Causal Discovery under Latent Class Confounding

Anonymous Author
Anonymous Institution

Abstract

An acyclic causal structure can be described with directed acyclic graph (DAG), where arrows indicate the possibility of direct causation. The task of learning this structure from data is known as "causal discovery." Diverse populations or changing environments can sometimes give rise to data that is heterogeneous in the following sense: each population/environment is a "source" which idiosyncratically determines the forms of those direct causal effects. From this perspective, the source is a latent common cause for every observed variable. While some methods for causal discovery are able to work around latent confounding in special cases, especially when only few observables are confounded, a global confounder is a difficult challenge. The only known ways to deal with latent global confounding involve assumptions that limit the structural equations and/or noise functions. We demonstrate that globally confounded causal structures can still be identifiable with arbitrary structural equations and noise functions, so long as the number of latent classes remains small relative to the size and sparsity of the underlying DAG.

1 INTRODUCTION

Structural Causal Models Many modern approaches to studying causal systems use structural causal models (SCMs) to graphically model causal relationships in a directed acyclic graph (DAG) (Pearl, 2009; Peters et al., 2017). In an SCM, $A \to B$ indicates "A is allowed to have a direct effect on B." These graphical models allow the determination of covariate adjustments to identify intervention outcomes.

"Causal discovery" is the task of recovering a causal DAG from data. The simplest algorithms for causal discovery make use of a correspondence between the conditional independence of the observed variables and graphical properties of the underlying SCM. These "d-separation rules"[1] give graphical criteria for independence and dependence under an assumption known as faithfulness, essentially that for each arrow is an actual direct effect. So, $A \perp\!\!\!\perp C \mid B$ is sufficient to imply that there is no arrow between A and C. We say that B is a "separating set" for A, C. A "Markov equivalence class" describes a set of possible DAG structures that agree with a set of conditional independencies, often depicted with a CPDAG that gives the undirected skeleton of the true graph and partial orientation of the edges (Andersson et al., 1997).

Causal discovery with unobserved confounding The PC algorithm was the first causal discovery algorithm to make use of conditional independence, outlined in Spirtes et al. (2000a). Many causal discovery algorithms have been developed since, summarized in Squires and Uhler (2023). A number of these algorithms address the presence of latent confounding using one of two assumptions that "limit" the confounding influence: (1) low out-degree latents or (2) parametric limitations in the random variables.

The first type of assumption involves limiting the *out-degree* of latent confounding. For example, the FCI algorithm can detect the presence of unobserved confounders that act on *only two* observed variables (Spirtes et al., 1993; Spirtes, 2001). Richardson and Spirtes (2002)'s seminal work introduced ancestral graphs for the general study of this setting.

Global Confounding Recent work has studied "global" or "pervasive" confounding (Gordon et al., 2023; Frot et al., 2019; Agrawal et al., 2023). Pervasive confounders affect all of the observable variables in a system and are often seen when data is gathered over a large area or over a long period of time. Such confound-

Preliminary work. Under review by AISTATS 2025. Do not distribute.

[1] See Pearl (2009) for a review of d-separation or Appendix A for a summary of important results used in this manuscript.

ing is graphically modeled with a single unobserved variable that points to every observable variable (see Figure 1). This structure d-connects all possible pairs of vertices, making all variables statistically dependent even after conditioning on any other variables, which rules out use of the PC algorithm.

Pervasively confounded DAGs cannot be learned using only conditional independence tests. Instead, these settings requiring parametric assumptions that restrict the space of probability distributions. For example, Frot et al. (2019) was able to show superior performance in settings with large-degree confounders with linear relationships and additive sub-Gaussian or elliptical noise. Other settings include linear structural equations with non-Gaussian additive noise (Cai et al., 2023) and non-linear structural equations within a finite-dimensional Hilbert space (Agrawal et al., 2023).

A relatively unstudied setting (with respect to causal discovery) involves a discrete unobserved pervasive confounder that imposes a "mixture" or "latent class". Such a setting commonly emerges whenever data is gathered from multiple populations (unobserved "type" of person, cell, country, etc.) or laboratories (unobserved "source" of data). Methods designed for continuous latent confounders are ill-equipped to take advantage of the special properties of mixture models.

A well studied strategy for recovering (and therefore deconfounding) mixture models involves making assumptions that again restrict the space of possible probability distributions. For example, assuming that all single-component distributions are Gaussian allows for the decomposition of a bimodal (non-Gaussian) distribution into two Gaussian components. These assumptions are restrictive and difficult to verify, particularly when dealing with categorical data with no restriction beyond having finite support. Instead, almost all of the research on categorical data leverages an assumption of mutually independent observed variables within the source distributions (Cryan et al., 2001; Freund and Mansour, 1999; Chaudhuri and Rao, 2008; Feldman et al., 2008; Rabani et al., 2014; Chen and Moitra, 2019; Gordon et al., 2021, 2024) (see also the earlier seminal work of Kearns et al. (1994)). We call this the k-MixProd problem (where k denotes the number of latent classes), whose identifiability is discussed in more detail in Appendix B.

k-MixProd assumes that the causal model within each source is an empty graph. Early work by Anandkumar et al. (2012) and Gordon et al. (2023) broadened this class of independence assumptions for mixture identifiability, exploring Markov random fields and Bayesian networks respectively. A key assumption of Gordon et al. (2023) and k-MixProd is that the Bayesian DAG structure within each mixture component is known. To date there is no result showing if and when the within-source DAG structure can be identified from the observed data of a mixture model, particularly because the assumptions for previous results on continuous pervasive confounding are too restrictive. This paper will develop an algorithm for recovering this structure, implicitly giving conditions for identifiability. This structure could then be used to recover the mixture under very mild assumptions.

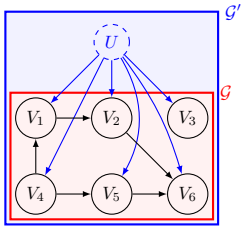


Figure 1: The goal is to learn the graph structure $\mathcal{G}$ *without* observing U.

Problem Statement Suppose we augment a DAG $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ with an unobserved U that has k "latent classes", each of which leaves a distinctive signal on all the observed variables $\boldsymbol{V}$. The result is a mixture-model whose DAG, $\mathcal{G}' = (\boldsymbol{V} \cup \{U\}, \boldsymbol{E}')$ includes additional arrows from U to every $V \in \boldsymbol{V}$. See Figure 1 for an example. We will refer to $\mathcal{G}$ as the observed sub-graph and we will use $\Delta := \max_{V \in \boldsymbol{V}} \deg^{\mathcal{G}}(V)$ to denote its maximum (in plus out) degree.

The goal is to to uncover the observed sub-graph's structure $\mathcal{G}$ up to a Markov equivalence class (i.e. a CPDAG) using statistics gained from the "observed probability distribution," i.e. $\Pr(\boldsymbol{V})$ marginalized over U. Notice that U confounds all pairs of variables in $\boldsymbol{V}$, which requires it to be in *every* separating set. This means that the correspondence between conditional independence and d-separation is no longer adequate to make any structural deductions about the graph.

Assumptions We will assume (1) that the distribution is faithful[2] with respect to $\mathcal{G}'$, and (2) that U is discrete with a known number of latent classes k. k represents the "complexity" of unobserved confounding, as more latent classes are capable of exerting "more complex" signals. Since our deductions rely upon each value of U having distinctive effects upon the observables, the most difficult case of the problem is Bernoulli observable variables that are expressed using only a single parameter. Compositions of such variables are adequate to express any finite-range variables. Hence we focus entirely upon the case of Bernoulli observables.

[2]The precise assumption is a slight extension of faithfulness to the mixture setting, discussed later.

Motivation A concrete motivating example is found in single-cell RNA sequencing, which measures the gene expressions of individual cells, often without knowledge of the underlying cell type. "Gene regulatory networks" can be modeled using SCMs whose arrows describe the causal relationships between the expression levels of each gene (Lähdesmäki et al., 2003). Here, the latent "cell-type" affects overall expression of genes which often act on the same gene regulatory network. Gene expression is often zero-inflated, meaning it can be reduced to a binary variable indicating "expressed" vs. "not expressed." Under this reduction, task of learning the regulatory network exactly corresponds to the problem studied here. While learning mixtures of cell-type-specific regulatory networks is a slightly more challenging problem[3], we will discuss how insights from Phase II could be used to uncover component-based differences in the regulatory networks.

1.1 Main Result

This paper will prove the identifiability of causal structures within the discrete mixture setting by developing the first known algorithm for this problem.

Theorem 1. *Consider $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ with mixture source $U \in \{1, \ldots, k\}$ and degree bound Δ. $\Omega(\log(k)\Delta^3)$ vertices[4] are sufficient to generically[5] identify $\mathcal{G}$ up to its Markov equivalence class using our algorithm.*

Using an oracle that can solve k-MixProd in time τ and an oracle that can solve for non-negative rank in time ρ we give an algorithm that runs in time $|\boldsymbol{V}|^{\mathcal{O}(\Delta^2 \log(k))}\rho + \mathcal{O}(k|\boldsymbol{E}|2^{\Delta^2})\tau$. See Appendix C for further discussion on the runtime of ρ and τ.

Our algorithm will build on two key ideas. The first of these ideas is that the rank of a matrix formed by the joint probabilities of two variables contains information about the graphical structure on those variables under latent class confounding. We observe that the joint probability distribution of two independent variables of cardinality ℓ forms an $\ell \times \ell$ matrix that can be written as a rank 1 outer-product. Hence, marginalizing over a k-mixture gives us a linear combination of these rank 1 matrices, which will generically be rank k.

Of course, finite data only affords a stochastic perturbation of the true matrix of joint probabilities. A naive approach to testing the rank of the underlying joint probability distribution involves thresholding singular values as in Anandkumar et al. (2012). Such an approach is unstable, with many practical difficulties associated with selecting the correct threshold. Instead, we modify a statistical test based on Ratsimalahelo (2001) to obtain a hypothesis test for the rank of an estimated joint probability matrix. We demonstrate the superiority of this approach and note that this test may be of fundamental use in other mixture settings.

The idea of using matrix rank is limited by the cardinality of our variables — the joint probability distribution of two binary observables forms a 2×2 matrix whose rank is no larger than 2. This problem leads to the second idea: We can "coarsen" variables by joining sets of variables. These "super variables" take on values in the Cartesian product of their components' alphabets. For example, three Bernouli variables can be combined into a single variable of cardinality 8. This paper develops the notion of testing rank on these coarsened variables for causal discovery.

