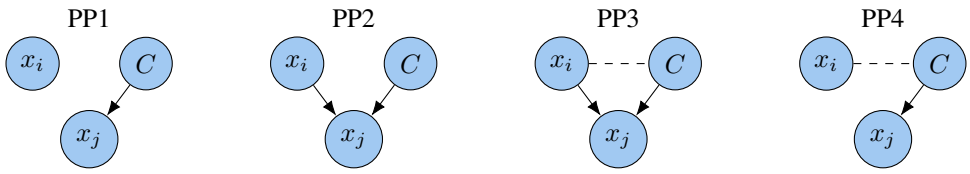


Figure 3: Enumeration of the potential active causal paths among a fixed variable x_j, one of its potential parents x_i, and $C = \text{PA}(x_j) \setminus x_i$. Solid arrows denote parenthood relations, and undirected dashed connections indicate the existence of active paths.

between active causal paths and parent-child relationships. We assume the unique identifiability of the nonlinear ANM as described by Peters et al. [25], and provide the conditions in Appendix B.1.

Nonlinear Topological Sort In the linear setting, we determined ancestral relations through a sequence of pairwise regressions that led to independent residuals. However, a naive extension of this method into the nonlinear setting would fail, as regressions yield independent residuals under different conditions in the nonlinear case. For clarity, we demonstrate how LHTS fails to correctly recover causal relationships in an exemplary 3-node DAG with nonlinear causal mechanisms.

Consider a DAG G with three vertices x_1, x_2, x_3, where $x_1 \to x_3, x_2 \to x_3$. The functional causal relationships are nonlinear, given by $x_1 = \varepsilon_1, x_2 = \varepsilon_2, x_3 = x_1 x_2 + \varepsilon_3$, where ε_i are mutually independent noise variables. We focus on whether LHTS can recover the parent-child relationship between x_1 and x_3. LHTS finds that the relationship between x_1, x_3 is unknown in Stage 1. In Stage 2, LHTS runs pairwise regressions between x_1, x_3 but *incorrectly concludes that x_1, x_3 are not in AP3 relation* because neither pairwise regression provides an independent residual; both parents of x_3 must be included in the covariate set for an independent residual to be recovered.

To handle nonlinear causal relationships, we shift our focus to searching for a different set of local substructures: the connection between active causal parental paths and the existence of parent-child relationships. We will use the existence of specific parent-child relationships to first obtain a superset of root vertices, then prune away non-roots. Once all root vertices are identified, we build the hierarchical topological sort through nonparametric regression, layer by layer.

Given a vertex x_j and one of its potential parents x_i, we first provide an enumeration of all potential active casual parental path types between them with respect to $C = \text{PA}(x_j) \setminus x_i$ (which could potentially be the empty set) in Figure 3 and Lemma 4.1 (proof in Appendix B.2).

Lemma 4.1 (Active Causal Parental Path Relation Enumeration)**.** *Let $x_i, x_j \in V$ be a pair of distinct nodes, where x_i is one of the potential parents of x_j. Let $C = PA(x_j) \setminus x_i$. There are in total four possible types of active causal parental path relations between x_i and x_j with respect to C: PP1) x_i and x_j are* not *directly causally related, and* no *active path exists between x_i and C; PP2) x_i directly causes x_j ($x_i \to x_j$), and* no *active path exists between x_i and C; PP3) x_i directly causes x_j ($x_i \to x_j$), and there exists an active path between x_i and C; PP4) x_i and x_j are* not *directly causally related, and there exists an active path between x_i and C.*

Next, for a pair of distinct vertices $x_i, x_j \in V$, we establish the connection between pairwise independence properties and active causal parental path relations in Lemma 4.2 (proof in Appendix B.3). This allows us to reduce the cardinality of the potential pairs of vertices under consideration in the later stages of the algorithm.

Lemma 4.2 (Non-PP1)**.** *Vertices $x_i, x_j \in V$ are not in PP1 relation if and only if $x_i \not\perp\!\!\!\perp x_j$.*

In Lemma 4.3, we show that all pairs of vertices that are in PP2 relation can be identified through local nonparametric regressions (proof in B.4).

Lemma 4.3 (PP2)**.** *Let $x_i, x_j \in V$, $P_{ij} = \{x_k \in V : x_k \perp\!\!\!\perp x_i, x_k \not\perp\!\!\!\perp x_j\}$, r_i^j be the residual of x_j nonparametrically regressed on x_i, and $r_{i,P}^j$ be the residual of x_j nonparametrically regressed on x_i and all $x_k \in P_{ij}$. Suppose x_i and x_j are not in PP1 relation. Then, x_i and x_j are in PP2 relation if and only if one of the following holds: 1) $x_i \perp\!\!\!\perp r_i^j$ or 2) $x_i \perp\!\!\!\perp r_{i,P}^j$.*

Condition 1) of Lemma 4.3 is relevant when $C = \emptyset$: in this case, pairwise regression identifies x_i as a parent of x_j. Condition 2) is relevant when $C \neq \emptyset$: we leverage the independence of x_i from the rest

Algorithm 2 NHTS: Nonlinear Hierarchical Topological Sort

input: vertices $x_1, \ldots, x_d \in V$.
output: hierarchical sort $\pi_L(V)$.
initialize: parent relations set PRS.
Stage 1: Not-PP1 Relations
for all pairs x_i, x_j **do**
  if $x_i \not\perp\!\!\!\perp x_j$ **then**
    PRS : x_i, x_j not in PP1 relations.
if x_i is in PP1 relation with all vertices **then**
  PRS: x_i is isolated, sort x_i: $\pi_L(x_i) = 0$.
Stage 2: PP2 Relations
for all x_i, x_j not in PP1 relations **do**
  Set r_i^j as residual of x_j regressed on x_i.
  Set $r_{i,P}^j$ as residual of x_j regressed on x_i, P_{ij}.
  if $x_i \perp\!\!\!\perp r_i^j$ **or** $x_i \not\perp\!\!\!\perp r_{i,P}^j$ **then**
    PRS : x_i, x_j are in PP2 relations, $x_i \in Pa(x_j)$.
Stage 3: Root Identification
for $x_i \in$ PP2 relation **do**
  Check if x_i does not d-separate any child from all marginally dependent vertices. If so, then update PRS : x_i is a root, sort x_i into layer 0, $\pi_L(x_i) = 0$.
Stage 4: Obtain Sort
for $k \in \{1, \ldots, d-1\}$ **do**
  for all unsorted x_i **do**
    Set r_i as the residual of x_i regressed on sorted features in π_H.
    If $r_i \perp\!\!\!\perp x_j \forall x_j \in \pi_H$, add x_i into the current layer $\pi_L(x_i) = k$.
return $\pi_L(V)$

of x_j's parents to generate P_{ij}, a set that contains C, but does not contain any of x_j's descendants. If an independent residual were to be recovered by nonparametrically regressing x_j onto x_i and P_{ij}, we identify x_i as a parent of x_j.

Let W be the set of all parent vertices that are in PP2 relation with at least one vertex, i.e., the union of x_i satisfying either condition of Lemma 4.3. We now show that all non-isolated root vertices are contained in W, and they can be differentiated from non-roots in W that are also in PP2 relations.

Lemma 4.4 (Roots)**.** *All non-isolated root vertices are contained in W. In addition, $x_i \in W$ is a root vertex if and only if 1) x_i is not a known descendant of any $x_j \in W$, and 2) for each $x_j \in W$, either a) $x_i \perp\!\!\!\perp x_j$, b) x_j is in PP2 relation to x_i, i.e., $x_j \in Ch(x_i)$, or c) there exists a child of x_i, denoted by x_k, that cannot be d-separated from x_j given x_i, i.e., $x_j \not\perp\!\!\!\perp x_k | x_i$.*

Lemma 4.4 relies on the following intuition: for any non-isolated root vertex x_i and descendant $x_k \in \mathrm{De}(x_i)$, there exists a child of x_i that 1) lies on a directed path between x_i and x_k, and 2) is in PP2 relation with x_i. Vertex x_i will fail to d-separate this specific child from the marginally dependent non-root descendant x_k.[1] On the other hand, non-roots in W must d-separate the children they are in PP2 relation to from all marginally dependent roots: this asymmetry allows non-roots to be pruned. See Appendix B.5 for a detailed proof.

We propose a method, Algorithm 2, that leverages the above results to discover a hierarchical topological ordering: we first use the above lemmas to identify the roots, then use nonparametric regression to discover the rest of the hierarchical sort.

In Stage 1, we discover all pairs x_i, x_j not in PP1 relation by testing for marginal dependence; in Stage 2, we leverage Lemma 4.3 to find the vertex pairs that are in PP2 relations, a superset of the root vertices; in Stage 3 we prune non-roots by finding they d-separate their children from at least one dependent vertex in W (Lemma 4.4); in Stage 4 we identify vertices in the closest unknown layer by regressing them on sorted nodes and finding independent residuals. We show the correctness of Algorithm 2 in Theorem 4.5 (proof in Appendix B.6), and the worst case time complexity in Theorem 4.6 (proof in Appendix B.7). We provide a walk-through of Algorithm 2 in Appendix B.8.

Theorem 4.5. *Given a graph G, Algorithm 2 asymptotically finds a correct hierarchical sort of G.*

Theorem 4.6. *Given n samples of d vertices generated by a identifiable nonlinear ANM, the worst case runtime complexity of Algorithm 2 is upper bounded by $O(d^3 n^3)$.*

The number of nonparametric regressions run by NHTS in each step is actually inversely related to the size of the covariate sets, while the number of regressions in each step of RESIT and NoGAM are directly proportionate to the covariate set size. We provide a formal analysis of the reduction in complexity for the worst case (fully connected DAG) in Theorem 4.7 (proof in Appendix B.9):

[1] Vertices x_i, x_j are said to be d-separated by a set $\mathbf{Z}$ iff there is no active path between x_i, x_j relative to $\mathbf{Z}$.

https://doi.org/10.52202/079017-4231

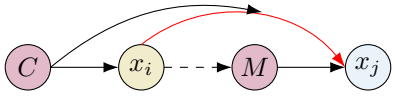


Figure 4: DAG corresponding to Lemma 5.1, which tests whether $x_i \in \text{Pa}(x_j)$ (i.e., whether the red arrow exists).

Algorithm 3 ED: Edge Discovery

input: features $x_1, \ldots, x_d \in V$, topological order π.
initialize: empty parent sets $\{\text{Pa}(x_i)\}_{i=1}^d$.
Step 1: Loop over π
for $j \in \{2, \ldots, d\}$ **do**
  Set x_j equal to j^{th} index of π.
  for $i \in \{j-1, \ldots, 1\}$ **do**
    Set x_i equal to i^{th} index of π.
    Set $Z_{ij} = \text{Pa}(x_i) \cup \text{Pa}(x_j)$.
    Step 2: Test for Parenthood
    if $x_i \not\perp\!\!\!\perp x_j | Z_{ij}$ **then**
      Update $x_i \in \text{Pa}(x_j)$.
return $\{\text{Pa}(x_i)\}_{i=1}^d$.

Theorem 4.7. *Consider a fully connected DAG $G = (V, E)$ with nonlinear ANM. Let $d := |V|$. Let n_k^{NHTS} be the number of nonparametric regressions with covariate set size $k \in [d-2]$ run by NHTS when sorting V; we similarly define n_k^{RESIT} and n_k^{NoGAM} respectively. Then, $n_k^{\text{NHTS}} = d - k$, and $n_k^{\text{RESIT}} = n_k^{\text{NoGAM}} = k + 1$. This implies that for all $k > \frac{d}{2}$, $n_k^{\text{NHTS}} < n_k^{\text{RESIT}} = n_k^{\text{NoGAM}}$.*

5 Edge Discovery

Parent selection from a topological ordering π via regression is traditionally a straightforward task in the infinite sample setting: for each vertex x_i, π establishes $J_i = \{x_k : \pi(x_k) < \pi(x_i)\}$, a superset of x_i's parents that contains none of x_i's descendants. A general strategy for pruning $\text{Pa}(x_i)$ from J_i is to regress x_i on J_i and check which $x_k \in J_i$ are relevant predictors. The key issue is that J_i grows large in high-dimensional graphs: current edge pruning methods either make strong parametric assumptions or suffer in sample complexity. Lasso and GAM methods impose linear and additive models, failing to correctly identify parents in highly nonlinear settings. RESIT assumes a more general nonlinear ANM, but requires huge sample sizes and oracle independence tests for accurate parent set identification: authors in [25] confirm this, saying "despite [our] theoretical guarantee[s], RESIT does not scale well to a high number of nodes."

