Extended version of paper accepted to the Proceedings of the Thirty-fourth AAAI Conference on Artificial Intelligence (AAAI-2020).

TECHNICAL REPORT
R-491-L
December 2019

A Simultaneous Discover-Identify Approach to Causal Inference in Linear Models

Chi Zhang,[1] **Bryant Chen,**[2] **Judea Pearl**[1]
[1]Department of Computer Science, University of California, Los Angeles, California, USA.
[2]Brex, San Francisco, California, USA*
zccc@cs.ucla.edu, bryant@brex.com, judea@cs.ucla.edu

Abstract

Modern causal analysis involves two major tasks, discovery and identification. The first aims to learn a causal structure compatible with the available data, the second leverages that structure to estimate causal effects. Rather than performing the two tasks in tandem, as is usually done in the literature, we propose a symbiotic approach in which the two are performed simultaneously for mutual benefit; information gained through identification helps causal discovery and vice versa. This approach enables the usage of Verma constraints, which remain dormant in constraint-based methods of discovery, and permit us to learn more complete structures, hence identify a larger set of causal effects than previously achievable with standard methods.

Introduction

Learning causal relationships is one of the most ambitious goals of scientific inquiry. Controlled randomized experiments can sometimes be used to both learn the causal structure among variables, as well as the size of the causal effects. However, such experiments are often too expensive or even impossible to conduct. Instead, learning causal relationships from observational data can be attempted; first by learning the causal structure from observational data, called discovery, and then identifying causal effects from the observational data and the partially specified causal structure. This paper introduces a method of performing both tasks simultaneously in a mutually beneficial way.

Many algorithms have been developed for causal discovery. These algorithms generally fall into two categories: score-based algorithms (e.g., Heckerman, Geiger, and Chickering (1995), Chickering (2002), Shpitser et al. (2012), Fast GES by Ramsey et al. (2017)) and constraint-based algorithms (e.g., IC algorithm by Verma and Pearl (1991), PC algorithm by Spirtes et al. (2000), FCI algorithm first by Spirtes et al. (2000) and improved by Zhang (2008)). Constraint-based algorithms aim to discover a class of graphs that encode the same constraints as those implied by the data. They perform a sequence of conditional independence tests to efficiently rule out impossible edge configurations. Constraint-based algorithms have significant advantage over score-based algorithms in that they are able to learn entire equivalence classes of models with unobserved variables, often called "semi-Markovian."

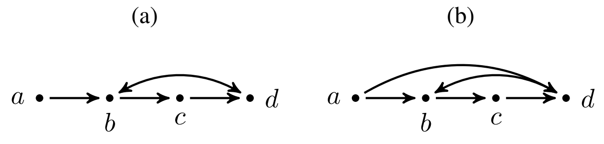

Figure 1: (a) a DAG where $\sigma_{ad}/\sigma_{ac} = \sigma_{cd\cdot b}$ (b) a DAG where $\sigma_{ad}/\sigma_{ac} \neq \sigma_{cd\cdot b}$

Existing constraint-based algorithms use conditional independences between model variables to learn the causal structure. However, since there are usually many structures consistent with any given set of conditional independences, these algorithms are only able to produce large equivalence classes of possible structures.

Verma constraints (Verma and Pearl 1991) impose additional constraints on the probability distribution beyond conditional independences, and thus allow the discovery of additional structures. For example, though Figures 1(a) and 1(b) are conditional-independence-equivalent, they imply different Verma constraints. 1(a) implies the Verma constraint $\sigma_{ad}/\sigma_{ac} = \sigma_{cd\cdot b}$, while 1(b) does not (hint: σ_{ad} is equal to the product of the three coefficients on $a \to b$, $b \to c$, and $c \to d$ in 1(a), while σ_{ad} is equal to the same product plus the coefficient on $a \to d$ in 1(b)). Several algorithms for deriving Verma constraints from a model's structure have been developed, including algorithms by Tian and Pearl (2002) and Shpitser and Pearl (2008) for non-parametric models and algorithms by Chen (2016) and Chen, Kumor, and Bareinboim (2017) for linear models. These algorithms can be used to derive Verma constraints from a hypothesized model structure and test it. However, it is not clear how to systematically find such constraints from data to discover the model's structure. Indeed, no constraint-based method for learning causal structures from Verma constraints currently exists in the literature.