1.2 Outline of Algorithm

The PC-algorithm works in two phases. The first phase begins with a complete graph (i.e. all possible edges) and uses conditional independence tests to find non-adjacencies and "separating sets" that d-separate them. When a non-adjacency is found, the corresponding edge is removed and its separating set is stored. The second phase uses the separating sets from the first phase to orient immoralities (as well as further propagation of edge orientation via the Meek rules, (Meek, 1995)). Our algorithm will mirror the PC Algorithm, but will split the first phase into two parts for three total phases.

Phase I will again begin with a complete graph and remove edges between variables when we find evidence of non-adjacency (this time using rank tests). Phase I will only test independence on groupings of variables, so its termination will not guarantee that we have discovered all possible non-adjacencies. Instead, a provably small subset of the graph will have *false positive* edges. In Phase II, we will make use of the structure we have uncovered so far to induce instances of k-MixProd within conditional probability distributions. A k-MixProd solver will then identify the joint probability distribution between subsets of $\boldsymbol{V}$ and the latent class U. Access to this joint probability distribution enables a search for separating sets that include U, meaning that the rest of the structure can be resolved using traditional conditional independence tests (following the standard steps of the PC algorithm).

Phase III mirrors the last phase of the PC algorithm: identifying immoralities using non-adjacencies and separating sets, and then propagating orientations according to Meek rules. This phase is no different from the PC algorithm, so its details are omitted.

[3] see Saeed et al. (2020) and (Varici et al., 2024) as well as Strobl (2023) for mixtures of cyclic DAGs

[4] Ω is asymptotic notation.

[5] Lebesgue measure 1 in the space of parameters.

1.3 Notation

We will use the capital Latin alphabet to denote random variables, which are vertices on our DAG. When referring to sets of these variables, we will use bolded font, e.g. $\boldsymbol{X} = \{X_1, X_2, \ldots\}$. To refer to components of a graph, we will use the following operators: $\mathbf{PA}(V), \mathbf{CH}(V)$ will refer to the parents and children of V. $\mathbf{AN}(V), \mathbf{DE}(V)$ will refer to the ancestors and descendants of V. $\mathbf{AN}(V) \cup \{V\}$ and $\mathbf{DE}(V) \cup \{V\}$ are denoted using $\mathbf{AN}^+(V), \mathbf{DE}^+(V)$ respectively. $\mathbf{MB}(V) = \mathbf{PA}(V) \cup \mathbf{CH}(V) \cup \mathbf{PA}(\mathbf{CH}(V))$ will refer to the Markov boundary (i.e. the unique minimal Markov blanket) of V (Pearl, 2009). $\mathbf{NB}_\ell(V)$ refers to the distance ℓ neighborhood of V.

As these operators act on graphs, they can specify the graph structure being used in the superscript, e.g. $\mathbf{PA}^{\mathcal{G}}(V)$. Unless otherwise specified, operators should be assumed to apply to the observed subgraph $\mathcal{G}$ and not $\mathcal{G}'$, i.e. $U \notin \mathbf{PA}^{\mathcal{G}}(V)$. We will also occasionally write tuples to indicate the intersection of the sets for two vertices, e.g. $\mathbf{CH}(V, W) = \mathbf{CH}(V) \cap \mathbf{CH}(W)$. Finally, these operators can also act on sets to indicate the union of the operation, e.g.

$$\mathbf{PA}(\boldsymbol{X}) = \bigcup_{X \in \boldsymbol{X}} \mathbf{PA}(X) \setminus \boldsymbol{X}. \tag{1}$$

We make use of lowercase letters to denote assignments, for example x will denote the assignment $X = x$. This can also be applied to operators, e.g. $\mathbf{mb}(V)$ denotes an assignment to $\mathbf{MB}(V)$. Assignments can be specified by a set of assignments in the subscript, e.g. $\mathbf{mb}_{\boldsymbol{c}}(V)$ obtains assignments for $\mathbf{MB}(V) \subseteq \boldsymbol{C}$ from the assignments of $\boldsymbol{c}$ to $\boldsymbol{C}$. $\mathrm{rk}_+(\boldsymbol{\mathcal{M}})$ will denote the nonnegative rank of matrix $\boldsymbol{\mathcal{M}}$.

2 RANK TESTS

This section will develop "rank tests" which will serve as a replacement for conditional independence tests as a test for d-separation or d-connectedness.

2.1 Checking for k-Mixture Independence

To determine non-adjacency, we will take advantage of a signature U leaves on the marginal probability distributions of variables which are independent conditional on U. First, we interpret the marginal probability distribution as a matrix.

Definition 1. Given two discrete variables $X, Y \in \boldsymbol{V}$ each with $|X| = |Y| = m$, define the "probability matrix" $\boldsymbol{\mathcal{M}}[X, Y] \in [0, 1]^{m \times m}$ to be

$$\boldsymbol{\mathcal{M}}[X, Y]_{x,y} := \Pr(x, y), \tag{2}$$

where x, y both range from $1, \ldots, m$. Similarly, for $\boldsymbol{C} \subseteq \boldsymbol{V}$, define

$$\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}]_{x,y} := \Pr(x, y \mid \boldsymbol{c}). \tag{3}$$

We now notice that we can decompose the probability matrix into a linear combination of conditional probability matrices for each source, for which Lemma 1 gives an upper bound on rank.

Lemma 1. *Given a mixture of Bayesian network distributions that are Markovian in $\mathcal{G}$, if $X \perp\!\!\!\perp_d^{\mathcal{G}} Y \mid \boldsymbol{C}$, then for all $\boldsymbol{c}$, $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}]) \leq k$.*

Proof. We can decompose $\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}]$ as follows:

$$\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}] = \sum_u \Pr(u) \boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}, u]. \tag{4}$$

$X \perp\!\!\!\perp Y \mid \boldsymbol{C}, U$, so $\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}, u]$ can be written as the outer product of two vectors describing the probabilities of each variable. Therefore, we conclude that $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}, u]) = 1$. If $U \in [k]$, then $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}]) \leq k$. □

Having shown that d-separation in $\mathcal{G}$ upper bounds the rank of probability matrices, we now seek a lower bound on the rank in the case of d-connectedness. This will require a "faithfulness-like" assumption that the $\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}, u]$ terms in in Equation 4 are linearly independent, resulting in an overall $\mathrm{rk}_+ \boldsymbol{\mathcal{M}}(X, Y \mid \boldsymbol{c}) > k$ when X and Y are not independent conditioned on U. Lemma 2 shows that this condition holds generically.

Lemma 2. *Given a mixture of Bayesian network distributions, each of which is faithful to $\mathcal{G}$. If $X \not\perp\!\!\!\perp_d^{\mathcal{G}} Y \mid \boldsymbol{C}$ and $|X| = n$, $|Y| = m$ with $n, m > k$, then for all $\boldsymbol{c}$, $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[X, Y \mid \boldsymbol{c}]) > k$ with Lebesgue measure 1.*

The proof of Lemma 2 (see Appendix F) involves applying faithfulness to each summand of Equation 4. Lemma 1 and Lemma 2 provide a generically necessary and sufficient condition for detecting $V_i \perp\!\!\!\perp_d^{\mathcal{G}} V_j$.

Lemma 3 (Rank Test)**.** *For V_i, V_j with cardinality $> k$, $V_i \perp\!\!\!\perp_d^{\mathcal{G}} V_j \mid \boldsymbol{C}$ if and only if (generically) $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[V_i, V_j \mid \boldsymbol{C}]) \leq k$.*

2.2 Hypothesis test for rank

We will now build a test for the null-hypothesis that $\boldsymbol{\mathcal{A}} := \boldsymbol{\mathcal{M}}[V, V' \mid \boldsymbol{C}]$ has rank $\leq k$. For this derivation, we use $V_i = V$ and $V_j = V'$ to free up i, j for indexing entries of matrices. This section will closely follow Ratsimalahelo (2001)'s test of the rank of a matrix $\boldsymbol{\mathcal{A}}$ under an asymptotically normal perturbation

$$\hat{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{A}} + \varepsilon(N) \boldsymbol{\mathcal{B}} \tag{5}$$

with $\varepsilon(N) \propto 1/N$ for N samples.

To apply the test of rank to our setting, we break down $\hat{\boldsymbol{\mathcal{A}}}$ into the sum of indicator functions which are applied to each of N data entries.

$$\boldsymbol{\mathcal{C}}(v, v')_{i,j} \coloneqq \frac{1}{N} \mathbb{1}[v = i] \mathbb{1}[v' = j], \tag{6}$$

$$\hat{\boldsymbol{\mathcal{A}}}_{i,j} = \sum_{v,v' \in \text{Data}} \boldsymbol{\mathcal{C}}(v, v') \tag{7}$$

From this decomposition, $\varepsilon(N)\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}}_{i,j} - \hat{\boldsymbol{\mathcal{A}}}_{i,j}$ approaches normal with variance proportional to $1/N$ for larger N. This means that the covariance of $\boldsymbol{\mathcal{B}}$ can be approximated using the covariance matrix for the entries of $\boldsymbol{\mathcal{C}}$, which are Bernouli random variables with probability estimates given by the entries of $\hat{\boldsymbol{\mathcal{A}}}$.

$$\text{Cov}(\boldsymbol{\mathcal{C}}_{ij}, \boldsymbol{\mathcal{C}}_{i'j'}) = \begin{cases} \boldsymbol{\mathcal{A}}_{ij}(1 - \boldsymbol{\mathcal{A}}_{ij}) & \text{if } i = i', j = j' \\ -\boldsymbol{\mathcal{A}}_{ij}\boldsymbol{\mathcal{A}}_{i'j'} & \text{otherwise.} \end{cases} \tag{8}$$

First, decompose $\boldsymbol{\mathcal{A}}$ according to the SVD $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{U}}^\top$. If $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m \times m}$ is rank k, we will have $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}} \in \mathbb{R}^{k \times m}$. When this decomposition is done on the empirical matrix, $\hat{\boldsymbol{\mathcal{A}}}$, we define $\boldsymbol{\mathcal{U}}_2$ and $\boldsymbol{\mathcal{V}}_2$ to be the $k+1$th to mth "extra" rows of $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ respectively, and $\boldsymbol{\mathcal{L}}$ to be the diagonal matrix with the $k+1$th through mth singular values. Vectorize $\boldsymbol{\mathcal{L}}$ to $\hat{l}$ by stacking the columns of $\boldsymbol{\mathcal{L}}$.