We propose an entirely nonparametric constraint-based method that uses a local conditioning set Z_{ij} to discover whether $x_i \in \text{Pa}(x_j)$, rather than J_i, outperforming previous methods by relaxing parametric assumptions and conditioning on fewer variables. The following lemma outlines a sufficient condition for determining whether an edge exists between two vertices (proof in Appendix C.1), visualized in Figure 4.

Lemma 5.1 (Parent Discovery)**.** *Let π be a topological ordering, x_i, x_j such that $\pi(x_i) < \pi(x_j)$. Let $Z_{ij} = C_{ij} \cup M_{ij}$, where $C_{ij} = \{x_k : x_k \in Pa(x_i), x_k \not\perp\!\!\!\perp x_j\}$, $M_{ij} = \{x_k : x_k \in Pa(x_j), \pi(x_i) < \pi(x_k) < \pi(x_j)\}$. Then, $x_i \to x_j \iff x_i \not\perp\!\!\!\perp x_j | Z_{ij}$.*

The intuition is that to determine whether $x_i \to x_j$, instead of conditioning on all potential ancestors of x_j, it suffices to condition on potential confounders of x_i, x_j (C_{ij}) and potential mediators between x_i and x_j (M_{ij}). This renders all backdoor and frontdoor paths inactive, except the frontdoor path corresponding to a potential direct edge from x_i to x_j.

Edge Discovery We propose an algorithm that leverages the above results to discover the true edges admitted by any topological ordering by running the described conditional independence test in a specific ordering. We give the implementation for pruning a topological sort here, but our approach can be generalized to a hierarchical version (see Appendix C.2). We show the correctness of Algorithm 3 in Theorem 5.2 (proof in Appendix C.3).

Theorem 5.2. *Given a correct topological ordering of G, Algorithm 3 asymptotically finds correct parent sets $PA(x_i), \forall x_i \in G$.*

The key insight is to check each potential parent-offspring relation using the conditional independence test $x_i \perp\!\!\!\perp x_j | Z_{ij}$ such that previous steps in the algorithm obtain all vertices in both C_{ij} and M_{ij}.

We first fix a vertex x_j whose parent set we want to discover. We then check if vertices ordered before x_j are parents of x_j in reverse order, starting with the vertex immediately previous to x_j in π. This process starts at the beginning of π, meaning we discover parent-offspring relations from root to leaf (see Appendix C.3 for a detailed walk-through and proof). We show the worst case time complexity of Algorithm 3 in Theorem 5.3 (proof in Appendix C.4).

Theorem 5.3. *Given n samples of d vertices generated by a model corresponding to a DAG G, the runtime complexity of ED is upper bounded by $O(d^2n^3)$.*

6 Experiments

Setup Methods[2] are evaluated on 20 DAGs in each trial. The DAGs are randomly generated with the Erdos-Renyi model [5]; the probability of an edge is set such that the average number of edges in each d-dimensional DAG is d. Gaussian, Uniform, or Laplace noise is used as the exogenous error. In experiments for linear topological sorting methods (Figure 5), we use linear causal mechanisms to generate the data; in experiments for nonlinear topological sorting methods (Figure 6) and edge pruning algorithms (Figure 7), we use quadratic causal mechanisms to generate the data. Existing ANM methods are prone to exploiting artifacts that are more common in simulated ANMs than real-world data [28, 29], inflating their performance on synthetic DAGs and leaving real-world applicability an open question. To reduce concerns about such artifacts, data were generated with reduced R^2-sortability [29], and standardized to zero mean and unit variance [28].

Metrics A_{top} is equal to the percentage of edges that can be recovered by the returned topological ordering (an edge cannot be recovered if a child is sorted before a parent). We note that A_{top} is a normalized version of the topological ordering divergence D_{top} defined in [30]. Edge pruning algorithms return a list of predicted parent sets for each vertex: $F_1 = 2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision + Recall}}$ measures the performance of these predictions.

Linear Topological Sorts Figure 5 demonstrates the performance of our linear topological ordering algorithm, LHTS, in comparison with the benchmark algorithms, DirectLiNGAM [38] and R^2-Sort [29]. R^2-Sort is a heuristic sorting algorithm that exploits artifacts common in simulated ANMs; both benchmarks are agnostic to the noise distribution. We observe that both DirectLiNGAM and LHTS significantly outperform R^2-sort. LHTS demonstrates asymptotic correctness in Figure 5(c), achieving near-perfect A_{top} at $n = 2000$. However, LHTS has consistently lower A_{top} than DirectLiNGAM in Figure 5(a). On the other hand, LHTS encodes more causal information: the orderings produced by LHTS in Figure 5(b) had roughly $\sim 70\%$ fewer layers than the orderings produced by DirectLiNGAM, reducing the size of potential parent sets J by identifying many non-causal relationships.

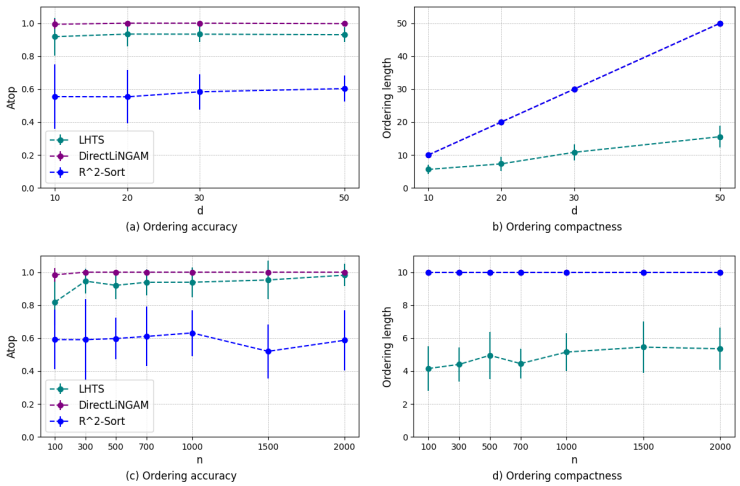


Figure 5: Performance of LHTS on synthetic data. Top row: $n = 500$ with varying dimension d. Bottom row: $d = 10$ with varying sample size n. See Appendix D.1 for runtime results.

[2]Code: https://github.com/Sujai1/hybrid-discovery.

https://doi.org/10.52202/079017-4231

Nonlinear Topological Sorts Figure 6 illustrates the performance of our nonlinear topological sorting algorithm. We take GES [3], GRaSP [12], GSP [39], DirectLiNGAM [38], NoGAM [20] , and R^2-Sort [29] as baseline comparators that are all agnostic to the noise distribution. We excluded PC and RESIT since in general they perform much worse than baseline methods [20]. We note that as DirectLiNGAM, NoGAM, and NHTS are FCM-based methods, they each return a unique topological ordering; however, as GES, GRaSP, and GSP are scoring-based methods [7], they return only a MEC. All topological orderings contained within an MEC satisfy every conditional independence constraint in the data, and therefore are all *equally valid*. To enable a fair comparison, we randomly select one ordering permitted by an outputted MEC for evaluation. We note that NHTS outperformed all baselines, achieving the highest median A_{top} in all trials. Furthermore, as expected from Theorem 4.6, NHTS ran up to $4\times$ faster than NoGAM (see Appendix D.2). We provide additional experiments over DAGs with increasing edge density in Appendix D.3.

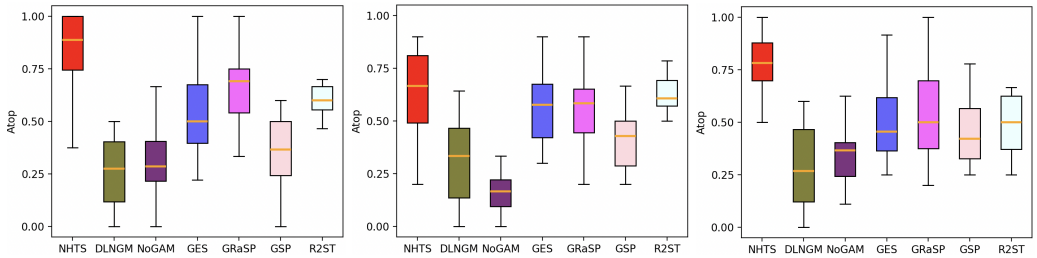


Figure 6: Performance of NHTS on synthetic data, $n = 300$, dimension $d = 10$, with varying error distributions: Gaussian, Laplace, Uniform (left, middle, right). See Appendix D.2 for runtime results.

Edge Pruning Figure 7 illustrates the performance of our edge pruning algorithm ED. We take covariate hypothesis testing with GAMs (CAM-pruning [1]), Lasso regression, and RESIT as baseline comparators that are all agnostic to the noise distribution. All algorithms were given correct topological sorts: ED significantly outperformed all baselines, with the highest median F_1 score in all trials. ED was slower than Lasso, but was significantly faster than the other nonlinear edge pruning algorithms, CAM-pruning and RESIT. RESIT was excluded from higher-dimensional tests due to runtime issues. The poor performance of baseline methods highlights the need for a sample efficient nonparametric method for accurate causal discovery of nonlinear DGPs. We provide additional experiments in settings with increasing density and varying noise distributions in Appendix D.4.

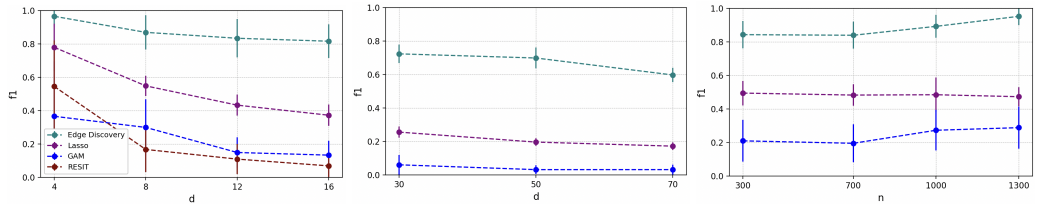


Figure 7: Performance of ED on synthetic data, uniform noise. Left, middle: $n = 300$ with varying dimension d. Right: $d = 10$ with varying sample size n. See Appendix D.5 for runtime results.

Discussion In this paper we developed novel global causal discovery algorithms by searching for and leveraging local causal relationships. We improved on previous topological ordering methods by running fewer regressions, each with lower dimensionality, producing hierarchical topological sorts. Additionally, we improved on previous edge pruning procedures by introducing a nonparametric constraint-based method that conditions on far fewer variables to achieve greater recovery of parent sets. We tested our methods on robustly generated synthetic data, and found that both our nonlinear sort NHTS and edge pruning algorithm ED significantly outperformed baselines. Future work includes extending the topological sorting algorithms to the full ANM setting with both linear and nonlinear functions, simultaneously exploiting both ancestor-descendent and parent-child relations, as well as adapting our approach to handle various forms of unmeasured confounding. Additionally, we aim to develop statistical guarantees of sample complexity for our methods, extending previous results [50] derived in the setting of nonlinear ANMs with Gaussian noise.

References

[1] Bühlmann, P., Peters, J., and Ernest, J. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 2014. URL http://arxiv.org/abs/1310.1533.

[2] Cheng, D., Li, J., Liu, L., Yu, K., Duy Le, T., and Liu, J. Toward Unique and Unbiased Causal Effect Estimation From Data With Hidden Variables. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. URL https://ieeexplore.ieee.org/document/9674196/.

[3] Chickering, D. M. Learning Equivalence Classes of Bayesian Network Structures. *Journal of Machine Learning Research*, 2002. URL https://arxiv.org/abs/1302.3566.

[4] Comon, P. Independent component analysis, a new concept? *Signal Processing*, 1994. URL https://hal.science/hal-00417283/document.

[5] Erdos, P. and Renyi, A. On the evolution of random graphs. *Publicationes Mathematicae*, 1960. URL https://publi.math.unideb.hu/load_doi.php?pdoi=10_5486_PMD_1959_6_3_4_12.

[6] Gan, K., Jia, S., and Li, A. Greedy approximation algorithms for active sequential hypothesis testing. *Advances in Neural Information Processing Systems*, 2021. URL https://dl.acm.org/doi/proceedings/10.5555/3540261.

[7] Glymour, C., Zhang, K., and Spirtes, P. Review of Causal Discovery Methods Based on Graphical Models. *Frontiers in Genetics*, 2019. URL https://www.frontiersin.org/article/10.3389/fgene.2019.00524/full.