*Much of the work by Chen was conducted while at IBM Research AI.

Fortunately, under the linear setting, a useful tool, called auxiliary variables (AVs) (Chen, Pearl, and Bareinboim 2015), can be used to reduce the problem of finding Verma constraints to one of finding conditional independences. AVs are constructed by subtracting known direct effects–if the coefficient from variables x to y, β, is known, an AV $y^* = y - \beta x$ is constructed by subtracting βx from y. Now, y^* may be conditionally independent of some variables that y was dependent of. This conditional independence, which is equivalent to a Verma constraint over the original model variables, can then be used to learn more of the structure.

Constructing AVs without prior knowledge requires identification of direct effects. Thus, in order to use AVs in causal discovery, we need a method to identify direct effects from an incomplete causal structure. To this end, we generalize the qID algorithm of Chen, Kumor, and Bareinboim (2017) for partially specified causal structures. Combining this algorithm with AVs, we are able to iteratively identify causal effects on an incomplete structure, construct AVs, and learn more of the structure. Each identification step enables the construction of more AVs, which helps to learn more of the structure. Similarly, each causal discovery step learns more of the structure, which helps to identify more causal effects.

In summary, we introduce a simultaneous discover-identify algorithm, where each task is performed to the other's benefit. To our knowledge, this algorithm is the first constraint-based causal discovery algorithm to use Verma constraints, and the first identification algorithm for partially specified linear causal models[1]. Lastly, we demonstrate that in high percentages of simulated cases, our method provides noticeable improvements in recovering random graph structures while guaranteeing correctness.

Preliminaries

The causal directed acyclic graph (DAG) of a structural equation model (SEM) is a graph, $G = (V, E)$, where V are nodes representing model variables and E are edges representing causal relations between two nodes. An edge in a causal graph can be directed ($\rightarrow$), bidirected ($\leftrightarrow$), or both. Directed edges encode the direction of causality, i.e., if x_i is in the structural equation that determines x_j, an edge is drawn from x_i to x_j. Each directed edge, therefore, is associated with a coefficient in the SEM, which we often refer to as its edge coefficient. A bidirected edge between two nodes indicates their corresponding error terms may be statistically dependent, while the lack of a bidirected edge indicates the error terms are independent. If both a directed edge and a bidirected edge exist between two nodes, it indicates one variable is directly affecting the other and they are both affected by an unobserved confounder at the same time.

In the following sections, we use standard graph terminology, where $He(E)$ denotes the heads of a set of directed edges, E, $Ta(E)$ denotes the tails, and for a node v, the set of edges for which $He(E) = v$ is denoted $Inc(v)$. We also restrict our attention to semi-Markovian linear causal models (Pearl 2009), models that are acyclic, that may contain latent confounders, and for which the causal relationships are linear. Lastly, we use the term *full DAG*[2] to refer to a standard causal graph, where the orientation of every edge is specified, and the term *true DAG* to refer to the full DAG that represents the underlying data generating process.

We use $\sigma_{xy \cdot W}$ to denote the partial covariance between two variables, x and y, given a set of variables, W. We also assume without loss of generality that the model variables have been standardized to mean 0 and variance 1.

Patterns

When learning a causal structure, constraints on the covariances between variables (conditional independence and Verma constraints) are generally insufficient to define a single DAG. Instead, they are only able to narrow down the set of possible structures to a large equivalence class. *Patterns* are motivated by the need to define a graph structure to represent such a class. Using causal discovery algorithms, we aim to learn a pattern that represents an equivalence class of graphs consistent with the constraints provided.

Similar concepts were previously defined in the literature, including *patterns* in Verma and Pearl (1991) (who first used the term "pattern") and *partial ancestral graphs (PAGs)* in Richardson (1996). PAGs are used to represent equivalence classes of maximal ancestral graphs (MAGs) (Richardson and Spirtes 2002). MAGs are abstractions of DAGs that keep only the conditional independence and ancestral relationships. More formally, MAGs are *maximal* and *ancestral*. There is an edge between two nodes a and b in the MAG if and only if there exists no set that can separate a and b in the DAG (maximal), and $a \rightarrow b$ is in the MAG if and only if a is an ancestor of b in the DAG (ancestral).