Ratsimalahelo (2001)'s estimator uses the covariance matrix of the entries of $\hat{\boldsymbol{\mathcal{B}}} - \boldsymbol{\mathcal{B}}$, denoted Σ. For our setting, we substitute the covariance matrix for the entries of $\boldsymbol{\mathcal{C}}$ whose empricial estimates are given by Equation 8.

$$\Sigma_{i+jm, i'+j'm} \coloneqq \text{Cov}(\boldsymbol{\mathcal{C}}_{ij}, \boldsymbol{\mathcal{C}}_{i'j'}). \tag{9}$$

Now, with $\hat{\Sigma}^\dagger$ indicating the Moore-Penrose pseudoinverse and $\otimes$ indicating the Kronecker product, define

$$\hat{\boldsymbol{\mathcal{Q}}}^\dagger \coloneqq (\boldsymbol{\mathcal{V}}_2^\top \otimes \boldsymbol{\mathcal{U}}_2^\top)\hat{\Sigma}^\dagger(\boldsymbol{\mathcal{V}}_2 \otimes \boldsymbol{\mathcal{U}}_2). \tag{10}$$

According to Ratsimalahelo (2001), if f is the rank of $\boldsymbol{\mathcal{A}}$, then $N\hat{l}^\top \hat{\boldsymbol{\mathcal{Q}}}^\dagger \hat{l}$ converges to χ^2_f under N samples. This gives us a hypothesis test that is specifically designed for the rank test from this section. This approach is tested and compared to thresholding the $k+1$th singular value in Section 4.

3 ALGORITHM

We will now outline the first two phases of our algorithm, leaving out the final phase of orienting edges with respect to Meek's rules. The first phase involves coarsening sets of variables and applying rank tests. We then analyze the output of the first phase $\mathcal{G}_1$ which has removed *most but not all* of the missing edges. We define FP edges as these "leftovers" and show that they are contained within a subset of bounded size. This fact allows us to resolve all of the non-adjacencies in $\mathcal{G}$ by setting up instances of k-MixProd in Phase II.

3.1 Phase I: Coarsened Rank Tests

Lemma 3 allows for a simple generalization of classical structure-learning algorithms (such as the PC algorithm) provided that our probability matrix $\boldsymbol{\mathcal{M}}[X, Y]$ is large enough. Unfortunately, categorical variables ranging over smaller alphabets (such as the binary alphabets addressed by this paper) do not contain sufficient information to detect non-adjacency in cases of larger k. We resolve this problem by coarsening sets of small-alphabet (binary) variables into supervariables of larger cardinality.

Definition 2. Consider DAG $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, $V_i, V_j \in \boldsymbol{V}$, and sets $\boldsymbol{S}_i, \boldsymbol{S}_j \subseteq \boldsymbol{V} \setminus \{V_i, V_j\}$. We call the ordered pair $(\boldsymbol{S}_i^+, \boldsymbol{S}_j^+) = (\boldsymbol{S}_i \cup \{V_i\}, \boldsymbol{S}_j \cup \{V_j\})$ an **independence preserving augmentation (IPA)** of (V_i, V_j) if, for some **IPA conditioning set** $\boldsymbol{C} \subset \boldsymbol{V}$,

$$\boldsymbol{S}_i^+ \perp\!\!\!\perp_d^{\mathcal{G}} \boldsymbol{S}_j^+ \mid \boldsymbol{C}.$$

The creation of supervariables allows us to use conditional rank tests in the place of conditional independence tests. This leads to a modified version of the PC algorithm that searches over pairs of supervariable coarsenings instead of pairs of vertices, given in Algorithm 1.

Lemma 4. *Algorithm 1 utilizes* $|\boldsymbol{V}|^{\mathcal{O}(\Delta^2 \log(k))}$ *non-negative rank tests.*

Proof. Lemma 11 tells us that the maximum size of a separating set, is $\alpha \coloneqq (\lceil \lg(k) \rceil + 1)\Delta^2$, so we need to check $\binom{|\boldsymbol{V}|}{\alpha} + \binom{|\boldsymbol{V}|}{\alpha-1} + \ldots + \binom{|\boldsymbol{V}|}{1}$ possible separating sets, which is $|\boldsymbol{V}|^{\mathcal{O}(\Delta^2 \log(k))}$. We must iterate over all possible supervariables for each separating sets, which is upper bounded by $\binom{|\boldsymbol{V}|}{2(\lceil \lg(k) \rceil + 1)}$, which is $|\boldsymbol{V}|^{\mathcal{O}(\log(k))}$. □

It is worth noting that the exponent of $\mathcal{O}(\Delta^2 \log(k))$ in Lemma 4 is never larger than $|\boldsymbol{V}|$ because it comes from the maximum size of a separating set.

3.1.1 FP Edges

Phase I of our algorithm removes an edge between two non-adjacent variables through a rank test so long as there exists an IPA for the non-adjacency. Not all

Algorithm 1: Phase I

Input: The marginal probability distribution $\Pr(\boldsymbol{V})$, marginalized over U.
Output: An undirected graph $\mathcal{G}_1 = (\boldsymbol{V}, \boldsymbol{E}_1)$ and a separating set $\boldsymbol{C}_{ij}$ for each detected non-adjacency.

Begin with a complete undirected graph $\mathcal{G}_1 = (\boldsymbol{V}, \boldsymbol{V} \times \boldsymbol{V})$ and $d_{\max} \leftarrow |\boldsymbol{V}| - 1$.
for $\ell = 0$ *to* $\ell = d_{\max}$ **do**
  for $\boldsymbol{C} \subset \boldsymbol{V}$ *and* $|\boldsymbol{C}| = \ell$ **do**
    for $\boldsymbol{S}, \boldsymbol{S}' \subseteq \boldsymbol{V} \setminus \boldsymbol{C}$, *with* $|\boldsymbol{S}| = |\boldsymbol{S}'| = \lceil \lg(k) \rceil + 1$ **do**
      if *arbitrary assignment* $\boldsymbol{c}$ *has* $\mathrm{rk}_+(\boldsymbol{\mathcal{M}}[\boldsymbol{S}, \boldsymbol{S}' \mid \boldsymbol{c}]) \leq k$ **then**
        Remove all edges between $\boldsymbol{S}$ and $\boldsymbol{S}'$ in $\mathcal{G}_1$.
        $\boldsymbol{C}_{i,j} \leftarrow \boldsymbol{C}$ for each $V_i \in \boldsymbol{S}, V_j \in \boldsymbol{S}'$
        Update Δ to the maximum degree of $\mathcal{G}_1$.
      end
    end
  end
end

non-adjacencies will contain an IPA, so the adjacency graph $\mathcal{G}_1$ contains a *superset* of the true adjacencies.

Definition 3. $\boldsymbol{E}_1 \setminus \boldsymbol{E}$ are **false positive (FP)** edges.

To see why FP edges exist, we will introduce the notion of immoral descendants.

Definition 4 (Immoral Descendants)**.** For non-adjacent V_i, V_j the **immoral descendants** of V_i and V_j are their co-children (often called immoralities (Pearl, 2009)) and those children's descendants.

$$\mathbf{IMD}(V_i, V_j) := \mathbf{CH}(V_i, V_j) \cup \mathbf{DE}(\mathbf{CH}(V_i, V_j)) \quad (11)$$

Observation 1. Any set $\boldsymbol{C}$ such that $V_i \perp\!\!\!\perp_d V_j \mid \boldsymbol{C}$ must be disjoint from $\mathbf{IMD}(V_i, V_j)$.

Clearly, an IPA conditioning set $\boldsymbol{C}$ will need to avoid the immoral descendants $\mathbf{IMD}(V_i, V_j)$ in order to preserve $V_i \perp\!\!\!\perp V_j \mid \boldsymbol{C}$. Lemma 5 will show that $\boldsymbol{S}_i, \boldsymbol{S}_j$ must also avoid $\mathbf{IMD}(V_i, V_j)$.

Lemma 5. *All IPAs for V_i, V_j are disjoint from the $\mathbf{IMD}(V_i, V_j)$. That is, $\mathbf{IMD}(V_i, V_j) \cap \boldsymbol{S}_i^+ = \emptyset$ and $\mathbf{IMD}(V_i, V_j) \cap \boldsymbol{S}_j^+ = \emptyset$ for all IPAs $(\boldsymbol{S}_i^+, \boldsymbol{S}_j^+)$ of (V_i, V_j).*

This illustrates that FP edges can occur as pairs of vertices with too many immoral descendants, leaving no vertices to form IPAs (shown in Figure 2).

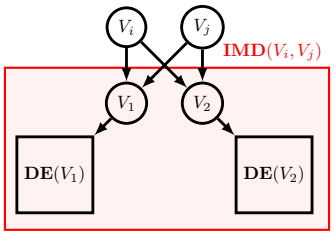


Figure 2: An FP edge after Phase I due to a large set of immoral descendants. The population variable U is omitted to avoid clutter. While V_i and V_j are d-separated by $\mathcal{C} = \emptyset$ no IPA can be made because all of the leftover vertices are immoral descendants.

3.1.2 Using non-descendants to form IPAs

Notice that the immoral descendants of a pair of vertices are always descendants of both V_i and V_j. To simplify our analysis, we focus on the existence of IPAs that are disjoint from the entire set of descendants. This allows us to show that FP edges only occur between "early vertices."

Definition 5. We define the early vertices,

$$\boldsymbol{H} := \{V \in \boldsymbol{V} \text{ s.t. } |\overline{\mathbf{DE}}(V)| < (2 + \Delta^2)(\lceil \lg(k) \rceil + 1)\}.$$

Lemma 6. *After Phase I (Algorithm 1), all of the false positive edges lie within the early vertices. More formally, $\boldsymbol{E}_1 \setminus \boldsymbol{E} \subseteq \boldsymbol{H} \times \boldsymbol{H}$.*

Observation 2. $|\boldsymbol{H}| \leq (2 + \Delta^2)\lceil \lg(k) \rceil$ and the maximum degree of $\mathcal{G}_1$ is bounded by $|\boldsymbol{H}|$.

3.2 Phase II: Handle FP Edges

Recall that the marginal probability distribution cannot use independence tests to discover non-adjacency because U confounds all of the independence properties. An important observation is that the within-source distribution $\Pr(\boldsymbol{V} \mid u)$ would not suffer from this limitation because it would allow us to condition on separating sets that include unobserved U.