[8] Gupta, S., Childers, D., and Lipton, Z. C. Local Causal Discovery for Estimating Causal Effects. In *Proceedings of the 2nd Conference on Causal Learning and Reasoning (CLeaR)*, 2023. URL http://arxiv.org/abs/2302.08070.

[9] Hoyer, P. O. and Hyttinen, A. Bayesian discovery of linear acyclic causal models. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009. URL https://arxiv.org/abs/1205.2641.

[10] Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems*, 2008. URL https://dl.acm.org/doi/10.5555/2981780.2981866.

[11] Hoyer, P. O., Shimizu, S., Kerminen, A. J., and Palviainen, M. Estimation of causal effects using linear non-Gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 2008. URL https://linkinghub.elsevier.com/retrieve/pii/S0888613X08000212.

[12] Lam, W.-Y., Andrews, B., and Ramsey, J. Greedy Relaxations of the Sparsest Permutation Algorithm. *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, 2022. URL https://proceedings.mlr.press/v180/lam22a/lam22a.pdf.

[13] Lee, J. J., Srinivasan, R., Ong, C. S., Alejo, D., Schena, S., Shpitser, I., Sussman, M., Whitman, G. J., and Malinsky, D. Causal determinants of postoperative length of stay in cardiac surgery using causal graphical learning. *The Journal of Thoracic and Cardiovascular Surgery*, 2022. URL https://linkinghub.elsevier.com/retrieve/pii/S002252232200900X.

[14] Li, A., Lee, J., Montagna, F., Trevino, C., and Ness, R. Dodiscover: Causal discovery algorithms in Python., 2022. URL https://github.com/py-why/dodiscover.

[15] Linder, J., Koplik, S. E., Kundaje, A., and Seelig, G. Deciphering the impact of genetic variation on human polyadenylation using aparent2. *Genome biology*, 2022. URL https://pubmed.ncbi.nlm.nih.gov/36335397/.

[16] Maasch, J., Pan, W., Gupta, S., Kuleshov, V., Gan, K., and Wang, F. Local discovery by partitioning: Polynomial-time causal discovery around exposure-outcome pairs. In *Proceedings of the 40th Conference on Uncertainy in Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2310.17816.

 https://doi.org/10.52202/079017-4231

[17] Marra, G. and Wood, S. Practical variable selection for generalized additive models. *Computational Statistics and Data Analysis*, 2023. URL https://www.sciencedirect.com/science/article/abs/pii/S0167947311000491.

[18] Marra, G. and Wood, S. Regression Shrinkage and Selection via the Lasso. *Computational Statistics and Data Analysis*, 2023. URL https://www.sciencedirect.com/science/article/abs/pii/S0167947311000491.

[19] Montagna, F., Mastakouri, A. A., Eulig, E., Noceti, N., Rosasco, L., Janzing, D., Aragam, B., and Locatello, F. Assumption violations in causal discovery and the robustness of score matching. *In 37th Conference on Neural Information Processing Systems*, 2023. URL http://arxiv.org/abs/2310.13387.

[20] Montagna, F., Noceti, N., Rosasco, L., Zhang, K., and Locatello, F. Causal Discovery with Score Matching on Additive Models with Arbitrary Noise. *Proceedings of the 2nd Conference on Causal Learning and Reasoning*, 2023. URL http://arxiv.org/abs/2304.03265.

[21] Montagna, F., Noceti, N., Rosasco, L., Zhang, K., and Locatello, F. Scalable Causal Discovery with Score Matching. *Proceedings of the 2nd Conference on Causal Learning and Reasoning*, 2023. URL http://arxiv.org/abs/2304.03382.

[22] Niu, W., Gao, Z., Song, L., and Li, L. Comprehensive Review and Empirical Evaluation of Causal Discovery Algorithms for Numerical Data, 2024. URL https://arxiv.org/abs/2407.13054.

[23] Pearl, J., Glymour, M., and Jewell, N. P. *Causal inference in statistics: a primer*. Wiley, 2016. URL https://www.datascienceassn.org/sites/default/files/CAUSAL%20INFERENCE%20IN%20STATISTICS.pdf.

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011. URL https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[25] Peters, J., Mooij, J., Janzing, D., and Schölkopf, B. Causal Discovery with Continuous Additive Noise Models. *Journal of Machine Learning Research*, 2014. URL http://arxiv.org/abs/1309.6779.

[26] Ramos-Carreno, C. and Torrecilla, J. dcor: Distance correlation and energy statistics in python. *SoftwareX*, 2023. URL https://www.sciencedirect.com/science/article/pii/S2352711023000225.

[27] Raskutti, G. and Uhler, C. Learning directed acyclic graphs based on sparsest permutations. *STAT*, 2013. URL https://arxiv.org/abs/1307.0366.

[28] Reisach, A., Seiler, C., and Weichwald, S. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2102.13647.

[29] Reisach, A. G., Tami, M., Seiler, C., Chambaz, A., and Weichwald, S. A Scale-Invariant Sorting Criterion to Find a Causal Order in Additive Noise Models. *37th Conference on Neural Information Processing Systems*, 2023. URL http://arxiv.org/abs/2303.18211.

[30] Rolland, P., Cevher, V., Kleindessner, M., Russel, C., Scholkopf, B., Janzing, D., and Locatello, F. Score Matching Enables Causal Discovery of Nonlinear Additive Noise Models. In *Proceedings of the 39th International Conference on Machine Learning*, 2022. URL https://arxiv.org/abs/2203.04413.

[31] Runge, J., Bathiany, S., Bollt, E., Camps-Valls, G., Coumou, D., Deyle, E., Glymour, C., Kretschmer, M., Mahecha, M. D., Muñoz-Marí, J., van Nes, E. H., Peters, J., Quax, R., Reichstein, M., Scheffer, M., Schölkopf, B., Spirtes, P., Sugihara, G., Sun, J., Zhang, K., and Zscheischler, J. Inferring causation from time series in Earth system sciences. *Nature Communications*, 2019. URL http://www.nature.com/articles/s41467-019-10105-3.

[32] Sanchez, P., Liu, X., O'Neil, A. Q., and Tsaftaris, S. A. Diffusion models for causal discovery via topological ordering. *Proceedings of the 39th International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2210.06201.

[33] Seabold, S. and Perktold, J. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010. URL https://www.statsmodels.org/stable/index.html.

[34] Shah, A., Shanmugam, K., and Ahuja, K. Finding Valid Adjustments under Non-ignorability with Minimal DAG Knowledge. *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022. URL http://arxiv.org/abs/2106.11560.

[35] Shah, A., Shanmugam, K., and Kocaoglu, M. Front-door adjustment beyond markov equivalence with limited graph knowledge. *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.11008.

[36] Shimizu, S. Lingam: Non-Gaussian Methods for Estimating Causal Structures. *Behaviormetrika*, 2014. URL http://link.springer.com/10.2333/bhmk.41.65.

[37] Shimizu, S., Hoyer, P. O., Hyvarinen, A., and Kerminen, A. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 2006. URL https://dl.acm.org/doi/10.5555/1248547.1248619.

[38] Shimizu, S., Inazumi, T., Sogawa, Y., Hyvarinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., Bollen, K., and Hoyer, P. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research*, 2011. URL https://dl.acm.org/doi/10.5555/1953048.2021040.

[39] Solus, L., Wang, Y., and Uhler, C. Consistency guarantees for greedy permutation-based causal inference algorithms. *Biometrika*, 2021. URL https://academic.oup.com/biomet/article-abstract/108/4/795/6062392?redirectedFrom=fulltext.

[40] Spirtes, P. An Anytime Algorithm for Causal Inference. *Proceedings of Machine Learning Research*, 2001. URL https://proceedings.mlr.press/r3/spirtes01a/spirtes01a.pdf.

[41] Spirtes, P. and Zhang, K. Causal discovery and inference: concepts and recent methodological advances. *Applied Informatics*, 2016. URL https://applied-informatics-j.springeropen.com/articles/10.1186/s40535-016-0018-x.

[42] Spirtes, P., Glymour, C., and Scheines, R. *Causation, Prediction, and Search*. Springer New York, 2000. URL http://link.springer.com/10.1007/978-1-4612-2748-9.

[43] Squires, C. Graphical Model Learning, 2021. URL https://github.com/uhlerlab/graphical_model_learning.

[44] Steeg, G. Non-parametric Entropy Estimation Toolbox, 2014. URL https://github.com/gregversteeg/NPEET.

[45] Takashi, I., Mayumi, I., Yan, Z., Takashi Nicholas, M., and Shohei, S. Python package for causal discovery based on lingam. *Journal of Machine Learning Research*, 2023. URL https://www.jmlr.org/papers/volume24/21-0321/21-0321.pdf.

[46] Wang, S. K., Nair, S., Li, R., Kraft, K., Pampari, A., Patel, A., Kang, J. B., Luong, C., Kundaje, A., and Chang, H. Y. Single-cell multiome of the human retina and deep learning nominate causal variants in complex eye diseases. *Cell genomics*, 2022. URL https://www.sciencedirect.com/science/article/pii/S2666979X22001069.

[47] Wong, A. K., Sealfon, R. S., Theesfeld, C. L., and Troyanskaya, O. G. Decoding disease: from genomes to networks to phenotypes. *Nature Reviews Genetics*, 2021. URL https://www.nature.com/articles/s41576-021-00389-x.

[48] Zhang, K. and Hyvarinen, A. On the Identifiability of the Post-Nonlinear Causal Model. *Uncertainty in Artificial Intelligence*, 2009. URL https://arxiv.org/pdf/1205.2599.

[49] Zheng, Y., Huang, B., Chen, W., Ramsey, J., Gong, M., Cai, R., Shimizu, S., Spirtes, P., and Zhang, K. Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, 2024. URL https://arxiv.org/abs/2307.16405.

[50] Zhu, Z., Locatello, F., and Cevher, V. Sample Complexity Bounds for Score-Matching: Causal Discovery and Generative Modeling, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0a3dc35a2391cabcb59a6b123544e3db-Paper-Conference.pdf.

Appendix

A Assumptions, Proofs, and Walkthrough for LHTS

A.1 Identifiability in Linear Setting

As an example [25]: suppose X, N are normally distributed, $X \perp\!\!\!\perp N$, and $Y = aX + N$. If $\bar{a} = \frac{aVar(X)}{a^2Var(X)+\sigma^2} \neq \frac{1}{a}$ and $\bar{N} = X - \bar{a}Y$, then $\bar{N} \perp\!\!\!\perp Y$ and the following DGP also fits the same distribution of X, Y, N: $X = \bar{a}Y + \bar{N}$. To generalize the intuition that non-Gaussianity could lead to identifiability, [37] use independent component analysis [4] to provide the following theorem:

Theorem A.1. *Assume a linear SEM with graph G, where $x_i = \sum_{k(j)<k(i)} b_{ij}x_j + \varepsilon_i, \forall j = 1, \ldots, d$, where all ε_j are jointly independent and non-Gaussian distributed. Additionally, for each $j \in \{1, \ldots, d\}$ we require $b_{ij} \neq 0$ for all $j \in Pa(x_i)$. Then, G is identifiable from the distribution.*

We assume that the identifiability conditions described in Theorem A.1 hold throughout Section 3.

A.2 Proof of Lemma 3.1

Lemma 3.1 (Active Causal Ancestral Path Relation Enumeration)**.** *Each pair of distinct nodes $x_i, x_j \in V$ can be in one of four possible active causal ancestral path relations: AP1) no active path exists between x_i, x_j; AP2) there exists an active backdoor path between x_i, x_j, but there is* no *active frontdoor path between them; AP3) there exists an active frontdoor path between x_i, x_j, but there is* no *active backdoor path between them; AP4) there exists an active backdoor path between x_i, x_j,* and *there exists an active frontdoor path between them.*

Proof of Lemma 3.1. For a pair of nodes x_i and x_j, either $x_i \perp\!\!\!\perp x_j$ or $x_i \not\perp\!\!\!\perp x_j$. If the former, then x_i, x_j are in AP1 relation. Suppose the latter is true: $x_i \not\perp\!\!\!\perp x_j$ implies $\exists$ at least one active path between x_i, x_j. Active paths can either be backdoor or frontdoor paths: therefore either a frontdoor path exists, a backdoor path exists, or both a frontdoor and backdoor path exist. Thus, x_i, x_j are either in AP2, AP3, or AP4 relation.