PAGs are useful for causal discovery algorithms such as FCI, which aims to recover a MAG. However, PAGs cannot distinguish between different DAGs sharing the same MAG abstraction, and therefore cannot distinguish between different DAGs sharing ancestral relationships and conditional independence constraints but have different Verma constraints. For example, in Figure 2(a), e and f are not conditionally independent. Therefore, a DAG with e and f connected and a DAG without them connected share the same MAG, even though they imply different Verma constraints. Since our method will enable us to distinguish between such structures, we need a more precise representation without the "maximal" or "ancestral" requirement.

Definition 1. *A* pattern*,* $P = (V, E)$*, is a graph whose edges contain three possible types of edge marks:* arrowheads, tails*, and* circles *(and hence four kinds of edges*[3][4]*:*

[1]Non-parametric algorithms can, of course, also be applied to linear models, but they are significantly weaker due to their inability to leverage the linearity assumption.

[2]We emphasize a DAG being "full" to distinguish it from a "pattern", which is a partially specified DAG. Note that we are not referring to a complete DAG, which is a DAG where all edges are present.

[3]These edge markings are adopted from PAGs.

[4]We assume no selection bias. The other two kinds of edges in PAGs defined in Zhang (2008), $-$ and $\circ\!-$, which only appear when there is selection bias, are thus not included.

TECHNICAL REPORT
R-490
December 2019

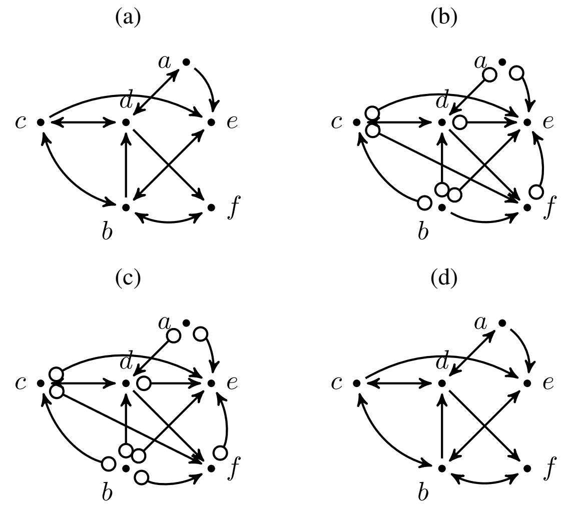

Figure 2: (a) underlying causal relationships (b) pattern learned by FCI (c) pattern learned by modified FCI, which does not learn inconsistent tails such as $b \to f$ in (b) (d) pattern learned by our method, LCDI

$\to, \leftrightarrow,$ ∘–∘, ∘→*). The edges denote possible causal relations between two nodes.*

Each pattern P can be used to represent (formally defined below) a class of full DAGs, denoted $[G]$. A circle mark indicates uncertainty, i.e., it is possible that the edge mark is arrowhead for some members in $[G]$, tail for some members, and both (having both a directed edge and a bidirected edge in between) for others. An edge mark is said to be *invariant* if the mark is the same in all members of $[G]$ (Zhang 2008).

Definition 2. *A pattern $P = \{V_P, E_P\}$ is defined to* represent *a class of full DAGs [G], if for each member $G = \{V_G, E_G\}$ in [G], (i) $V_P = V_G$, and (ii) each $e \in E_P$ is either extraneous (the two same nodes in G are not connected by an edge), or the arrowhead and tail edge marks on e are invariant in [G].*

In Figure 2, 2(a) is both in the class represented by 2(c) and the class represented by 2(d). This is seen by checking each edge. For example, a ∘→ d in 2(c) has an arrowhead at d and a circle at a, so the DAGs in the class it represents must have an arrowhead at d but can have anything at a. $a \leftrightarrow d$ in 2(a) satisfies the requirement. e ←∘f in 2(c) is extraneous since it is not in 2(a), which also satisfies Definition 2. Note that from a causal discovery perspective, learning 2(d) is preferable to learning 2(b) since the class of graphs represented by 2(d) is a subset of the class represented by 2(b).