Phase II will make use of this observation by selecting subsets of variables $\boldsymbol{T} \subseteq \boldsymbol{V}$ on which to obtain $\Pr(\boldsymbol{T} \mid u)$ using techniques from discrete mixture models. We will then apply regular conditional independence tests on the recovered $\Pr(\boldsymbol{T} \mid u)$ to detect FP edges. We will use a separate $\boldsymbol{T}_{ij} \ni V_i, V_j$ coarsening to verify each edge $(V_i, V_j) \in \boldsymbol{E}_1$, though this process can likely be optimized further.

The primary result on mixture model identifiability is given by Allman et al. (2009) as a direct consequence of a result by Kruskal (1977).

Lemma 7 (Allman et al. (2009))**.** *Consider the discrete mixture source $U \in \{1, \ldots, k\}$ and discrete variables*

X_1, X_2, X_3 with cardinality $\kappa_1, \kappa_2, \kappa_3$ respectively and $X_1 \perp\!\!\!\perp X_2 \perp\!\!\!\perp X_3 \mid U$. The mixture is generically identifiable (with Lebesgue measure 1 on the parameter space) if

$$\min(\kappa_1, k) + \min(\kappa_2, k) + \min(\kappa_3, k) \geq 2k + 2.$$

We can again use coarsening to form X_i with large enough κ_i. The conditions for identifiability are therefore quite mild — Phase I only needs to uncover enough sparsity to d-separate three sufficiently large independent coarsenings, one of which will be $\boldsymbol{T}_{ij}$. Conveniently, the constrained nature of our FP edges means that the graph $\mathcal{G}_1$ is sufficiently sparse to allow the construction of this setting.

$\boldsymbol{T}_{ij}$ must be designed to include enough information to discover a nonadjacency between V_i, V_j. In other words, we need to ensure that $\boldsymbol{T}_{ij}$ contains a separating set $\boldsymbol{C} \subset \boldsymbol{T}_{ij}$ such that $V_i \perp\!\!\!\perp_d^{\mathcal{G}} V_j \mid \boldsymbol{C}$. It turns out that augmenting V_i, V_j with their distance-1 neighborhood is enough to guarantee this requirement.

Definition 6. Given vertices V_i, V_j, let $\boldsymbol{T}_{ij}$ be the set containing V_i, V_j and all vertices that are distance 1 in $\mathcal{G}_1$ from V_i or V_j.

Lemma 8. *If vertices V_i, V_j are nonadjacent, the set $\boldsymbol{T}_{ij}$ contains a valid separating set $\boldsymbol{C} \subseteq \boldsymbol{T}_{ij}$ such that $V_i \perp\!\!\!\perp_d V_j \mid \boldsymbol{C}$.*

Lemma 8 guarantees that the conditional probability distribution $\Pr(\boldsymbol{T}_{ij} \mid u)$ has sufficient information to verify or falsify the adjacency of V_i and V_j.

The rest of the construction of the k-MixProd instances is left to Appendix D. Generally, it involves ensuring that the recovered $\mathcal{G}_1$ from Phase I is sparse enough to d-separate all $\boldsymbol{T}_{ij}$ from two other supervariables of sufficient cardinality. The procedure is outlined by Algorithm 2 and then Algorithm 3 performs the actual correction with the statistics recovered from k-MixProd oracles. Lemma 9 summarizes the results proved in Appendix D.

Lemma 9. *Phase II requires $\Omega(\Delta^3 \log(k))$ vertices and solves k-MixProd $\mathcal{O}(k|\boldsymbol{E}|2^{\Delta^2})$ times.*

As with Lemma 4, the Δ^2 in Lemma 9 is never larger than $|\boldsymbol{V}|$ because it comes from the maximum size of a separating set.

4 EMPIRICAL RESULTS

Our theoretical results guarantee the success of our algorithm when given infinite data. We now employ three empirical tests to show the superiority of our derived hypothesis-based rank test as well as investigate the sensitivity of Phase I. The graph structures of our tests are described here, with the details of the structural equations left to Appendix E.

4.1 Test 1: Rank Hypothesis Test

We begin with a comparison of the test developed in Section 2.2 to a naive thresholding of singular values as in Anandkumar et al. (2012). To study the differences between these tests, we generate data from two observed subgraphs.

1. "Connected" $\mathcal{G}^c$: $V_1 \to V_2 \to V_3 \to V_4$
2. "Split" $\mathcal{G}^s$: $V_1 \to V_2 \qquad V_3 \to V_4$

If we coarsen our vertices into $\boldsymbol{S}_i^+ = \{V_1, V_2\}$ and $\boldsymbol{S}_j^+ = \{V_3, V_4\}$, then these two DAGs differ in that $\boldsymbol{S}_i^+ \not\perp\!\!\!\perp_d^{\mathcal{G}^c} \boldsymbol{S}_j^+$ and $\boldsymbol{S}_i^+ \perp\!\!\!\perp_d^{\mathcal{G}^s} \boldsymbol{S}_j^+$. We note that this is a challenging test, as the connection between the two partitions is only driven by the $V_2 \to V_3$ arrow.

We varied the number of samples from these distributions from 1000 to 9000 and studied the distributions of the two reported p-values and $k+1$th singular values across 200 runs. The results are reported in Figure 3, showing that the hypothesis test has significant difference in p-values beyond 4000 samples (relative to the difference in the $k+1$th singular values).

The hypothesis test does appear to remove edges more aggressively in the low-data regime — i.e. the blue curves overlap with much of the orange curve in the low-data regime. However, it is worth noting that it is almost impossible to choose a threshold for the $k+1$th singular value ahead of time, whereas hypothesis tests give a meaningful interpretation to significance.

4.2 Test 2: Recovering a "Y" Graph

The second test is a simple recovery test on a 7 vertex graph in which two disconnected parents form a child, with a series of descendants. We sample $10,000$ data points and test with a p-value of .0005[6] — i.e. we remove edges when we get a p-value of .9995 in our rank test (a value that is motivated by knock-on effects and the concentration of the "split" graph results in Test 1). We report our results from 100 different tests (and implicitly show the tested graph) in Figure 4.

This test illustrates a few things that are not revealed in our theoretical results. The first is that edges which are far-enough apart often appear independent even before conditioning on their separating sets, presumably due

[6]Very small p-values are necessarily because many tests are applied to each pair of vertex augmentations, resulting in "p-value hacking" from the many chances to remove a particular edge.

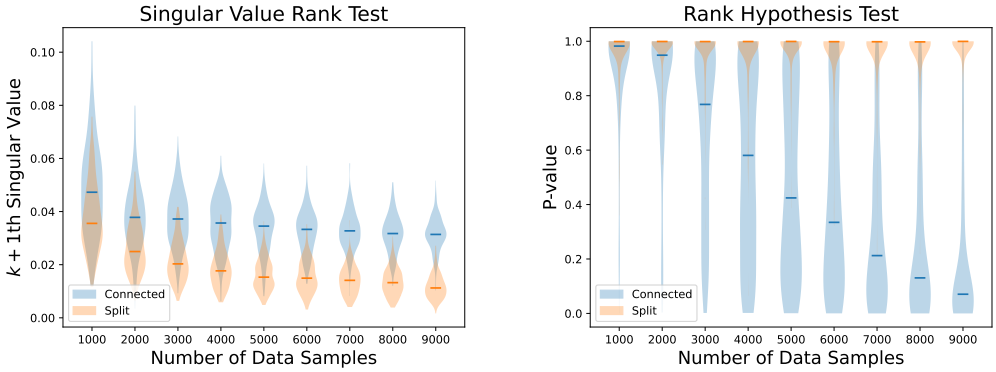
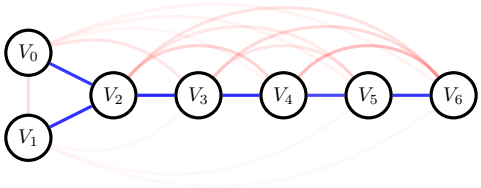


Figure 3: The results of Test 1.



	V_0	V_1	V_2	V_3	V_4	V_5	V_6
V_0	0.0	0.14	0.79	0.1	0.04	0.05	0.05
V_1	0.14	0.0	0.81	0.03	0.01	0.03	0.03
V_2	0.79	0.81	0.0	0.85	0.17	0.09	0.11
V_3	0.1	0.03	0.85	0.0	0.82	0.1	0.15
V_4	0.04	0.01	0.17	0.82	0.0	0.72	0.26
V_5	0.05	0.03	0.09	0.1	0.72	0.0	0.79
V_6	0.05	0.03	0.11	0.15	0.26	0.79	0.0

Figure 4: Results from Test 2. Correctly returned edges are blue. Red edges are **not** in the true model. Opacity shows how frequently the edge is in our returned model (we want faint red lines and strong blue lines). These frequencies are given in a table using the same scheme.

to their weak dependence. In these cases our algorithm will "incorrectly" remove edges between vertices too early, but still give the correct result (as with any other independence-based algorithm).

As is the case with most causal discovery algorithms, the frequency of false-positive edges tends to increase with the size of the separating set between the vertices. Vertices with a large separating set require a rank tests for each assignment to that separating set, leading to more "accidentally" dependence. These "knock-on effects" are often handled using p-value adjustments, suggesting that a smaller p-value thresholds would serve a similar purpose for our algorithm.

It is worth emphasizing this effect is dependent on the separating set for the IPA rather than the two vertices themselves. For the graph tested in Figure 3 conditioning on V_5 is sufficient to induce independence between V_4 and V_6 in the unconfounded setting. In the presence of a global confounder, however, we require an additional vertex to be coarsened with V_6 and independent from V_4. As V_6 has no descendants, we must obtain this vertex from $V_0, \ldots, V_2$, requiring an additional vertex to be conditioned on. For this reason, false-positive edges are especially likely to occur at the *end* of our chain.

4.3 Test 3: Varying Density

In our third test, we explore the role that graph density plays in accurately detecting graph adjacency. For this test, we sample random Erdös-Renyi undirected graph structures on 7 vertices and orient them according to a random permutation of the vertices. We vary the probability of edge-occurrence in our graphs from .1 to .9 in .1 increments, sampling 20 graph structures for each. Among these graphs we draw $10,000$ datapoints and study the role of maximum in-degree and total number of edges on the percentage of correctly recovered edges (p-value .0005 again). The results are given in Figure 5.

For $p = 0.1$ to $p = 0.3$ the medians of both true positive and true negative edge reconstruction are at 100%, with the distributions showing the occasional error. As the density of the graph increases, we fail to detect edges (lower blue marks), and incorrectly return edges where there are none (lower orange marks). At high densities, our accuracy for detecting true edges returns to higher levels, but at the cost of occasionally adding false positive edges. The second figure shows that these false positive edges may be due to the larger in-degree of more dense networks. We see very good recovery for networks with limited in-degree, and significantly more error with larger in-degrees.