□

A.3 Proof of Lemma 3.2

Lemma 3.2. *The ancestral relationship between a pair of distinct nodes $x_i, x_j \in V$ can be expressed using active causal path relations: x_i, x_j are not ancestrally related if and only if they are in AP1 or AP2 relation; and x_i, x_j are ancestrally related if and only if they are in AP3 or in AP4 relation.*

Proof of Lemma 3.2. Note, x_i, x_j are ancestrally related if and only if there exists an active frontdoor path between them. The conclusion follows immediately. □

A.4 Proof of Lemma 3.3

Lemma 3.3 (AP1)**.** *Vertices x_i, x_j are in AP1 relation if and only if $x_i \perp\!\!\!\perp x_j$.*

Proof of Lemma 3.3. Note, $x_i \perp\!\!\!\perp x_j$ if and only if there does not exist an active causal path between them. The conclusion follows immediately. □

A.5 Proof of Lemma 3.4

Lemma 3.4 (AP2)**.** *Let M be the set of mutual ancestors between a pair of vertices x_i and x_j, i.e., $M = An(x_i) \cap An(x_j)$. Let x_i^M, x_j^M be the result of sequentially regressing all mutual ancestors in M out of x_i, x_j with univariate regressions, in any order. Then, let r_i^j be the residual of x_j^M regressed on x_i^M, and r_j^i be the residual of x_i^M regressed on x_j^M. Suppose $x_i \not\perp\!\!\!\perp x_j$. Then, x_i, x_j are in AP2 relation if and only if $r_i^j \perp\!\!\!\perp x_i^M$ and $r_j^i \perp\!\!\!\perp x_j^M$.*

https://doi.org/10.52202/079017-4231

Proof of Lemma 3.4. Under an identifiable LiNGAM, and given a linear topological ordering $k : V \mapsto \mathbb{R}$, x_i and x_j admit the following representation:

$$x_i = \sum_{k(m)<k(i)} \alpha_{im}\varepsilon_m + \varepsilon_i,$$
$$x_j = \sum_{k(m)<k(j)} \alpha_{jm}\varepsilon_m + \varepsilon_j,$$

where ε_m are jointly independent noise terms and non-Gaussian distributed, following from Theorem A.1. Additionally, for each $m : k(m) < k(i)$ we require $\alpha_{im} \neq 0$ for all $m \in Pa(x_i)$. Similarly for α_{jm}.

We first show the forward direction. Suppose x_i, x_j are in AP2 relation. Note, $x_i \not\perp\!\!\!\perp x_j$. Let $\overline{M}$ be the complement of M, i.e., $\overline{M} = V \setminus M$. Then, $\exists$ the following decomposition of x_i, x_j:

$$x_i = \sum_{x_m \in M} \alpha_{im}\varepsilon_m + \sum_{x_m \in \overline{M} \cap \text{An}(x_i)} \alpha_{im}\varepsilon_m + \varepsilon_i,$$
$$x_j = \sum_{x_m \in M} \alpha_{jm}\varepsilon_m + \sum_{x_m \in \overline{M} \cap \text{An}(x_j)} \alpha_{jm}\varepsilon_m + \varepsilon_i.$$

Then, the x_i^M, x_j^M have the general form

$$x_i^M = \sum_{x_m \in Y} \beta_{im}\varepsilon_m + \varepsilon_i,$$
$$x_j^M = \sum_{x_m \in Z} \beta_{jm}\varepsilon_m + \varepsilon_j,$$

where $Y \subseteq \overline{M} \cap \text{An}(x_i))$, $Z \subseteq \overline{M} \cap \text{An}(x_j))$. Note, $Y \cap Z = \emptyset$. As ε_i are all mutually independent, this implies that $r_i^j \perp\!\!\!\perp r_j^i$.

We now show the reverse direction. Suppose $x_i \not\perp\!\!\!\perp x_j$, $r_i^j \perp\!\!\!\perp x_i^M$, and $r_j^i \perp\!\!\!\perp x_j^M$. Note that $x_i \not\perp\!\!\!\perp x_j \implies x_i, x_j$ are not in AP1 relation. Suppose for contradiction that x_i, x_j are in AP3 or AP4 relation, where $x_i \in \text{An}(x_j)$ (WLOG). Then $\exists$ a frontdoor path between x_i, x_j, WLOG $x_i \in \text{An}(x_j)$. Regressing the descendent on the ancestor will always result in a dependent residual, even after projecting out the influence of the mutual ancestors. Therefore, $x_j^M \not\perp\!\!\!\perp r_j^i$, leading to a contradiction. Therefore, x_i, x_j must be in AP2 relation. □

A.6 Proof of Lemma 3.5

Lemma 3.5 (AP3)**.** *Let r_i^j be the residual of the x_j regressed on x_i, and r_j^i be the residual of x_i regressed on x_j. Vertices x_i, x_j are in AP3 relation if and only if $x_i \not\perp\!\!\!\perp x_j$ and one of the following holds: 1) $x_i \perp\!\!\!\perp r_i^j$ and $x_j \not\perp\!\!\!\perp r_j^i$, corresponding to $x_i \in An(x_j)$, or 2) $x_i \not\perp\!\!\!\perp r_i^j$ and $x_j \perp\!\!\!\perp r_j^i$, corresponding to $x_j \in An(x_i)$.*

Proof of Lemma 3.5. We first show the forwards direction. Suppose x_i, x_j are in $AP3$ relation. Note an active path exists between x_i, x_j, so $x_i \not\perp\!\!\!\perp x_j$. WLOG, let $x_i \in \text{An}(x_j)$. Then, x_i, x_j admit the following representation:

$$x_i = x_i$$
$$x_j = b_{ji}x_i + \sum_{x_m \in \text{An}(x_j) \setminus x_i} \alpha_{jm}x_m + \varepsilon_j.$$

Note, as there does not exist a backdoor path between x_i, x_j, we have $x_i \perp\!\!\!\perp x_m \forall x_m \in \text{An}(x_j) \setminus x_i$. Therefore, $r_i^j \perp\!\!\!\perp x_i$. Note that $x_j \in \text{De}(x_i)$: pairwise regression will leave $r_j^i \not\perp\!\!\!\perp x_j$ as this an example of reverse causality, a special case of endogeneity [23].

We now show the reverse direction. WLOG, suppose x_i, x_j satisfy condition 1) in Lemma 3.5. As $x_i \not\perp\!\!\!\perp x_j$, they cannot be in AP1 relation. Suppose for contradiction that they are in AP2 or AP4 relation. Then, $\exists$ a backdoor path between x_i, x_j, and a confounding x_z on that backdoor path. As x_z confounds x_i, x_j and is not adjusted for in a pairwise regression, we have $x_i \not\perp\!\!\!\perp r_i^j, x_j \not\perp\!\!\!\perp x_j^i$, a contradiction. Therefore, x_i, x_j must be in AP2 relation. □

A.7 Proof of Lemma 3.6

Lemma 3.6 (AP4)**.** *Let M, $r_i^j, r_j^i, x_i^M, x_j^M$ be as defined in Lemma 3.4. Suppose x_i, x_j not in AP3 relation. Then, x_i, x_j are in AP4 relation if and only if $x_i \not\perp\!\!\!\perp x_j$ and one of the following holds: $r_i^j \perp\!\!\!\perp x_i^M$ and $r_j^i \not\perp\!\!\!\perp x_j^M$ corresponding to $x_i \in An(x_j)$, or 2) $r_i^j \not\perp\!\!\!\perp x_i^M$ and $r_j^i \perp\!\!\!\perp x_j^M$ corresponding to $x_j \in An(x_i)$.*

Proof of Lemma 3.6. We first show the forward direction. Suppose x_i, x_j are in AP4 relation. Note that an active path exists between x_i, x_j, so $x_i \not\perp\!\!\!\perp x_j$. WLOG, suppose $x_i, \in \text{An}(x_j)$. Note, x_i^M, x_j^M are the result of projecting mutual ancestors M out of x_i, x_j. Therefore, x_i^M, x_j^M admit the following representation:

$$x_i^M = \sum_{x_m \in \overline{M} \cap \text{An}(x_i)} \alpha_{im}\varepsilon_m + \varepsilon_i$$

$$x_j^M = \alpha_{ji}\varepsilon_i + \sum_{x_m \in \overline{M} \cap \text{An}(x_j)} \alpha_{jm}\varepsilon_m + \varepsilon_j$$

Note that $\overline{M} \cap \text{An}(x_i) \cap (\overline{M} \cap \text{An}(x_j) \cup x_j) = \emptyset$. Therefore, $r_i^j \perp\!\!\!\perp x_i^M$. Note that $\varepsilon_i \in$ both x_i^M, x_j^M, therefore $r_j^i \not\perp\!\!\!\perp x_j^M$.

We now show the reverse direction. Suppose condition 1) holds, i.e., $x_i \not\perp\!\!\!\perp x_j$ and WLOG $r_i^j \perp\!\!\!\perp x_i^M, r_j^i \not\perp\!\!\!\perp x_j^M$. Suppose for contradiction that x_i, x_j not in AP4 relation. They cannot be in AP1 relation because $x_i \not\perp\!\!\!\perp x_j$. They cannot be in AP2 relation because $x_j^M \not\perp\!\!\!\perp r_j^i$. By assumption they are not in AP3 relation. Therefore, we have a contradiction, therefore they are in AP4 relation. □

A.8 Ancestor Sort

Algorithm 4 AS: Ancestor Sort

input: set of ancestral relations ARS.
initialize: hierarchical topological order O [], remaining variables RV $[x_1, \ldots, x_d]$, layer L [].
while RV is not empty **do**
Stage 1: Determine Roots
    for $x_i \in RV$ **do**
        if x_i has no ancestors $\in$ RV **then**
            Append x_i to L.
Stage 2: Update Sort and Remaining Variables
    Append L to O.
    Remove all vertices in L from RV.
return O

Overview In each iteration, this algorithm uses the set of ancestral relations to identify which vertices have no ancestors amongst the vertices that have yet to be sorted. It peels those nodes off, adding them as a layer to the hierarchical topological sort. First the roots are peeled off, then the next layer, and so on.

Proof of Correctness

Proof. The input to the algorithm is an ancestor table ARS cataloging all ancestral relations for every pair of nodes in an unknown DAG G. Let H be the minimum number of layers ($H - 1$ is the length of the longest directed path in the graph) necessary in a valid hierarchical topological sort of G.

https://doi.org/10.52202/079017-4231

Base Cases (1,2)

1. $H = 1$. None of the nodes have ancestors, so all nodes are added to layer 1 in Stage 1.
2. $H = 2$. Nodes with no ancestors are added to layer 1 in Stage 1, and then removed from the remaining variables in Stage 2. As $H = 2$, the longest directed path is 1, so all nodes in the remaining variables must have no ancestors in the remaining variables (otherwise, this would make the longest directed path 2). The nodes are added to layer 2 in Stage 1, and removed from the remaining variables in Stage 2. Remaining variables is empty, so now the order is correctly returned.

Inductive Assumption For any graph G with $H < k$, Hierarchical Topological Sort will return a minimal hierarchical topological ordering.

We now show that hierarchical topological sort now yields a minimal hierarchical topological ordering for G where $H = k$.

1. In the first iteration of Stage 1, nodes with no ancestors will be added to layer 1, then removed from the remaining variables in Stage 2.
2. At this point, note that we can consider the induced subgraph formed by the nodes left in remaining, G'. The minimal hierarchical topological ordering that represents G' must have $k-1$ layers. It cannot be greater than $k-1$, because G' is a subgraph of G and we removed nodes in layer 1 of, reducing the maximal path length by 1. It cannot be less than $k-1$, because that would imply that k was not the minimal number of layers needed to represent G.
3. By Inductive Assumption, Hierarchical Topological sort will return the a correct minimal hierarchical topological ordering for the induced subgraph G'.
4. Appending the layer 1 and the sort produced by G' yields the full hierarchical topological sort for G.

Inductive assumption is satisfied for $H = k$, so for a graph G with arbitrary H, the algorithm recovers the correct hierarchical topological sort.

□

A.9 Proof of Correctness for Linear Hierarchical Topological Sort

Theorem A.2. *3.7 Given a graph G, Algorithm 1 asymptotically finds a correct hierarchical sort of G.*

Proof. The goal is to show that all ancestral relations between distinct pair of nodes $x_i, x_j \in V$ determined by LHTS in Stages 1, 2, and 3 are correct. By A.8, Stage 4 will return a valid hierarchical topological sort given fully identified ancestral relations.

Stage 1 identifies x_i, x_j as in AP1 relaton if and only if $x_i \perp\!\!\!\perp x_j$: it follows by Lemma 3.3 that vertices in AP1 relation are correctly identified, and vertices in AP2, AP3 and AP4 relation are not identified.