Edge Orientation Rules Based on Verma Constraints

In this section, we first review how conditional independence constraints are used by current causal discovery algorithms before describing how we extend these algorithms by incorporating Verma constraints. First, conditional independence constraints are found by checking the partial correlation between each pair of variables given all subsets of other variables. Assuming faithfulness, each vanishing partial correlation indicates there is no edge between the pair of variables, and the conditioning set contains the variables that, when conditioned on, d-separates the pair in the graph. Therefore, we are able to rule out the edge orientations that leave an unblocked path between the pair.

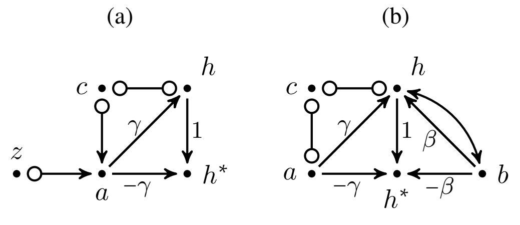

Figure 3: (a) an AV in a pattern (b) an AV generated by two variables in a pattern

Current constraint-based causal discovery methods use only conditional independence constraints because conditional independence constraints can be easily found, and their implications on the structure is clear. In contrast, Verma constraints are hard to find without the aid of a full DAG, because their functional forms are far less restricted. Additionally, it is also not always clear how they constrain the graph structure.

However, by identifying causal effects and constructing AVs, we may generate new conditional independences between the AVs and the original model variables. These conditional independences, which we describe as *AV conditional independence constraints*, are Verma constraints. Thus, by using AVs, we can reduce the problem of finding and using Verma constraints for causal discovery to a problem of finding and using conditional independences–a problem that is already well understood.

Intuitively, AVs negate the effect of problematic paths by subtracting out known direct effects. Let P^{E+} denote the augmented pattern with AVs generated using edges E added. In Figure 3(a), if the direct effect of a on h, γ, is identified, an AV, $h^* = h - \gamma a$ can be generated, giving P^{ah+}. Similarly, in Figure 3(b), an AV $h^* = h - \gamma a - \beta b$ can be generated using edges $a \to h, b \to h$, giving $P^{\{ah,bh\}+}$. Generating AVs from patterns will allow us to search the data for new conditional independences involving the AVs and learn more of the model's structure. These conditional independences correspond to Verma constraints over the original model variables as explained in the following lemma.

Lemma 1. *Given an AV, $z^* = z - \Sigma_i e_i t_i$, the conditional independence constraint, $\sigma_{az^* \cdot S} = 0$, is equivalent to the Verma constraint, $\sigma_{az \cdot S} - \Sigma_i e_i \sigma_{at_i \cdot S} = 0$, where S is a set of variables. Furthermore, this Verma constraint cannot, in general, be represented as a conditional independence constraint over the original model variables, V.*

Lemma 1 makes it possible to easily find Verma constraints that are AV conditional independence constraints[5].

[5]There might exist other types of Verma constraints that cannot be expressed as AV conditional independence constraints. Those constraints are outside the scope of this paper.

TECHNICAL REPORT
R-490
December 2019

We can simply check whether each AV can be made conditionally independent of other AVs or the original model variables. Similar to traditional conditional independence constraints, AV conditional independence constraints refine the structure by limiting edge marks to those that block all the paths between the independent variables in the augmented pattern. Furthermore, this is in fact equivalent to blocking paths in the pattern without the edges used to generate the AVs, as stated in the following corollary, derived from Theorem 1 in Chen, Kumor, and Bareinboim (2017).