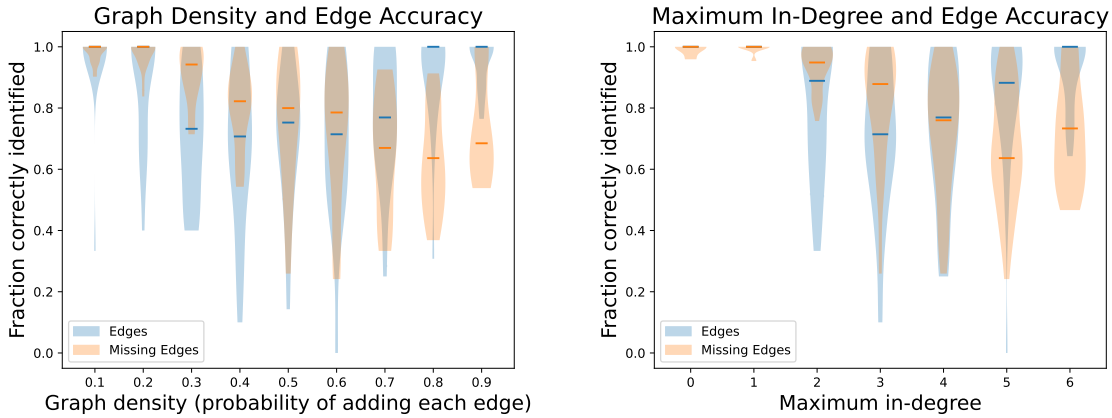


Figure 5: The results of Test 3. Blue gives the percentage of edges that are correctly identified and orange gives the percentage of missing edges that are correctly removed.

5 DISCUSSION

This paper provides an algorithm for causal DAG discovery under confounding by a mixture or latent class. It does so under the setting of Bernoulli observables, which contain only one parameter. While fewer parameters often aid identifiability, this setting sees the opposite because the additional degrees of freedom give a more "rich" pallet for detecting conditional independence (i.e. the probability matrix can be made larger using more moments). The lack of "richness' of Bernoulli random variables brings upon the need for coarsening, which we develop in detail. In principle, our approach can be adapted for easier settings, e.g. discrete observables with higher cardinality alphabets or continuous observables with higher moments. Instead, our result illustrates the relationship between the amount of observed information (given by the number of Bernoulli random variables and their graphical sparsity) and the amount of unobserved information (given by the number of latent classes k).

The majority of our approach relies heavily on the assumption of a known k. This restriction is unavoidable in all mixture-settings. In practice, such a k must be determined through trial and error. If k is too large, then the resulting DAG will be very sparse. If k is too small, then the resulting DAG will be very densely connected. Fortunately, k is sometimes known approximately, e.g. the number of cell-types that are relevant within a dataset. Furthermore, if some observables are *known* to be independent (conditioned on U), then a test of rank can be used to determine k — i.e. the rank of the probability matrix between two conditionally independent variables will be rank k under the previously discussed faithfulness assumptions. We also assume that all latent classes obey the *same* DAG structure. So long as there is enough shared-sparsity from Phase I, Phase II can detect mild class-specific differences in graphical structure.

The results presented in this paper are limited to discrete observables. For continuous data, variables can be discretized (e.g. map all values above the mean to 1 and below the mean to 0). Alternatively, we can modify our probability matrix (Definition 1) to a matrix of moments, e.g. $\boldsymbol{\mathcal{M}}[\mathbf{X}, \mathbf{Y}]_{ij} = E[X_i Y_j]$. The "sets" of variables here can be generated from continuous variables with more degrees of freedom. For example, we could replace a Gaussian X with both $X_1 = X$ and $X_2 = X^2$. All of the results presented here with respect to the rank of probability matrices should transfer to this setting.

Finally, it is worth noting that the test of rank developed in Section 2 may have uses for causal representation learning (CRL), e.g. Squires et al. (2023) uses the rank of probability matrices for causal disentanglement. Additionally, the notion of coarsening may have implications for the abstraction of coarse-grained concepts within CRL problems.

References

Acid, S. and De Campos, L. M. (1996). An algorithm for finding minimum d-separating sets in belief networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 3–10.

Agrawal, R., Squires, C., Prasad, N., and Uhler, C. (2023). The DeCAMFounder: nonlinear causal discovery in the presence of hidden variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, page qkad071.

Allman, E. S., Matias, C., and Rhodes, J. A. (2009). Identifiability of parameters in latent structure models with many observed variables. *Ann. Statist.*, 37(6A):3099–3132.

Anandkumar, A., Hsu, D., Huang, F., and Kakade, S. M. (2012). Learning high-dimensional mixtures of graphical models. *arXiv preprint arXiv:1203.0697*.

Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541.

Cai, R., Huang, Z., Chen, W., Hao, Z., and Zhang, K. (2023). Causal discovery with latent confounders based on higher-order cumulants. In *International conference on machine learning*, pages 3380–3407. PMLR.

Chaudhuri, K. and Rao, S. (2008). Learning mixtures of product distributions using correlations and independence. In *Proc. 21st Ann. Conf. on Learning Theory - COLT*, pages 9–20. Omnipress.

Chen, S. and Moitra, A. (2019). Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proc. 51st Ann. ACM Symp. on Theory of Computing*, pages 869–880.

Cryan, M., Goldberg, L., and Goldberg, P. (2001). Evolutionary trees can be learned in polynomial time in the two state general Markov model. *SIAM J. Comput.*, 31(2):375–397.

Ding, J., d'Orsi, T., Liu, C.-H., Steurer, D., and Tiegel, S. (2022). Fast algorithm for overcomplete order-3 tensor decomposition. In *Conference on Learning Theory*, pages 3741–3799. PMLR.

Feldman, J., O'Donnell, R., and Servedio, R. A. (2008). Learning mixtures of product distributions over discrete domains. *SIAM J. Comput.*, 37(5):1536–1564.

Freund, Y. and Mansour, Y. (1999). Estimating a mixture of two product distributions. In *Proc. 12th Ann. Conf. on Computational Learning Theory*, pages 53–62.

Frot, B., Nandy, P., and Maathuis, M. H. (2019). Robust causal structure learning with some hidden variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 81(3):459–487.

Gordon, S. L., Jahn, E., Mazaheri, B., Rabani, Y., and Schulman, L. J. (2024). Identification of mixtures of discrete product distributions in near-optimal sample and time complexity. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 2071–2091. PMLR.

Gordon, S. L., Mazaheri, B., Rabani, Y., and Schulman, L. J. (2021). Source identification for mixtures of product distributions. In *Proc. 34th Ann. Conf. on Learning Theory - COLT*, volume 134 of *Proc. Machine Learning Research*, pages 2193–2216. PMLR.

Gordon, S. L., Mazaheri, B., Rabani, Y., and Schulman, L. J. (2023). Causal inference despite limited global confounding via mixture models. In *2nd Conference on Causal Learning and Reasoning*.

Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R., and Sellie, L. (1994). On the learnability of discrete distributions. In *Proc. 26th Ann. ACM Symp. on Theory of Computing*, pages 273–282.

Kruskal, J. B. (1977). Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138.

Lähdesmäki, H., Shmulevich, I., and Yli-Harja, O. (2003). On learning gene regulatory networks under the boolean network model. *Machine learning*, 52:147–167.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H.-G. (1990). Independence properties of directed markov fields. *Networks*, 20(5):491–505.

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proc. Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 403–411. Morgan Kaufmann.

Moitra, A. (2016). An almost optimal algorithm for computing nonnegative rank. *SIAM Journal on Computing*, 45(1):156–173.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann.

Pearl, J. (2009). *Causality*. Cambridge university press.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Rabani, Y., Schulman, L. J., and Swamy, C. (2014). Learning mixtures of arbitrary distributions over large discrete domains. In *Proc. 5th Conf. on Innovations in Theoretical Computer Science*, pages 207–224.

Ratsimalahelo, Z. (2001). Rank test based on matrix perturbation theory. Technical report, EERI Research Paper Series.

Richardson, T. and Spirtes, P. (2002). Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030.

Saeed, B., Panigrahi, S., and Uhler, C. (2020). Causal structure discovery from distributions arising from mixtures of dags. In *International Conference on Machine Learning*, pages 8336–8345. PMLR.

Spirtes, P. (2001). An anytime algorithm for causal inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 278–285. PMLR.

Spirtes, P., Glymour, C., Scheines, R., Kauffman, S., Aimale, V., and Wimberly, F. (2000a). Constructing Bayesian network models of gene expression networks from microarray data.

Spirtes, P., Glymour, C., Scheines, R., Spirtes, P., Glymour, C., and Scheines, R. (1993). Discovery algorithms for causally sufficient structures. *Causation, prediction, and search*, pages 103–162.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000b). *Causation, prediction, and search*. MIT press.

Squires, C., Seigal, A., Bhate, S. S., and Uhler, C. (2023). Linear causal disentanglement via interventions. In *International Conference on Machine Learning*, pages 32540–32560. PMLR.

Squires, C. and Uhler, C. (2023). Causal structure learning: A combinatorial perspective. *Foundations of Computational Mathematics*, 23(5):1781–1815.

Strobl, E. V. (2023). Causal discovery with a mixture of dags. *Machine Learning*, 112(11):4201–4225.

Tahmasebi, B., Motahari, S. A., and Maddah-Ali, M. A. (2018). On the identifiability of finite mixtures of finite product measures. (Also in "On the identifiability of parameters in the population stratification problem: A worst-case analysis," Proc. ISIT'18 pp. 1051-1055).

Varici, B., Katz, D., Wei, D., Sattigeri, P., and Tajer, A. (2024). Separability analysis for causal discovery in mixture of dags. *Transactions on Machine Learning Research*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yese]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A D-separation

We will rely on the concepts of **d-separation**, **active paths**, and **separating sets**. Active paths are defined relative to sets of variables to be conditioned on ("conditioning sets").

A key concept in active paths is that of a collider, C, which takes the form $V_1 \to C \leftarrow V_2$ along an undirected path. Undirected paths through unconditioned variables with no colliders are considered active because they can "carry dependence" from one end of the path to the other. If a non-collider along one of these non-collider paths is conditioned on, then the path is inactive.

Collider steps of paths behave in the opposite fashion: paths with an unconditioned collider are inactive, whereas conditioning on a collider or any descendant of that collider "opens" the active path.