Stage 2 identifies x_i, x_j unidentified in Stage 1 as in AP3 relation if and only if after pairwise regression, one of the residuals is dependent, and the other is independent: it follows by Lemma 3.5 vertices in AP3 relation are correctly identified, and vertices in AP2 or AP4 relation are not identified.

Consider a hierarchical topological sort of DAG G π_H, with h layers. Note, Layer 1 of π_H consists of root nodes. Ancestral relations between root nodes and all other nodes are of type AP1 and AP3, and were discovered in Stage 1 and Stage 2.

We induct on layers to show that Stage 3 recovers all ancestral relations of type AP2 and AP4, one layer at a time in each iteration.

Base Iterations (1,2)

1. All ancestral relationships are known for nodes in layer 1 of π_H, therefore all mutual ancestors of layer 2 and every lower layer are known. Then, by Lemma 3.4 and Lemma 3.6 ancestral relations of type AP2 and AP3 between vertices in layer 2 and all lower layers are discovered in iteration 1.

2. All ancestral relationships are known for nodes in layer 1 and 2 of π_H, therefore all mutual ancestors of layer 3 and every lower layer are known. Then, by Lemma 3.4 and Lemma 3.6 ancestral relations of type AP2 and AP3 between vertices in layer 2 and all lower layers are discovered in iteration 2.

Iteration $k-1$, Inductive Assumption We have recovered all ancestral relationships of nodes in layers 0 to $k-1$.

1. As ancestral relationships of nodes in layers 0 to $k-1$ are known, mutual ancestors of vertices in layer k and every lower layer are known, so by Lemma 3.4 and 3.4 all ancestral relations of type AP2 and AP4 between vertices in layer k and every lower layer are discovered.

Iteration $k-1$ Inductive Assumption is satisfied for iteration k, therefore we recover all ancestral relations of type AP2 and AP4 and 4) for nodes in layers 0 to k. Thus, for a DAG with an arbitrary number of layers, Stage 3 recovers all ancestral relations of type AP2 and AP4.

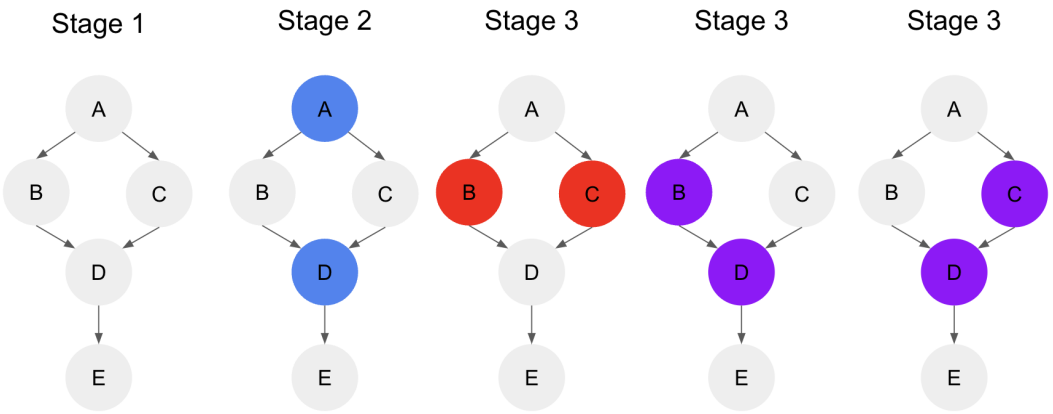
□

A.10 Time Complexity Proof for Linear Hierarchical Topological Sort

Proof. In Stage 1, LHTS runs $O(d^2)$ marginal independence tests that each have $O(n^2)$ complexity. In each step for Stage 2 and Stage 3, LHTS runs $O(d^2)$ marginal independence tests each with $O(n^2)$ complexity. In the worse case of a fully connected DAG, there are d^2 steps in total, across Stage 2 and Stage 3: this is because in each step one layer of the layered sort DAG is identified, and a fully connected DAG has d layers. Therefore, the overall sample complexity of LHTS is $O(d^3n^2)$. □

A.11 Walk-through of LHTS

The following diagram illustrates each stage of LHTS on an exemplary 5-node DAG:



All vertices are dependent on each other, so no AP1 relations are discovered in Stage 1. In Stage 2, vertex A is discovered in AP2 relation to all vertices, and vertex D is discovered in AP2 relation to vertex E. In Stage 3, vertices B and C are discovered to be in AP3 relation to each other; then, we discover vertex B in AP4 relation to vertex D, and vertex C in AP4 relation to vertex D. Therefore, in Stage 4, we recover the topological sort after running the AS subroutine.

https://doi.org/10.52202/079017-4231

A.12 Proof of Lemma 3.9: Root Active Causal Ancestral Path Relations

Lemma 3.9. *A vertex is a root vertex if and only if it has AP1 or AP3 relations with all other vertices.*

Proof of Lemma 3.9. We first show the forward direction. If x_i is a root, then $\text{De}(x_i) = \emptyset$. Therefore, it cannot have a backdoor path between it and any other vertex, therefore it must be in either $AP1$ or $AP3$ relation with other vertices.

We now show the reverse direction. If x_i is in AP1 or AP3 (as an ancestor) relations with all other nodes, it cannot have any parents. Therefore, x_i is a root.

□

B Assumptions, Proofs, and Walkthrough for NHTS

B.1 Identifiability in Nonlinear Setting

Following the style of [19], we first observe that the following condition guarantees that the observed distribution of a pair of variables x_i, x_j can only be generated by a unique ANM:

Condition B.1 (Hoyer & Hyttinen [9]). *Given a bivariate model $x_i = \varepsilon_i, x_j = f_j(x_i) + \varepsilon_j$ generated according to (1), we call the SEM an identifiable bivariate ANM if the triple $(f_i, p_{\varepsilon_i}, p_{\varepsilon_j})$ does not solve the differential equation $k''' = k''(-\frac{g'''f'}{g''} + \frac{f''}{f'}) - 2g''f''f' + g'f''' + \frac{g'g'''f''f'}{g''} - \frac{g'(f'')^2}{f'}$ for all x_i, x_j such that $f'(x_i)g''(x_j - f_j(x_i)) \neq 0$, where $p_{\varepsilon_i}, p_{\varepsilon_j}$ are the density of $\varepsilon_i, \varepsilon_j$, $f = f_j, k = \log p_{\varepsilon_i}, g = p_{\varepsilon_j}$. The arguments $x_j - f_j(x_i), x_i$ and x_i of g, k and f respectively, are removed for readability.*

There is a generalization of this condition to the multivariate nonlinear ANM proved by [25]:

Theorem B.2. *(Peters et al. [25]). An ANM corresponding to DAG G is identifiable if $\forall x_j \in V, x_i \in Pa(x_j)$ and all sets $S \subseteq V$ with $Pa(x_j) \setminus \{i\} \subseteq S \subseteq \overline{De(j)} \setminus \{x_i, x_j\}$, $\exists\ X_S$ with positive joint density such that the triple $\left(f_j(Pa(j) \setminus \{x_i\}, x_i), p_{x_i|X_s}, p_{\varepsilon_j}\right)$ satisfies Condition B.1, and f_j are non-constant in all arguments.*

We assume that the identifiability conditions described in Theorem B.2 hold throughout Section 4.

B.2 Proof of Lemma 4.1

Lemma 4.1 (Active Causal Parental Path Relation Enumeration). *Let $x_i, x_j \in V$ be a pair of distinct nodes, where x_i is one of the potential parents of x_j. Let $C = PA(x_j) \setminus x_i$. There are in total four possible types of active causal parental path relations between x_i and x_j with respect to C: PP1) x_i and x_j are* not *directly causally related, and* no *active path exists between x_i and C; PP2) x_i directly causes x_j ($x_i \to x_j$), and* no *active path exists between x_i and C; PP3) x_i directly causes x_j ($x_i \to x_j$), and there exists an active path between x_i and C; PP4) x_i and x_j are* not *directly causally related, and there exists an active path between x_i and C.*

Proof. Either $x_i \perp\!\!\!\perp x_j$ or $x_i \not\perp\!\!\!\perp x_j$: if the former, then x_i, x_j are in PP1 relation. Suppose the latter is true. Then, either $x_i \perp\!\!\!\perp C$ or $x_i \not\perp\!\!\!\perp C$: if the former, then x_i, x_j are in PP2 relation. Suppose the latter is true. Then, either $x_i \in \text{PA}(x_j)$ or $x_i \notin \text{PA}(x_j)$. If the former, then x_i, x_j are in PP3 relation, and if the latter, then x_i, x_j are in PP4 relation. □

B.3 Proof of Lemma 4.2

Lemma 4.2 (Non-PP1). *Vertices $x_i, x_j \in V$ are not in PP1 relation if and only if $x_i \not\perp\!\!\!\perp x_j$.*

Proof. We first show the forward direction. If x_i, x_j are not in PP1 relation, then either $x_i \in \text{Pa}(x_j)$ or there exists an active causal path between x_i, x_j; therefore, $x_i \not\perp\!\!\!\perp x_j$.

We now show the reverse direction. If $x_i \not\perp\!\!\!\perp x_j$ there exists an active causal path between x_i, x_j, therefore they cannot be in PP1 relation. □

B.4 Proof of Lemma 4.3

Lemma 4.3 (PP2). *Let $x_i, x_j \in V$, $P_{ij} = \{x_k \in V : x_k \perp\!\!\!\perp x_i, x_k \not\perp\!\!\!\perp x_j\}$, r_i^j be the residual of x_j nonparametrically regressed on x_i, and $r_{i,P}^j$ be the residual of x_j nonparametrically regressed on x_i and all $x_k \in P_{ij}$. Suppose x_i and x_j are not in PP1 relation. Then, x_i and x_j are in PP2 relation if and only if one of the following holds: 1) $x_i \perp\!\!\!\perp r_i^j$ or 2) $x_i \perp\!\!\!\perp r_{i,P}^j$.*

Proof. Note, there exists two sub cases of PP2 relation: PP2a) x_i is the only parent of x_j, i.e. $|C| = 0$ and PP2b) x_i is not the only parent of x_j, i.e., $|C| > 0$.

https://doi.org/10.52202/079017-4231

We first show the forward direction. Suppose x_i, x_j are in PP2a relation. Then, as the

$$x_j = f_j(x_i) + \varepsilon_j$$

$x_i \perp\!\!\!\perp r_i^j$. Suppose that x_i, x_j are in a PP2b relation. Consider P_{ij}: note that $x_i \perp\!\!\!\perp C$, and $x_j \not\perp\!\!\!\perp C$, $C \subseteq P_{ij}$. Note that $\text{De}(x_j) \not\perp\!\!\!\perp x_i$, therefore $\text{De}(x_j) \notin P_{ij}$. Therefore, P_{ij} contains all parents of x_j, and excludes all descendent of x_j. Then, as

$$x_j = f_j(x_i, C) + \varepsilon_j$$

we have $x_i \perp\!\!\!\perp r_{j_P}^i$.

We now show the reverse direction. Suppose $x_i \perp\!\!\!\perp r_i^j$. Note by assumption x_i, x_j are not in PP1 relation. Suppose for contradiction that x_i, x_j are in PP3 or PP4 relation. Then, there must exist at least one active path between x_i, x_j that goes through C. If $\exists$ a backdoor path from x_i, x_j, then at least one vertex in C is a confounder of x_i, x_j, and thus $x_i \not\perp\!\!\!\perp r_i^j$; therefore, there must exist a frontdoor path between x_i, x_j through C, with at least one mediator x_k. Note that x_k is dependent on both x_i and x_j, and therefore is excluded from P_{ij}. The exclusion of x_k introduces omitted variable bias [23], and thus $x_i \not\perp\!\!\!\perp r_{i_P}^j$, leading to contradiction. Thus, x_i, x_j are in PP2 relation.

□

B.5 Proof of Lemma 4.4

Lemma 4.4 (Roots)**.** *All non-isolated root vertices are contained in W. In addition, $x_i \in W$ is a root vertex if and only if 1) x_i is not a known descendant of any $x_j \in W$, and 2) for each $x_j \in W$, either a) $x_i \perp\!\!\!\perp x_j$, b) x_j is in PP2 relation to x_i, i.e., $x_j \in Ch(x_i)$, or c) there exists a child of x_i, denoted by x_k, that cannot be d-separated from x_j given x_i, i.e., $x_j \not\perp\!\!\!\perp x_k | x_i$.*

Proof. We first show that all non-isolated root vertices are contained in W. Note, if a root x_i is not isolated, it has at least one child x_j. By definition, x_i has no parents, so there cannot exist a backdoor path between x_i, x_j. Consider $x_j \in \text{Ch}(x_i)$ that is in a hierarchical layer closest to x_j. Note that there cannot exist an active path from x_i to $\text{Pa}(x_j) \setminus x_i$, because that would imply that x_j is not the child of x_i that is in the closest hierarchical layer. Therefore, x_i, x_j must be in PP2 relation, and $x_i \in W$.

We now show the forward direction. Suppose x_i is a root vertex. Then, x_i has no parents, so it cannot be the descendent of any vertex, let alone any vertex in W: Condition 1) is satisfied. For each $x_j \in W$, there either exists an active path between x_i, x_j, or there does not. If there does not exist an active path, then $x_i \perp\!\!\!\perp x_j$, satisfying Condition a). If there does exist an active path, then as x_i is a root, $x_j \in \text{De}(x_i)$: the active path must be a frontdoor path. Consider the vertex $x_k \in \text{Ch}(x_j)$ that is in a hierarchical layer closest to x_j, AND is along the active frontdoor path from x_i to x_j. Note that x_i is in PP2 relation with x_k. If $x_k = x_j$, then Condition b) is satisfied. If $x_k \neq x_j$, then we note that there is an active frontdoor path between x_k and x_j. Therefore, x_k, x_j are dependent even after conditioning on the mutual ancestor x_i, i.e., $x_j \not\perp\!\!\!\perp x_k | x_i$: Condition c) is satisfied. Therefore, Condition 2) is always satisfied.

We now show the reverse direction. Suppose Condition 1) and Condition 2) are satisfied for $x_j \in W$. Suppose for contradiction that x_j is not a root. Note that $\exists x_i \in W$ such that x_i is a root and $x_j \in \text{De}(x_i)$. Consider any vertex $x_k \in \text{Ch}(x_j)$ such that x_j is in PP2 relation to x_k. Note that all active paths between x_i and x_k must be frontdoor paths, and x_j must lie along all these active frontdoor paths: otherwise, this would contradict that x_j and x_k are in PP2 relation. Therefore, $x_i \perp\!\!\!\perp x_k | x_j$. This contradicts Condition 2c), therefore Condition 1) and Condition 2) cannot hold true for $x_i \in W$ that is not a root.

□

B.6 Proof of Correctness for Nonlinear Hierarchical Topological Sort

Theorem 4.5. *Given a graph G, Algorithm 2 asymptotically finds a correct hierarchical sort of G.*

Proof. This proof can be broken into two parts: we first show 1) all root vertices are identified after Stage 3, then we show 2) that Stage 4 recovers a correct hierarchical topological sort given the root vertices.

Note that Stage 1 identifies x_i, x_j as not in PP1 relation if and only if $x_i \not\perp\!\!\!\perp x_j$: it follows by Lemma 4.2 that these identifications are correct.

Note that Stage 2 identifies the set $x_i \in W$ if and only if $\exists x_j$ such that either 1) $x_i \perp\!\!\!\perp r_i^j$ or 2) $x_i \perp\!\!\!\perp r_{i,P}^j$: it follows by Lemma 4.3 that these identifications are correct.

Note that Stage 3 explicitly uses a condition from Lemma 4.4 to find all non-isolated root vertices - it is therefore correct. Note that all isolated root vertices were identified in Stage 1. Therefore, all roots are identified.

Consider a hierarchical sort of DAG G π_L: suppose the maximal directed path in G has size h. Note that layer 0 of π_L consists of root nodes, and therefore has been identified. We note the following **observation**: when x_i is nonparametrically regressed on all $x_k \in \pi_L$ to obtain residual r_i, if $\exists$ an ancestor of x_i not in π_L, then r_i will be dependent on at least one $x_i \in \pi_L$: due to omitted variable bias [23].

We now induct on the layers of π_L to show that Stage 4 correctly recovers all layers.

Base Iterations (1,2)

1. All roots are identified, so layer 0 of π_L is identified.
2. By **observation**, only x_i in layer 1 of π_L will have independent residuals r_i after nonparametric regression on $x_i \in \pi_L$, so layer 1 is correctly identified.

Iteration $k-1$, Inductive Assumption We have recovered all layers of π_L up to layer $k-1$.

1. By **observation**, only x_i in layer k of π_L will have independent residuals r_i after nonparametric regression on $x_i \in \pi_L$, so layer k is correctly identified.

Iteration $k-1$ Inductive Assumption is satisfied for iteration k, therefore we recover all layers of π_L from 0 to k. Thus, for a DAG with an arbitrary number of layers, Stage 4 recovers the full hierarchical sort. □

B.7 Time Complexity for NHTS

Theorem 4.6. *Given n samples of d vertices generated by a identifiable nonlinear ANM, the worst case runtime complexity of Algorithm 2 is upper bounded by $O(d^3n^3)$.*

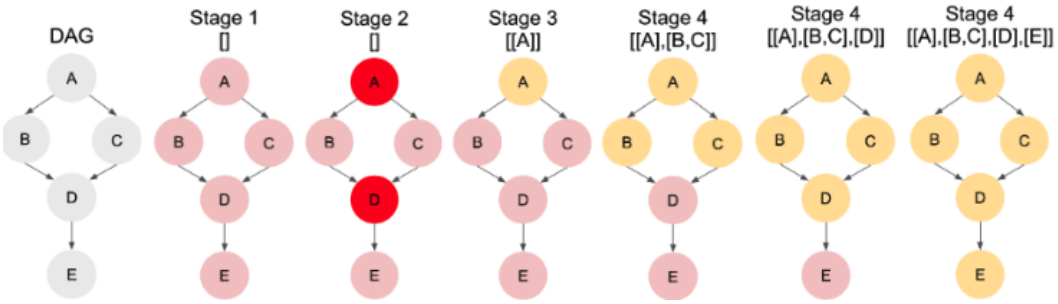
Proof. In Stage 1, NHTS runs $O(d^2)$ marginal independence tests that each have $O(n^3)$ complexity. In Stage 2, NHTS runs $O(d^2)$ nonparametric regressions and $O(d^2)$ marginal independence tests, each of which have $O(n^3)$ complexity. In Stage 3 NHTS runs at most $O(d^2)$ conditional independence tests, each of which has $O(n^3)$ complexity. In the worst case of a fully connected DAG, NHTS goes through $O(d)$ steps in Stage 4: in each step of Stage 4, NHTS runs $O(d)$ nonparametric regressions and $O(d^2)$ marginal independence tests, each of which has $O(n^3)$ complexity. Therefore, the overall sample complexity of NHTS is $O(d^3n^3)$. □

B.8 Walk-through of NHTS

The following diagram illustrates each stage of NHTS on an exemplary 5-node DAG:

In Stage 1, we discover that none of the vertices are in PP1 relations. In Stage 2, we discover that vertex A is in PP2 relation to vertices B and C, and vertex D is in PP2 relation to vertex E: therefore, A and D are our potential roots. In Stage 3, we find that vertex D d-separates its child, vertex E, from vertex A: therefore A is the root vertex. In Stage 4, we regress the unsorted vertices onto A, finding that B,C are independent of the residual in the first round, then D, then E in the last round.

https://doi.org/10.52202/079017-4231



B.9 Time Complexity of NHTS vs RESIT, NoGAM

Theorem 4.7. *Consider a fully connected DAG $G = (V, E)$ with nonlinear ANM. Let $d := |V|$. Let n_k^{NHTS} be the number of nonparametric regressions with covariate set size $k \in [d-2]$ run by NHTS when sorting V; we similarly define n_k^{RESIT} and n_k^{NoGAM} respectively. Then, $n_k^{\mathrm{NHTS}} = d - k$, and $n_k^{\mathrm{RESIT}} = n_k^{\mathrm{NoGAM}} = k + 1$. This implies that for all $k > \frac{d}{2}$, $n_k^{\mathrm{NHTS}} < n_k^{\mathrm{RESIT}} = n_k^{\mathrm{NoGAM}}$.*

Proof. RESIT and NoGAM both identify leaf vertices in an iterative fashion, regressing each unsorted vertex on the rest of the unsorted vertices; RESIT tests the residual for independence with the covariate set while NoGAM uses the residual for score matching. Therefore, the number of regressions run in both methods in each step equals one plus the covariate set size. Therefore, when the covariate set size is $k > \frac{d}{2}$, there are $k + 1$ regressions run.

In the case of a fully directed graph, the first stage of NHTS only runs pairwise regressions with empty conditioning sets. After the first stage, NHTS regresses each unsorted vertex onto all sorted vertices, finding vertices with independent residuals. Therefore, the number of regressions run is equal to d minus the size of the covariate set. Therefore, when the covariate set size is $k > \frac{d}{2}$, there are $d - k$ regressions run. □

C Proofs and Walkthrough for ED

C.1 Proof of Lemma

Lemma 5.1 (Parent Discovery)**.** *Let π be a topological ordering, x_i, x_j such that $\pi(x_i) < \pi(x_j)$. Let $Z_{ij} = C_{ij} \cup M_{ij}$, where $C_{ij} = \{x_k : x_k \in Pa(x_i), x_k \not\perp\!\!\!\perp x_j\}, M_{ij} = \{x_k : x_k \in Pa(x_j), \pi(x_i) < \pi(x_k) < \pi(x_j)\}$. Then, $x_i \to x_j \iff x_i \not\perp\!\!\!\perp x_j | Z_{ij}$.*

Proof. Note, $C_{i,j}$ blocks all backdoor paths between x_i, x_j and $M_{i,j}$ blocks all frontdoor paths between x_i, x_j, except the direct edge.

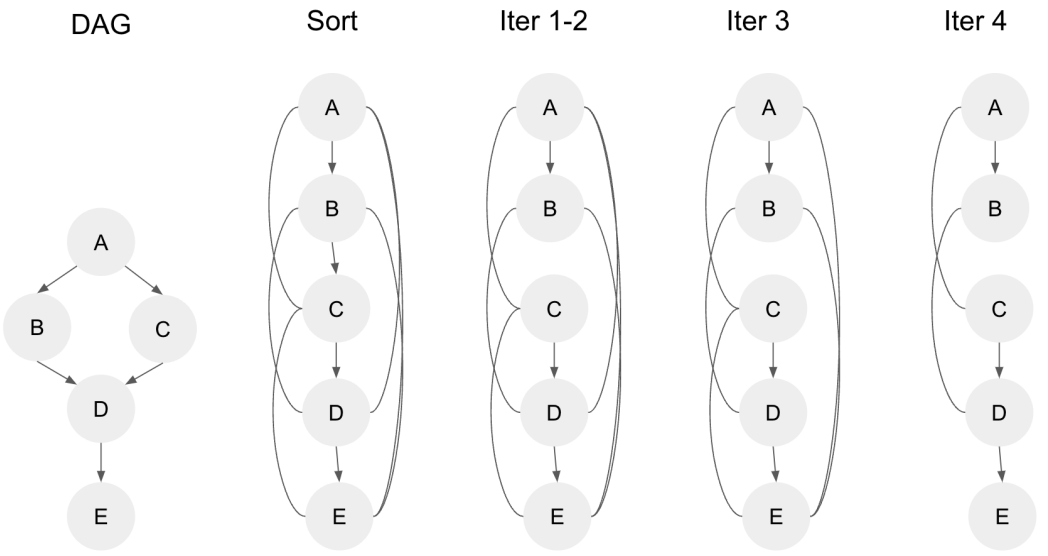
We first show the forward direction. If $x_i \not\perp\!\!\!\perp x_j | Z_{i,j}$, there must be an direct edge between x_i, x_j, and as $k(i) < k(j)$, we have $x_i \to x_j$.