When two variables have an active path between them, we say that they are d-connected. If there are no active paths between the variables, we say they are d-separated. A separating set between two variables is a set of variables which, when conditioned on, break all active paths between those variables. While only loosely described here, a full precise definition of d-separation can be found in Pearl (1988) and Pearl (2009) (for a more extensive study).

Pearl (1988) uses structural causal models to justify the *local Markov condition*, which means that d-separation always implies independence and allows DAG structures to be factorized. It is possible that two d-connected variables by chance exhibit some unexpected *statistical* independence. This complication is often assumed away using "faithfulness" (Spirtes et al., 2000b), which ensures that d-connectedness implies statistical dependence. Together, the local Markov condition and faithfulness give a correspondence statistical dependence and the graphical conditions of the causal DAG which can be leveraged for causal structure learning.

The following fact will be useful when building separating sets.

Lemma 10 (Pearl (2009))**.** *If vertices V_i, V_j are nonadjacent in $\mathcal{G}$, either $\mathbf{PA}^{\mathcal{G}}(V_i)$ or $\mathbf{PA}^{\mathcal{G}}(V_j)$ are a valid separating set for V_i, V_j.*

We will often want to bound the cardinality of separating sets relative to the degree bound of the graph (Δ). When dealing with a separating set between two vertices V_i, V_j, Lemma 10 implies a simple upper bound of Δ. Separating sets for *sets* (or coarsenings) of vertices are significantly more complicated because conditioning may d-separate some pairs of vertices while d-connecting others.

To unify the treatment of separating sets, we will make use of **moral graphs**, which can be thought of as undirected equivalents of DAGs (Lauritzen et al., 1990). We will denote the moral graph of $\mathcal{G}$ as $\mathcal{G}^{(m)}$. To transform $\mathcal{G}$ into $\mathcal{G}^{(m)}$, we add edges between all immoralities, i.e. nonadjacent vertices with a common child, sometimes called an unsheilded collider. After this, we change all directed edges to undirected edges.

A very useful fact from Lauritzen et al. (1990) (also Eq. 1 in Acid and De Campos (1996)) is that all separating sets $\boldsymbol{C} \subseteq \boldsymbol{V}$ for $\boldsymbol{S}, \boldsymbol{S}' \subseteq \boldsymbol{V}$ in $\mathcal{G}$ are also separating sets in $(\mathcal{G}[\mathbf{AN}^+(\boldsymbol{S} \cup \boldsymbol{S}' \cup \boldsymbol{C})])^{(m)}$. This transforms complicated active path analysis into simple connectedness arguments on undirected moral graphs (of special subgraphs of $\mathcal{G}$). A convenient consequence of this transformation, which we will use throughout the paper, is Lemma 11.

Lemma 11. *If $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ has degree bound Δ, then the size of a separating set between $\boldsymbol{S}, \boldsymbol{S}' \subseteq \boldsymbol{V}$ is no larger than $\min(|\boldsymbol{S}|, |\boldsymbol{S}'|)\Delta^2$.*

Lemma 11 is a consequence of the maximum increase in the degree of the moral graph.

Proof. A key observation is that the moral graph of a subgraph of $\mathcal{G}$ has no additional edges relative to $\mathcal{G}^{(m)}$. That is, if $\mathcal{G}[\boldsymbol{W}] = (\boldsymbol{W}, \boldsymbol{F})$ is a subgraph of $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ then the corresponding edge-sets of the moral graphs obey $\boldsymbol{F}^{(m)} \subseteq \boldsymbol{E}^{(m)}$ because adding vertices cannot have invalidated any previously contained immoralities.

Abbreviate $(\mathcal{G}[\mathbf{AN}^+(\boldsymbol{S} \cup \boldsymbol{S}' \cup \boldsymbol{C})])^{(m)}$ as $\mathcal{G}_{\boldsymbol{C}}^{(m)}$. Even though we do not know what $\boldsymbol{C}$ is, we know that the 1-neighborhood of $\boldsymbol{S}$ in $\mathcal{G}_{\boldsymbol{C}}^{(m)}$ suffices as a separating set. $\mathcal{G}^{(m)}$ has all of the edges of $\mathcal{G}_{\boldsymbol{C}}^{(m)}$, so

$$\mathbf{NB}_1^{\mathcal{G}_{\boldsymbol{C}}^{(m)}}(\boldsymbol{S}) \subseteq \mathbf{NB}_1^{\mathcal{G}^{(m)}}(\boldsymbol{S}). \tag{12}$$

Note that $\mathbf{NB}_1^{\mathcal{G}^{(m)}}(\boldsymbol{S})$ is not necessarily a separating set for $\boldsymbol{S}, \boldsymbol{S}'$ in $\mathcal{G}^{(m)}$, in fact it may include some vertices in

$\boldsymbol{S}$ itself. However, the size of the separating set is bounded by $|\mathbf{NB}_1^{\mathcal{G}^{(m)}}(\boldsymbol{S})|$, which is no larger than $|\boldsymbol{S}|\Delta^2$. As we chose $\boldsymbol{S}$ arbitrarily, this bound also holds for $\boldsymbol{S}'$. □

Another helpful result from moral graphs is Lemma 12, which will help us when we prove the existence of separating sets.

Lemma 12 (Corollary of Theorem 1 in Acid and De Campos (1996))**.** *For DAG $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, and $\boldsymbol{S}, \boldsymbol{S}' \subseteq \boldsymbol{V}$, separating sets $\boldsymbol{S}, \boldsymbol{S}$ in $(\mathcal{G}[\mathbf{AN}^+(\boldsymbol{S} \cup \boldsymbol{S}')])^{(m)}$ are also separating sets in $\mathcal{G}$.*

B Identifying Mixtures of Discrete Products

The main tool used in Gordon et al. (2023) is a solution to identifying discrete k-mixtures of product distributions (k-**MixProd**) - i.e. $\boldsymbol{X} = X_1, \ldots, X_n$ and latent global confounder or "source" U such that $X_i \perp\!\!\!\perp X_j \mid U$ for all i, j. The key complexity parameter for identifiability is k, the cardinality of the support of U.[7]

At its core, k-MixProd shows how coincidences of multiple independent events reveal information about their confoundedness. Of course, it is possible for U with sufficiently large k to completely control the distribution on $\boldsymbol{X}$. For example, for binary $X_i \in \{0, 1\}$, a cardinality of $k = 2^n$ would be sufficient to assign each binary sequence in $\boldsymbol{X}$ to a latent class in U. Such a powerful U could generate *any* desired probability distribution on $\boldsymbol{X}$ by simply controlling the probability distribution on U. Limiting k, however, limits the space of marginal probability distributions on $\boldsymbol{X}$, eventually giving rise to identifiability.

Under a cardinality bound k on the support of U, Allman et al. (2009) showed that $n \geq \Omega(\log(k))$ is sufficient for the generic identification of k-MixProd. In other words, other than a Lebesgue measure 0 set of exceptions, most instances of k-MixProd have a one-to-one correspondence with their observed statistics (the probability distribution on $\boldsymbol{X}$ marginalized over U) and generating model (up to a set of $k!$ models with permuted labels of U).

For guaranteed identifiability, Tahmasebi et al. (2018) demonstrated that a linear lower bound ($n \geq 2k - 1$), in conjunction with a separation condition in the distributions of $X_i \mid U$, is sufficient to guarantee identifiability. The best known algorithm for identification is given in Gordon et al. (2024), which nearly matches the known lower bounds for sample complexity. This paper will use the result from Allman et al. (2009), but our methods easily extend to stronger identifiability conditions with modifications in the sparsity requirements.

C Further Runtime Discussion

The non-negative rank of a $k + 1 \times k + 1$ matrix (as used for our algorithm) can be solved in time $k^{\mathcal{O}(k^2)}$ (Moitra, 2016). In the absence of non-negative rank tests, Anandkumar et al. (2012) demonstrated that regular rank tests generally work well in place of non-negative rank tests in practice. We will develop a hypothesis test for matrix-rank that requires $\mathcal{O}(k^6)$ operations to invert a $(k + 1)^2 \times (k + 1)^2$ covariance matrix of the estimated elements of our $k + 1 \times k + 1$ matrix.

Our solution also requires solving k-MixProd on 3 variables of cardinality $\mathcal{O}(k)$, which corresponds to decomposing a $\mathcal{O}(k) \times \mathcal{O}(k) \times \mathcal{O}(k)$ tensor into rank 1 components. When the rank of such a tensor is known to be linear in k, this decomposition can be solved in $O(k^{6.05})$ (Ding et al., 2022), though Gordon et al. (2021) showed that the problem generally suffers from instability, with sample complexity exponential in k. This step may be considered optional, as it is used to refine a small number of "false-positive" adjacencies confined to a provably small subset of the DAG.

D Utilizing $\mathcal{G}_1$ to set up k-MixProd

The first step to recovering $\Pr(\boldsymbol{T}_{ij} \mid u)$ will be to select some $\boldsymbol{Z}_{ij}$ and recover $\Pr(\boldsymbol{T}_{ij} \mid u, \boldsymbol{z}_{ij})$ using instances of k-MixProd induced on the conditional probability distribution $\Pr(\boldsymbol{V} \mid \boldsymbol{z}_{ij})$. Recall that k-MixProd requires three independent variables of sufficient cardinality. Hence, we must find $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{T}_{ij}$ which are sufficiently large, and d-separated from each other by $\boldsymbol{Z}_{ij}$ in $\mathcal{G}$. See Figure 6 for an example.

[7] We will refer to the cardinality of the support of discrete random variables as their cardinality.

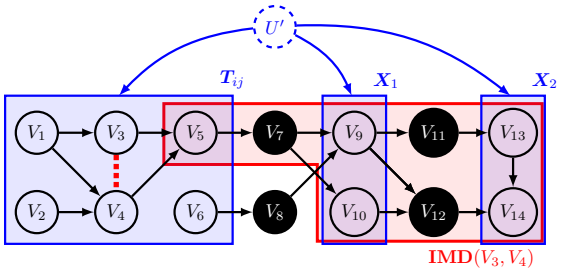


Figure 6: The given graph has an FP edge between V_3 and V_4, indicated by a dashed line, caused by a large set of immoral descendants (shown in red). Conditioning on V_7, V_{11}, V_{12} creates an instance of k-MixProd on $\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2$. Notice that V_7, V_{11}, V_{12} are all in $\mathbf{IMD}(V_3, V_4)$, which means that the $\Pr(\boldsymbol{T}_{ij} \mid V_7, V_{11}, V_{12}, u')$ recovered by k-MixProd will not be sufficient for detecting the FP edge. This obstacle will be solved in Subsection D.2.