We now show the reverse direction. If $x_i \to x_j$, then there does not exist a conditioning set that makes $x_i \perp\!\!\!\perp x_j$, which implies $x_i \not\perp\!\!\!\perp x_j | Z_{i,j}$. □

C.2 Hierarchical Version of Edge Discovery

The current implementation of Edge Discovery takes a linear hierarchical topological sort as input; this is equivalent to a hierarchical topological sort where every layer contains only one vertex. To generalize ED to a hierarchical sort, we simply adjust how the algorithm loops over the sort, and which vertices are included in which conditioning sets C_{ij}, M_{ij}. We give an example of how the latter would change: suppose the algorithm is checking whether $x_i \in \text{Pa}(x_j)$ where x_i is in layer 2 ($\pi_L(x_i) = 2$) and x_j is layer 6 ($\pi_L(x_j) = 6$). M_{ij} would be equal to the vertices who are parents of x_j that are in layers that are between layer 2 and 6: i.e. M_{ij} equals $x_k \in \text{Pa}(x_j)$ such that $2 < \pi_L(x_k) < 6$. C_{ij}, would stay the same, just being equal to $\text{Pa}(x_i)$. Now, to generalize the looping part, notice that the linear version of ED loops over indices of the linear topological sort. The hierarchical version of Edge Discovery would loop over and within layers of the topological sort. We give an example of this: suppose the algorithm has found all parents of vertices in layer 4 or lower: the next step would be to find the parents of vertices in layer 5. The algorithm sets any vertex in layer 5 as x_j, and first finds all parents of x_j in the immediately preceding layer, layer 4. The algorithm then finds all parents of x_j in the layer preceding layer 4, layer 3. This is essentially the same approach as the linear version, except the size of the conditioning sets and number of conditional tests run is far lower, as the layers provide extra knowledge, limiting which variables can be parents or confounders of other variables.

C.3 Edge Discovery



We first provide a walk-through of ED on an exemplary 5-node DAG:

https://doi.org/10.52202/079017-4231

We first use a topological ordering algorithm to obtain a sort from the DAG: the sort corresponds to a fully connected DAG, from which we will prune spurious edges. In the first two iterations we find that $A \to B$, but $B \not\to C$, as $B \perp\!\!\!\perp C|A$. In the third iteration we find that $B \to D, C \to D$, but $A \not\to D$, as $A \perp\!\!\!\perp D|B, C$. In the final iteration, we find that $D \to E$, but $A \not\to E, B \not\to E, C \not\to E$, given $A \perp\!\!\!\perp E|D, B \perp\!\!\!\perp E|D, C \perp\!\!\!\perp E|D$. Therefore, we recover the correct set causal edges, removing all spurious edges. We now present a statement of correctness and proof for ED:

Theorem 5.2. *Given a correct topological ordering of G, Algorithm 3 asymptotically finds correct parent sets PA*$(x_i), \forall x_i \in G$.

Proof of Correctness

Background We are given a linear topological sort $\pi(V) = [x_1, x_2, \ldots, x_d]$ (where x_1 has no parents). For x_a that appears before x_b, let $Z_{a,b} = C_{a,b} \cup M_{a,b}$, where $C_{a,b}$ is the set of potential confounders of x_a, x_b, and $M_{a,b}$ is the set of known mediators of x_a, x_b (defined precisely in Lemma 5.1). Let $x \perp\!\!\!\perp y|\cdot$ be the value of a conditional independence test between x and y.

Proof.

Base Case Iterations (1,2,3,4)

1. Finding parents of x_1: by the topological sort, x_1 has no parents.

2. Finding parents of x_2: As x_1 is the only possible parent of x_2, there are no possible confounders or mediators between x_1 and x_2, so we initialize $Z_{1,2} = \emptyset$, then by $L3$ $x_1 \not\perp\!\!\!\perp x_2|Z_{1,2} = x_1 \not\perp\!\!\!\perp x_2 \iff x_1 \to x_2$. Thus, we recover all possible edges between $[x_1, x_2]$.

3. Finding parents of x_3: first we check whether $x_2 \to x_3$, then we check whether $x_1 \to x_3$. From iteration 2, we know all edges between $[x_1, x_2]$.

   Case 1: if $x_1 \to x_2$, then it is a possible confounder. Therefore, we initialize $C_{2,3} = x_1$. By topological sort there is no mediator between x_2 and x_3, therefore we initialize $M_{2,3} = \emptyset$. Then by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_3|Z_{2,3} = x_2 \not\perp\!\!\!\perp x_3|x_1 \iff x_2 \to x_3$. If an edge between x_2 and x_3 exists, initialize $M_{1,3}$ accordingly. Note, x_1 has no parents, so we initialize $C_{1,3} = \emptyset$. Then, by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_3|Z_{1,3} \iff x_1 \to x_3$. We recover all possible edges between $[x_1, x_2, x_3]$.

   Case 2: if $x_1 \not\to x_2$, there are no possible confounders between x_2 and x_3. Therefore, we initialize $C_{2,3} = \emptyset$. By topological sort there is no mediator between x_2 and x_3, therefore we initialize $M_{2,3} = \emptyset$. Then by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_3|Z_{2,3} = x_2 \not\perp\!\!\!\perp x_3 \iff x_2 \to x_3$. Note, x_1 has no parents, so we initialize $C_{1,3} = \emptyset$. Note, as $x_1 \not\to x_2$, there are no possible mediators between x_1 and x_3: we initialize $M_{1,3} = \emptyset$. Then, by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_3|Z_{1,3} = x_1 \not\perp\!\!\!\perp x_3 \iff x_1 \to x_3$. We recover all possible edges between $[x_1, x_2, x_3]$.

4. Finding parents of x_4: first we check whether $x_3 \to x_4$, then whether $x_2 \to x_4$, then whether $x_1 \to x_4$. From iteration 3, we know all edges between $[x_1, x_2, x_3]$.

   Case 1: if $[x_1, x_2, x_3]$ has no edges, then no node causes x_3 directly or indirectly, therefore we initialize $C_{3,4} = \emptyset$. There are no possible mediators between x_3 and x_4, so $M_{3,4} = \emptyset$. Therefore, by Lemma 5.1 $x_3 \not\perp\!\!\!\perp x_4|Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 \iff x_3 \to x_4$. As $[x_1, x_2, x_3]$ has no edges, no node causes x_2 directly or indirectly, therefore we initialize $C_{2,4} = \emptyset$. Note, $x_2 \not\to x_3$, so there are no possible mediators between x_2 and x_4, so we initialize $M_{2,4} = \emptyset$. Then, by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_4|Z_{2,4} = x_2 \not\perp\!\!\!\perp x_4 \iff x_2 \to x_4$. Note, x_1 has no parents, so we initialize $C_{1,4} = \emptyset$. As, x_1 does not cause x_2 or x_3 there are no possible mediators between x_1 and x_4, therefore we initialize $M_{1,4} = \emptyset$. Then, by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_4|Z_{1,4} = x_1 \not\perp\!\!\!\perp x_4 \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

   Case 2: if $x_1 \to x_2$ is the only edge between the nodes $[x_1, x_2, x_3]$, then no node causes x_3 directly or indirectly, therefore $C_{3,4} = \emptyset$. By topological sort there is no mediator

between x_3, x_4, so we initialize $M_{3,4} = \emptyset$. Then, by Lemma 5.1 $x_3 \not\perp\!\!\!\perp x_4 | Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 \iff x_3 \to x_4$. As $x_1 \to x_2$, we initialize $C_{2,4} = x_1$. As $x_2 \not\to x_3$, there are no possible mediators between x_2 and x_4, so we initialize $M_{2,4} = \emptyset$. Then, by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_4 | Z_{2,4} = x_2 \not\perp\!\!\!\perp x_4 | x_1 \iff x_2 \to x_4$. If an edge exists between x_2 and x_4, we initialize $M_{1,4}$ accordingly. As x_1 has no parents, we initialize $C_{1,4} = \emptyset$. Then, by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_4 | Z_{1,4} \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

Case 3: if $x_1 \to x_3$ is the only edge between the nodes $[x_1, x_2, x_3]$, then x_1 is the only potential confounder of x_3 and x_4 so we initialize $C_{3,4} = x_1$. There are no possible mediators between x_3 and x_4, so we initialize $M_{3,4} = \emptyset$. Then, by Lemma 5.1 $x_3 \not\perp\!\!\!\perp x_4 | Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 | x_1 \iff x_3 \to x_4$. Note, x_2 has no parents, so we initialize $C_{2,4} = \emptyset$. Note, $x_2 \not\to x_3$, so we initialize $M_{2,4} = \emptyset$. Then, by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_4 | Z_{2,4} = x_2 \not\perp\!\!\!\perp x_4 \iff x_2 \to x_4$. If an edge exists between x_3 and x_4, we initialize $M_{1,4}$ accordingly. As x_1 has no parents, we initialize $C_{1,4} = \emptyset$. Then, by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_4 | Z_{1,4} \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

Case 4: if $x_1 \to x_2, x_2 \to x_3$ are the only edges between nodes $[x_1, x_2, x_3]$, then x_1 and x_2 cause x_3 either indirectly or directly. Therefore, we initialize $C_{3,4} = \{x_1, x_2\}$. There are no possible mediators between x_3 and x_4, so we initialize $M_{3,4} = \emptyset$. Then, by Lemma 5.1 $x_3 \not\perp\!\!\!\perp x_4 | Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 | x_1, x_2 \iff x_3 \to x_4$. If an edge exists between x_3 and x_4, we initialize $M_{2,4}$ accordingly. Note, x_1 is a parent of x_2, so we initialize $C_{2,4} = x_1$. Then, by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_4 | Z_{2,4} \iff x_2 \to x_4$. If an edge exists between x_3 and x_4, and/or between x_2 and x_4, initialize $M_{1,4}$ accordingly. Note, x_1 has no parents, so initialize $C_{1,4} = \emptyset$. Then by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_4 | Z_{1,4} \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

Case 5: if $x_1 \to x_3, x_2 \to x_3$are the only edges between nodes $[x_1, x_2, x_3]$, then x_1 and x_2 cause x_3 directly. Therefore, we initialize $C_{3,4} = \{x_1, x_2\}$. There are no possible mediators between x_3 and x_4, so we initialize $M_{3,4} = \emptyset$. Then, by $L3$ $x_3 \not\perp\!\!\!\perp x_4 | Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 | x_1, x_2 \iff x_3 \to x_4$. If an edge exists between x_3 and x_4, we initialize $M_{2,4}$ accordingly. Note, x_2 has no parents, so we initialize $C_{2,4} = \emptyset$. Then,by $L3$ $x_2 \not\perp\!\!\!\perp x_4 | Z_{2,4} \iff x_2 \to x_4$. If an edge exists between x_3 and x_4, initialize $M_{1,4}$ accordingly. Note, x_1 has no parents, so initialize $C_{1,4} = \emptyset$. Then by $L3$ $x_1 \not\perp\!\!\!\perp x_4 | Z_{1,4} \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

Case 6: if $x_1 \to x_2, x_1 \to x_3, x_2 \to x_3$ are the only edges between nodes $[x_1, x_2, x_3]$, then x_1 and x_2 cause x_3 directly. Therefore, we initialize $C_{3,4} = \{x_1, x_2\}$. There are no possible mediators between x_3 and x_4, so we initialize $M_{3,4} = \emptyset$. Then, by Lemma 5.1 $x_3 \not\perp\!\!\!\perp x_4 | Z_{3,4} = x_3 \not\perp\!\!\!\perp x_4 | x_1, x_2 \iff x_3 \to x_4$. If an edge exists between x_3 and x_4, we initialize $M_{2,4}$ accordingly. Note, x_1 is a parent of x_2, so we initialize $C_{2,4} = x_1$. Then, by Lemma 5.1 $x_2 \not\perp\!\!\!\perp x_4 | Z_{2,4} \iff x_2 \to x_4$. If an edge exists between x_3 and x_4, and/or between x_2 and x_4, initialize $M_{1,4}$ accordingly. Note, x_1 has no parents, so initialize $C_{1,4} = \emptyset$. Then by Lemma 5.1 $x_1 \not\perp\!\!\!\perp x_4 | Z_{1,4} \iff x_1 \to x_4$. We recover all possible edges between $[x_1, x_2, x_3, x_4]$.

5. Finding parents of x_5: first we check whether $x_4 \to x_5$, then whether $x_3 \to x_5$, then whether $x_2 \to x_5$, then whether $x_1 \to x_5$.

   $\vdots$

Iteration $k-1$ Inductive Assumption We have recovered the edges between $[x_1, \ldots, x_{k-1}]$.

Iteration k We now find all nodes in $[x_1, \ldots, x_{k-1}]$ that cause x_k (which yields all edges between $[x_1, \ldots, x_k]$).

Base Case Sub-Iteration (1,2)

1. We first check whether $x_{k-1} \to x_k$. We initialize the potential confounders $C_{k-1,k}$ using$[x_1, \ldots, x_{k-1}]$. The set of mediators $M_{k-1,k}$ is empty by the topological sort. Then, by Lemma 5.1 $x_{k-1} \to x_k \iff x_{k-1} \not\perp\!\!\!\perp x_k | Z_{k-1,k}$.

https://doi.org/10.52202/079017-4231

2. We now check whether $x_{k-2} \to x_k$. We initialize the potential confounders $C_{k-1,k}$ using$[x_1, \ldots, x_{k-2}]$. Only x_{k-1} can be a mediator: we know whether x_{k-2} causes x_{k-1}, and in our previous step we found whether x_{k-1} causes x_k. Thus, we initialize $M_{k-2,k}$ accordingly. Thus, by Lemma 5.1 $x_{k-2} \to x_k \iff x_{k-2} \not\perp\!\!\!\perp x_k | Z_{k-2,k}$.