Of course we have access to $\mathcal{G}_1$, not $\mathcal{G}$. $\mathcal{G}_1$ contains no orientations[8] and may contain extra false-positive edges. We will need to build a conditioning set $\boldsymbol{Z}_{ij}$ that achieves a guaranteed instance of k-MixProd nonetheless. While Markov boundaries cannot be computed without $\mathcal{G}$, we can easily use $\mathcal{G}_1$ to find a superset that *contains* the Markov bounary of a given vertex.

Lemma 13. *The 2-neighborhood of $\boldsymbol{X} \subseteq \boldsymbol{V}$ in $\mathcal{G}_1$ contains $\mathbf{MB}^{\mathcal{G}}(\boldsymbol{X})$.*

Proof. The distance between $X \in \boldsymbol{X}$, and $V \in \boldsymbol{V}$ in $\mathcal{G}_1$ is less than or equal to the distance in $\mathcal{G}$, because $\boldsymbol{E}_1 \supseteq \boldsymbol{E}$. This means that the 2-neighborhood of $\boldsymbol{X}$ in $\mathcal{G}_1$ includes the 2-neighborhood of $\boldsymbol{X}$ in $\mathcal{G}$. Furthermore because all vertices in $\mathbf{MB}^{\mathcal{G}}(\boldsymbol{X})$ are distance ≤ 2 from at least one $X \in \boldsymbol{X}$, we have that $\mathbf{MB}^{\mathcal{G}}(\boldsymbol{X})$ is contained in the 2-neighborhood of $\boldsymbol{X}$ in $\mathcal{G}_1$. □

The sparsity of $\mathcal{G}_1$ will dictate the number of necessary vertices to successfully set up k-MixProd. We will want to limit the size of $\boldsymbol{Z}_{ij}$ as much as possible. Fortunately, the bounds on the size of $\boldsymbol{H}$ mean that *most* of $\mathcal{G}_1$ is degree bounded by Δ. We will avoid large degree vertices in $\boldsymbol{H}$ by strategically selecting $\boldsymbol{X}_1, \boldsymbol{X}_2$ with the smallest 2-neighborhoods. $\boldsymbol{Z}_{ij} = \mathbf{MB}(\boldsymbol{X}_1) \cup \mathbf{MB}(\boldsymbol{X}_2)$ will be sufficient to d-separate all three vertices, so we need not worry about potentially large degree in $\boldsymbol{T}_{ij}$. This process is described by Algorithm 2.

Algorithm 2: Formation of k-MixProd Instances

Input: Two vertices $V_i, V_j \in \boldsymbol{V}$ and $\mathcal{G}_1 = (\boldsymbol{V}, \boldsymbol{E}_1)$ from the output of Algorithm 1.
Output: $\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2$ and $\boldsymbol{Z}_{ij}$ such that $\boldsymbol{T}_{ij} \perp\!\!\!\perp_d \boldsymbol{X}_1 \perp\!\!\!\perp_d \boldsymbol{X}_2 \mid \boldsymbol{Z}_{ij}$.
Let $\boldsymbol{T}_{ij} = \{V_i, V_j\} \cup \mathbf{NB}_1^{\mathcal{G}_1}(V_i) \cup \mathbf{NB}_1^{\mathcal{G}_1}(V_j)$.
$\boldsymbol{V}' \leftarrow \boldsymbol{V} \setminus (\boldsymbol{T}_{ij} \cup \mathbf{NB}_2(\boldsymbol{T}_{ij}))$
$\boldsymbol{X}_1, \boldsymbol{X}_2$ all begin as empty sets.
while $2^{|\boldsymbol{X}_1|} + 2^{|\boldsymbol{X}_2|} < 2k + 2 - \min(k, 2^{|\boldsymbol{T}_{ij}|})$ **do**
  Add $V \in \boldsymbol{V}' \setminus (\mathbf{NB}_2(\boldsymbol{X}_2) \cup \boldsymbol{X}_2)$ to $\boldsymbol{X}_1$.
  Add $V \in \boldsymbol{V}' \setminus (\mathbf{NB}_2(\boldsymbol{X}_1) \cup \boldsymbol{X}_1)$ to $\boldsymbol{X}_2$.
end
$\boldsymbol{Z}_{ij} \leftarrow \mathbf{NB}_2(\boldsymbol{X}_1) \cup \mathbf{NB}_2(\boldsymbol{X}_2)$.

Lemma 14. *Algorithm 2 terminates successfully (without running out of vertices in $\boldsymbol{V}'$) with $\Omega(\Delta^3 \log(k))$ vertices.*

Proof. The algorithm designates vertices in $\boldsymbol{V}$ into the following sets and succeeds so long as those sets are disjoint.

1. $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$
2. $\mathbf{NB}_2(\boldsymbol{X}_1), \mathbf{NB}_2(\boldsymbol{X}_2)$

[8] It is, in principle, possible to orient immoralities within $\mathcal{G}_1$ at this stage, but this gives no complexity improvements.

3. $\boldsymbol{T}_{ij}$
4. $\mathbf{NB}_1(\boldsymbol{T}_{ij})$

$|\boldsymbol{T}_{ij}| \geq 2$ and $k \geq 2$, so the number of vertices added to $\boldsymbol{X}_1, \boldsymbol{X}_2$ in the loop of Algorithm 2 is at most $\lceil \lg(2k+2-2) \rceil < \lg(k) + 2$. To bound the 2-neighborhood, we notice that we cannot easily apply our degree bound of Δ because $\boldsymbol{H}$ could be a clique in $\mathcal{G}_1$ (from FP edges). Instead, we bound

$$|\mathbf{NB}_2(\boldsymbol{X}_1) \cup \mathbf{NB}_2(\boldsymbol{X}_2) \cup \boldsymbol{X}_1 \cup \boldsymbol{X}_2| \leq \Delta^2 |\boldsymbol{X}_1 \cup \boldsymbol{X}_2| + \Delta |\boldsymbol{H}| \tag{13}$$

because the distance 1 neighborhood could include all of $\boldsymbol{H}$ but all additional neighborhoods are bounded by Δ. $|\boldsymbol{H}|$ is $\mathcal{O}(\Delta^2 \log(k))$ by Observation 2, so this bound is $\mathcal{O}(\Delta^3 \log(k))$.

The size of $\boldsymbol{T}_{ij}$ is the largest when including V_i or V_j in $\boldsymbol{H}$, for which $\mathbf{NB}_1(V_i)$ could be all of $\boldsymbol{H}$ and $\mathbf{NB}(V_i)$ then necessarily falls outside of $\boldsymbol{H}$. This worst case gives

$$|\boldsymbol{T}_{ij}| \leq |\boldsymbol{H}| + \Delta^2, \tag{14}$$

which is $\mathcal{O}(\Delta^2 \log(k))$. Expanding to the 1 neighborhood picks up another factor of Δ, bringing us again to $\mathcal{O}(\Delta^3 \log(k))$.

□

D.1 Aligning multiple k-MixProd runs

k-MixProd distributions are symmetric with respect to the $k!$ permutations on the label of their source. For this reason, there is no guarantee that multiple calls to a k-MixProd solver will return the same permutation of source labels.

To solve this, Gordon et al. (2023) noticed that any two solutions to k-MixProd problems that share the same conditional probability distribution for at least one "alignment variable" can be "aligned" by permuting the source labels until the distributions on that variable match up. We will only need alignment along runs for different assignments to each $\boldsymbol{Z}_{ij}$, used in the next section. Explicitly, two assignments $\boldsymbol{z}_{ij}$ and $\boldsymbol{z}'_{ij}$, need least one $\boldsymbol{X}^* \in \{\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2\}$ such that $\mathbf{mb}_{\boldsymbol{z}_{ij}}(\boldsymbol{X}^*)$ and $\mathbf{mb}_{\boldsymbol{z}'_{ij}}(\boldsymbol{X}^*)$ are the same, in order for alignability to be satisfied.

To align sets of k-MixProd results which are not all pairwise alignable, Gordon et al. (2023) introduced the concept of an "alignable set of runs" for which chains of alignable pairs create allow alignability.

Lemma 15. *The set of k-MixProd instances on the same $\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2$ with all possible assignments $\boldsymbol{z}_{ij}$ to $\boldsymbol{Z}_{ij}$ is alignable.*

Proof. Any two runs with assignments $\boldsymbol{z}_{ij}$ and $\boldsymbol{z}'_{ij}$ that differ in their assignment to only one variable are alignable. Therefore, any two non-alignable runs can be aligned using a chain of Hamming-distance one alignments. □

D.2 Recovering the unconditioned within-source distribution

After all our calls to the k-MixProd oracle, we have access to $\Pr(\boldsymbol{T}_{ij} \mid u, \boldsymbol{z}_{ij})$ and $\Pr(u \mid \boldsymbol{z}_{ij})$ for every assignment $\boldsymbol{z}_{ij}$ and u. $\Pr(\boldsymbol{T}_{ij} \mid u, \boldsymbol{z}_{ij})$ is insufficient to determine the adjacency of V_i, V_j because $\boldsymbol{Z}_{ij}$ may contain vertices in the immoral descendants of V_i, V_j, prohibiting the discovery of a separating set within $\boldsymbol{T}_{ij}$.

Instead, we must recover $\Pr(\boldsymbol{T}_{ij} \mid u)$, which is not conditioned on $\boldsymbol{Z}_{ij}$. To do this, we can apply the law of total probability over all possible assignments to $\boldsymbol{Z}_{ij}$.

$$\Pr(\boldsymbol{T}_{ij} \mid u) = \sum_{\boldsymbol{z}_{ij}} \Pr(\boldsymbol{z}_{ij} \mid u) \Pr(\boldsymbol{T}_{ij} \mid \boldsymbol{z}_{ij}, u) \tag{15}$$

We can obtain $\Pr(\boldsymbol{z}_{ij} \mid u)$ by using Bayes rule on the k-MixProd output, $\Pr(u \mid \boldsymbol{z}_{ij})$.

$$\Pr(\boldsymbol{z}_{ij} \mid u) = \frac{\Pr(u \mid \boldsymbol{z}_{ij}) \Pr(\boldsymbol{z}_{ij})}{\Pr(u)}. \tag{16}$$

$\Pr(\boldsymbol{z}_{ij})$ can be obtained by counting the frequency of $\boldsymbol{z}_{ij}$ in the data. In addition, $\Pr(u) = \sum_{\boldsymbol{z}_{ij}} \Pr(\boldsymbol{z}_{ij}) \Pr(u \mid \boldsymbol{z}_{ij})$ is computable by the law of total probability after the runs for each assignment $\boldsymbol{z}_{ij}$, have been aligned. Equivalently, we can normalize such that $\sum_{\boldsymbol{z}_{ij}} \Pr(\boldsymbol{z}_{ij} \mid u) = 1$.