   $\vdots$

Sub-Iteration j Inductive Assumption We have recovered edges between $[x_j, \ldots, x_k]$

Sub-Iteration $j-1$ We now find if x_{j-1} causes x_k.

1. For node x_{j-1} where $1 \leq j-1 < k$, we obtain $C_{j-1,k}$ by Iteration $k-1$ Inductive Assumption and $M_{j-1,k}$ by Sub-Iteration j Inductive Assumption. Then, by Lemma 5.1 $x_{j-1} \to x_k \iff x_{j-1} \not\perp\!\!\!\perp x_k | Z_{j-1,k}$.

Sub-Iteration j Inductive Assumption is satisfied for $j-1$, therefore we recover all nodes in $[x_1, \ldots, x_{k-1}]$ that cause x_k. This satisfies Iteration $k-1$ Inductive Assumption for k, which means we recover all edges between $[x_1, \ldots, x_k]$. Thus, for a topological sort of arbitrary length, the algorithm recovers all possible edges.

□

C.4 Time Complexity for Edge Discovery

Theorem 5.3. *Given n samples of d vertices generated by a model corresponding to a DAG G, the runtime complexity of ED is upper bounded by $O(d^2 n^3)$.*

Proof. ED checks for the existence of every edge permitted by a topological sort π by running one conditional independence test that has complexity $O(n^3)$. In the worst case, there are $O(d^2)$ possible edges, so the overall complexity is $O(d^2 n^3)$. □

D Additional Experiments and Runtimes

D.1 Runtimes for Linear Topological Sort

d	LHTS	DirectLiNGAM	R^2-Sortability
10	0.48 ± 0.08	0.06 ± 0.01	0.00 ± 0.00
20	1.37 ± 0.49	0.42 ± 0.08	0.00 ± 0.00
30	3.34 ± 0.86	3.51 ± 0.22	0.02 ± 0.02
50	15.31 ± 4.46	20.39 ± 0.40	0.08 ± 0.04

n	LHTS	DirectLiNGAM	R^2-Sortability
100	0.28 ± 0.12	0.05 ± 0.01	0.00 ± 0.00
300	0.34 ± 0.07	0.05 ± 0.01	0.00 ± 0.00
500	0.58 ± 0.21	0.06 ± 0.00	0.00 ± 0.00
700	1.96 ± 0.46	0.10 ± 0.02	0.00 ± 0.00
1000	3.16 ± 0.75	0.10 ± 0.01	0.00 ± 0.00
1500	5.48 ± 1.29	0.24 ± 0.03	0.00 ± 0.00
2000	8.71 ± 2.76	0.27 ± 0.04	0.00 ± 0.00

Figure 10: Runtimes for linear topological sorts, left: top row; right: bottom row, see Figure 5.

D.2 Runtimes for Nonlinear Topological Sort

Noise	NHTS	DLNGM	NoGAM	GES	GRaSP	GSP	R2ST
Gaussian	15.32 ± 8.73	0.15 ± 0.03	63.95 ± 1.51	0.25 ± 0.18	0.01 ± 0.01	0.01 ± 0.00	0.00 ± 0.00
Uniform	26.38 ± 12.35	0.16 ± 0.04	63.55 ± 1.42	0.54 ± 0.27	0.03 ± 0.02	0.01 ± 0.00	0.00 ± 0.00
Laplace	22.81 ± 6.59	0.17 ± 0.03	64.14 ± 0.48	0.67 ± 0.18	0.03 ± 0.02	0.01 ± 0.01	0.00 ± 0.00

Figure 11: Runtimes for nonlinear topological sorts, see Figure 6.

D.3 Topological Sorts on Nonlinear Data

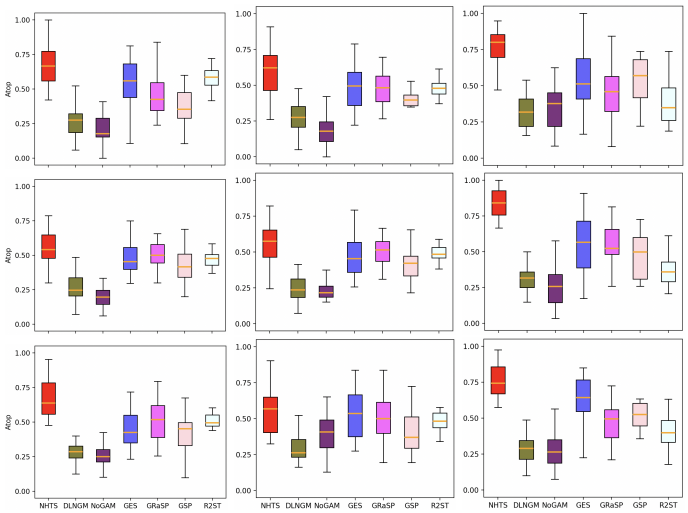


Figure 12: Performance of NHTS on data generated with Gaussian, Laplace, or Uniform noise (left, middle, right columns), the average number of edges set to $2d$, $3d$, or $4d$ (top, middle, bottom rows).

In figure 12 we provide additional experiments for nonlinear topological sorts ($d = 10, n = 300$); we see that NHTS maintains superior performance even as the density of the underlying graph increases, although the performance gap decreases, especially for laplacian noise, as the graph becomes denser.

https://doi.org/10.52202/079017-4231

D.4 Edge Pruning on Nonlinear Data

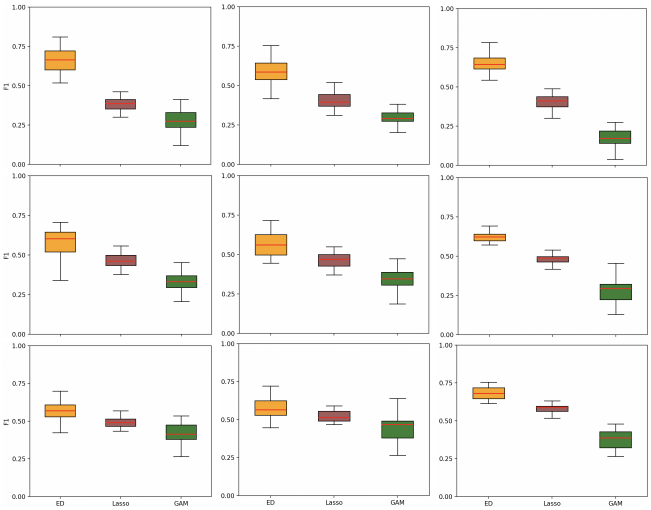


Figure 13: Performance of ED on data generated with Gaussian, Laplace, or Uniform noise (left, middle, right columns), the average number of edges set to $2d$, $3d$, or $4d$ (top, middle, bottom rows).

In figure 13 we provide additional experiments for edge pruning methods ($d = 20, n = 300$); ED generally maintains superior performance as the noise distribution is varied and density is increased, although the performance gap decreases for all noise distributions as the graph becomes denser.

D.5 Runtimes for Edge Pruning

d	ED	Lasso	GAM	RESIT
4	0.16 ± 0.08	0.01 ± 0.00	0.06 ± 0.01	0.37 ± 0.02
8	0.94 ± 1.38	0.07 ± 0.05	0.16 ± 0.02	3.81 ± 0.05
12	2.10 ± 2.25	0.43 ± 0.65	0.24 ± 0.03	14.05 ± 0.38
16	2.42 ± 1.27	0.78 ± 1.44	0.33 ± 0.04	36.18 ± 2.64

d	ED	Lasso	GAM
30	9.54 ± 6.81	2.79 ± 0.45	0.69 ± 0.11
50	15.20 ± 7.48	11.36 ± 5.93	6.46 ± 0.51
70	33.92 ± 26.03	35.13 ± 8.29	14.92 ± 1.14

n	ED	Lasso	GAM
300	1.38 ± 1.40	0.18 ± 0.24	0.21 ± 0.02
700	7.39 ± 5.79	0.20 ± 0.04	0.22 ± 0.03
1000	16.17 ± 10.24	0.23 ± 0.04	0.23 ± 0.03
1300	51.68 ± 23.77	0.63 ± 0.36	0.32 ± 0.05

Figure 14: Runtimes for edge pruning: tables correspond to the graphs in Figure 7, from left to right.

E Implementation Details

All tests were done in Python. All runtimes were computed locally on an Apple M2 Pro Chip, 16 Gb of RAM, with no parallelization.

E.1 Topological Sort for LiNGAM

DirectLiNGAM was imported from the `lingam` [45] package, R^2-sort was imported from the `CausalDisco` [28, 29] package, and LHTS was implemented using the `Sklearn` [24] package. All assets used have a CC-BY 4.0 license. We follow [38] to generate the data for Figure 5, using linear causal mechanisms with randomly drawn coefficient values, plus independent uniform noise. Data is standardized to remove shortcuts [28]. See Github repository for more details. Cutoff values for independence tests were set to $\alpha = 0.05$ for all methods.

E.2 Topological Sort for Nonlinear ANM

GES and GRaSP were imported from the `causal-learn` [49] package; GSP was imported from the `graphical_model_learning` [43] package. DirectLiNGAM was imported from the `lingam` package [45]. NoGAM was imported from the `dodiscover` [14] package. R^2-sort was imported from the `CausalDisco` package. NHTS and LoSAM were implemented using the kernel ridge regression (KRR) function from the `Sklearn` package, used independence tests from either the `causal-learn` package or the `dcor` [26] package, and a mutual information estimator from the `npeet` [44] package. All assets used have a CC-BY 4.0 license. We follow [16] to generate the data used for Figure 6 and Figure 12, using quadratic causal mechanisms with randomly drawn coefficient values, plus independent gaussian, laplace or uniform noise. Features generated with quadratic mechanisms were standardized after being generated to remove shortcuts [28] and to prevent the quadratic mechanisms from driving all values close to 0 (ensuring stability). See Github repository for more details. Note that, to enable a fair comparison between NHTS and other topological ordering methods, we implement a version of NHTS that returns a linear topological sort, rather than a hierarchical topological sort, by adding only one vertex to the sort in each iteration of its sorting procedure. For NHTS, in Stage 2 we used KRR with polynomial kernel, $\alpha = 1$, degree = 3, coef0 = 1, and in Stage 4 we used KRR with RBF kernel, $\alpha = 0.1$, $\gamma = 0.01$. Cutoff values for independence tests were set to $\alpha = 0.05$ for all methods, no cross validation was allowed for any method. Otherwise, default settings were used for all baselines.

E.3 Edge Pruning

Lasso and RESIT were implemented using the `sklearn` package, hypothesis testing with GAMs was implemented using Bsplines and GLMGam from the `statsmodel` [33] package. Independence tests used either the `causal-learn` package or the `dcor` package. All assets used have a CC-BY 4.0 license. We follow [16] to generate the data used for Figure 7 and Figure 13, using quadratic causal mechanisms with randomly drawn coefficient values, plus independent uniform, gaussian, or laplace noise. Features generated with quadratic mechanisms were standardized after being generated to remove shortcuts [28] and to prevent the quadratic mechanisms from driving all values close to 0 (ensuring stability). See Github repository for more details. Cutoff values for independence tests were set to $\alpha = 0.05$ for all methods.

 https://doi.org/10.52202/079017-4231

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We address all claims made in the abstract and contributions section throughout the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly outline the assumptions and identifiability conditions needed for our methods to hold, only claiming aysymptotic correctness when appropriate. See Definition 2.2, Appendix A.1 and Appendix B.1.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Each theoretical result includes the necessary assumptions, and links to a correct proof in the Appendix.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experimental procedures are described in both Section 6 and Appendix E, and code is released on github.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code for the algorithms and data generation is provided in the supplemental material and github link.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 6, Appendix E, and the supplemental material or github link.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars for experiments for linear topological sorts and edge pruning methods in the main text are standard deviations. Error bars for nonlinear topological sorts and edge pruning experiments not in the main text are whiskers on a boxplot (1.5 times the IQR).

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See Appendix E.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All parts of the Code of Ethics were followed.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper addresses a foundational problem in the field of causal discovery, an issue not fundamentally tied to any specific set of applications.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Not applicable.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix E.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets introduced in the paper.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

https://doi.org/10.52202/079017-4231