Lemma 16. *We can compute* $\Pr(\boldsymbol{T}_{ij} \mid u)$ *using known quantities,*

$$\Pr(\boldsymbol{T}_{ij} \mid u) = \frac{\sum_{\boldsymbol{z}_{ij}} \Pr(u \mid \boldsymbol{z}_{ij}) \Pr(\boldsymbol{z}_{ij}) \Pr(\boldsymbol{T}_{ij} \mid \boldsymbol{z}_{ij}, u)}{\sum_{\boldsymbol{z}_{ij}} \Pr(\boldsymbol{z}_{ij}) \Pr(u \mid \boldsymbol{z}_{ij})}.$$

$\Pr(\boldsymbol{T}_{ij} \mid u)$ is a completely deconfounded distribution on which we can run the PC-algorithm. The full procedure is given in Algorithm 3, in which we use Algorithm 2 followed by alignment and Lemma 16 in order to remove all of the false-positive edges from $\mathcal{G}_1$.

Algorithm 3: Phase II: Detection and correction of FP edges.

Input: $\Pr(\boldsymbol{V})$ marginalized over U, a black box solver for k-MixProd, and $\mathcal{G}_1 = (\boldsymbol{V}, \boldsymbol{E}_1)$ from the output of Algorithm 1.
Output: $\mathcal{G}_2 = (\boldsymbol{V}, \boldsymbol{E}_2)$, an undirected skeleton of $\mathcal{G}$ and separating sets for nonadjacencies (vertices not in $\boldsymbol{E}_2$) .
Start with $\boldsymbol{E}_2 \leftarrow \boldsymbol{E}_1$.
for *each* $\{V_i, V_j\} \in \boldsymbol{E}_1$ **do**
  Retrieve $\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{Z}_{ij}$ from Algorithm 2.
  for *each assignment* $\boldsymbol{z}_{ij}$ **do**
    Run the k-MixProd solver on $\boldsymbol{T}_{ij}, \boldsymbol{X}_1, \boldsymbol{X}_2$ on $\Pr(\boldsymbol{V} \mid \boldsymbol{z}_{ij})$.
  end
  Perform alignment of the $2^{\boldsymbol{z}_{ij}}$ runs to retrieve $\Pr(\boldsymbol{T}_{ij} \mid \boldsymbol{Z}_{ij}, U)$.
  Calculate $\Pr(\boldsymbol{T}_{ij} \mid u)$ for every u using Lemma 16.
  Run PC or any other structure learning algorithm on $\Pr(\boldsymbol{T}_{ij} \mid u)$ to find a separating set $\boldsymbol{C}_{ij}$ (or verify adjacency) for V_i, V_j. If $V_i \perp\!\!\!\perp V_j \mid \boldsymbol{C}_{ij}, u$ for all u, remove $\{V_i, V_j\}$ from $\boldsymbol{E}_2$ and store $\boldsymbol{C}_{ij}$.
end

Lemma 17. *Algorithm 3 requires solving* k*-MixProd* $\mathcal{O}(k|\boldsymbol{E}|2^{\Delta^2})$ *times.*

Proof. This algorithm requires running k-MixProd for every possible assignment to the conditioning set $\boldsymbol{D}_{ij}$, for which we have $|\boldsymbol{D}_{ij}| \leq (\lg(k) + 2)\Delta^2$ total binary variables. This gives an upper bound of $2k2^{\Delta^2}$ runs of k-MixProd for each edge. □

E Experiment Details

E.1 Deferred Experimental Details

Experiments were run on a Macbook Air with an M1 processor. Running each test can be done using terminal commands in the attached folder.

```
python3 test_1_rank_experiments.py
python3 test_2_learn_structure.py
python3 test_3_change_density.py
```

Test 1 takes a few minutes to run, while test 2 and 3 can take 1-2 hours depending on how many runs are made to capture statistical significance.

E.2 Structural Equation Setup

SCMs are made up of a graphical structure and accompanying structural equations. We focus our tests primarily on varying the graphical structure, using a standard set of structural equations on these graphs. Our U are

generated using a fair coin ($k = 2$), and all other vertices are Bernoulli random variables with bias p_V determined by V's parents (including U):

$$p_V = \frac{1 + \sum_{W \in \mathbf{PA}^{\mathcal{G}'}(V)} W}{|\mathbf{PA}^{\mathcal{G}'}(V)| + 2}. \tag{17}$$

Structural equations of this form have a reasonable strength between vertices that is decreased relative to in-degree.

F Deferred Proofs

F.1 Proof of Lemma 6

To prove this lemma we first define the entire set of descendants $\boldsymbol{D}_{ij} \supseteq \mathbf{IMD}(V_i, V_j)$.

Definition 7. Let the set $\boldsymbol{D}_{ij} := \mathbf{DE}(V_i, V_j)$ be the descendants of both vertices and $\boldsymbol{A}_{ij} := \boldsymbol{V} \setminus \boldsymbol{D}_{ij}$.

Recall that a separating set for any two sets of variables exists as a separating set in the moral graph of their ancestors (Lemma 12). By restricting our focus to $\boldsymbol{S}_i, \boldsymbol{S}_j \subseteq \boldsymbol{A}_{ij}$ will also have $\mathbf{AN}^+(\boldsymbol{S}_i^+, \boldsymbol{S}_j^+) \subseteq \boldsymbol{A}_{ij}$ which guarantees that our separating sets will not overlap with $\mathbf{IMD}(V_i, V_j)$. Lemma 18 will tell us how large we need $\boldsymbol{A}_{ij}$ to be in order to be guaranteed an IPA.

Lemma 18. *An IPA $\boldsymbol{S}_i^+, \boldsymbol{S}_j^+$ for $V_i, V_j \in \boldsymbol{V}$ exists so long as $|\boldsymbol{A}_{ij}| \geq (2 + \Delta^2)(\lceil \lg(k) \rceil + 1)$.*

Proof. We can form $\boldsymbol{S}_i^+$ out of V_i and $\lceil \lg(k) \rceil$ arbitrary other vertices from $\boldsymbol{A}_{ij}$. Now, let the separating set be $\boldsymbol{C} := \mathbf{MB}^{\mathcal{G}[\boldsymbol{A}_{ij}]}(\boldsymbol{S}_i^+)$ and note that $\boldsymbol{C}$ d-separates $\boldsymbol{S}_i^+$ from all other elements of $\boldsymbol{A}_{ij}$. Since we know $|C| \leq \Delta^2(\lceil \lg(k) \rceil + 1)$, we have at least $\lceil \lg(k) \rceil$ vertices in $\boldsymbol{A}_{ij}$ left to join with V_j and make $\boldsymbol{S}_j^+$. □

We are now ready to prove Lemma 6.

Proof. A convenient consequence of Lemma 18 is that it guarantees the existance of IPAs everywhere except within a small subset of vertices. Let $\overline{\mathbf{DE}}(V) := \boldsymbol{V} \setminus \mathbf{DE}(V)$ be the "non-descendants" of V. Note that $\boldsymbol{A}_{ij} = \overline{\mathbf{DE}}(V_i) \cup \overline{\mathbf{DE}}(V_j)$. This implies that

$$|\boldsymbol{A}_{ij}| \geq \max(|\overline{\mathbf{DE}}(V_i)|, |\overline{\mathbf{DE}}(V_j)|). \tag{18}$$

Hence, so long as at least one vertex has enough non-descendants, $\boldsymbol{A}_{ij}$ will be large enough to form an IPA. This set of vertices with enough non-descendants corresponds to the complement of the early vertices. □

F.2 Proof of Lemma 2

Proof. We will drop the conditioning on $\boldsymbol{c}$ in this proof for simplicity. Consider the sum

$$\sigma_j := \sum_{i=1}^{j} \Pr(u_i) \boldsymbol{\mathcal{M}}[X, Y \mid u_i], \tag{19}$$

and note that $\sigma_k = \boldsymbol{\mathcal{M}}[X, Y]$. Faithfulness with respect to $\mathcal{G}'$ tells us that there is some assignment, which we call u_1 wlog, such that $X \not\perp\!\!\!\perp Y \mid u_1$. Hence $\mathrm{rk}_+(\sigma_1) > 1$.

Now, we show inductively that $\mathrm{rk}_+(\sigma_i) = \mathrm{rk}_+(\sigma_{i-1}) + 1$ is a measure 1 event for $i = 1, \ldots, k$. Denote $\boldsymbol{\mathcal{M}}[X, Y \mid u_i] = v_i w_i^\top$ with column space v_i drawn from a subspace with non-zero measure on $\mathbb{R}^n$. The column space of σ_{i-1} is rank $\leq i - 1 < m$, so it has measure zero on $\mathbb{R}^m$. Hence, v_i being in the column space of σ_{i-1} is a measure 0 event. We conclude that $\mathrm{rk}_+(\sigma_{i-1} + \boldsymbol{\mathcal{M}}^{u_i}[X, Y]) = \mathrm{rk}_+(\sigma_{i-1}) + 1$ with measure 1. Inducting on i gives $\mathrm{rk}_+(\sigma_k) > k$ with measure 1. □

F.3 Proof of Lemma 5

Proof. Suppose for contradiction that some vertex $B \in \mathbf{IMD}(V_i, V_j) \cap \boldsymbol{S}_j^+$. $B \in \mathbf{IMD}(V_i, V_j)$ implies that there is a directed path $\boldsymbol{P} \subseteq \mathbf{IMD}(V_i, V_j)$ from V_i to B. By the definition of an IPA, $B \in \boldsymbol{S}_j^+$ means that there must be some $\boldsymbol{C}$ with $B \perp\!\!\!\perp_d V_i \mid \boldsymbol{C}$. We conclude that $\boldsymbol{C}$ must contain some $C \in \boldsymbol{P}$ in order to block $\boldsymbol{P}$ from being an active path. However, this also means that $C \in \mathbf{IMD}(V_i, V_j)$, which contradicts Observation 1. The same argument holds for $B \in \boldsymbol{S}_i^+$. □

F.4 Proof of Lemma 8

Proof. $\boldsymbol{T}_{ij}$ contains both $\mathbf{PA}(V_i)$ and $\mathbf{PA}(V_j)$, so Lemma 10 tells us that we contain a separating set. □