Corollary 1. *Given a linear pattern P representing $[G]$, where $E \subset Inc(z)$ is a set of edges whose coefficient values are known, if $(W \cup \{y\}) \cap (V \setminus NDe^*(z)) = \varnothing$, and G_{E-} represents the graph G with the edges for E removed, then $\sigma_{z^* y \cdot W} = 0$ only if $(z \perp\!\!\!\perp y | W)_{G_{E-}}$ for all G in $[G]$.*

See the pattern $P^{\{a\to h\}+}$ in Figure 3(a), where the edge coefficient on $a \to h$, γ, is identified (using z as an instrumental variable) and the AV, $h^* = h - a\gamma$, is constructed. If $\exists S_{ah^*}$, $\sigma_{ah^* \cdot S_{ah^*}} = 0$, then Corollary 1 implies for all G in $[G]$ represented by P, $(h \perp\!\!\!\perp a | S_{ah^*})_{G_{\{a\to h\}-}}$. On the other hand, no information can be obtained using traditional conditional independence constraints. $\nexists S_{ah}, a \perp\!\!\!\perp h | S_{ah}$ since a and h are directly connected by an edge.

Assuming a generalized version of faithfulness[6], the only path between a and h in $G_{\{a\to h\}-}$, $a \leftarrow\!\circ\, c \circ\!\!-\!\!\circ h$, must be blocked by S_{ah^*}. If, for example, $c \notin S_{ah^*}$, c must be a collider in any G, and we can thus orient $a \leftrightarrow c \leftarrow\!\circ h$ in P.

To formally construct the edge orientation rules, we need to characterize such a relationship between two variables like a and h that are not necessarily non-adjacent in the original pattern, but are non-adjacent in P_{E-} due to the independence between their AVs. We also need to characterize variables remaining to be adjacent in P_{E-} such that the adjacencies of all variables are with respect to the same pattern, with or without E virtually removed, to ensure consistent edge orientations. We describe such adjacency relationships in the following definition.

Definition 3. *Given an AV-augmented pattern P^{E+} where AVs, $a^* = a - \sum_i e_{ai} t_{ai}$ and $b^* = b - \sum_j e_{bj} t_{bj}$, are generated, and $E = \{e_{ai}\}_i \cup \{e_{bj}\}_j$ is the set of all edges subtracted to construct a^* and b^*. a and b are* generalized adjacent *in P^{E+}, denoted $adj_E(a,b)$, if $\nexists S, \sigma_{a^* b^* \cdot S} = 0$. Otherwise, a and b are* generalized non-adjacent *in P^{E+}, denoted $nadj_E(a,b)$. We denote the set S where $\sigma_{a^* b^* \cdot S} = 0$ as S^*_{ab}.*

A special case of Definition 3 is when only one AV, b^*, is generated, i.e., $adj_E(a,b)$ if $\nexists S, \sigma_{ab^* \cdot S} = 0$, and $nadj_E(a,b)$ otherwise. Next, we generalize *discriminating path* given in Zhang (2008), which is necessary for constructing one of the edge orientation rules. See Figure 4 for a graphical illustration.

[6]Typically, faithfulness implies that path-separation (Pearl 2009) in the true DAG precisely characterizes conditional independence in the data distribution. In our case, we require a slightly stronger version of this assumption in which Theorem 1 of Chen, Kumor, and Bareinboim (2017) precisely characterizes the AV-conditional independence constraints in the data.

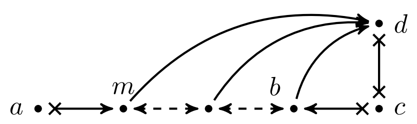

Figure 4: A generalized discriminating path, $u = \langle a, m, \cdots, b, c, d\rangle$, between a and d for c

Definition 4 (generalized discriminating path)**.** *$u = \langle a, \cdots, b, c, d\rangle$, is a* generalized discriminating path *between a and d for c if*

(i) u includes at least three edges;

(ii) c is a non-end node on u, and is adjacent to d on u;

(iii) every node between a and c is a collider on u and is a parent of d; and

(iv) denote m as the node following a on u (can be b). $\exists E \in E_{ad}$, $nadj_E(a,d)$, $adj_E(a,m)$, and for every node n between a and d, $adj_E(n,d)$.

Now, we construct the edge orientation rules based on AV conditional independence constraints. These rules generalize the rules of the FCI algorithm for DAGs for generalized adjacency and non-adjacency and are iteratively performed. E_K denotes the set of known or identified directed edges at the current iteration. For simplicity, for each pair of variables a and b, we define $E_{ab} = \{E_K \cap Inc(a), E_K \cap Inc(b), E_K \cap (Inc(a) \cup Inc(b))\}$. The edge mark $*$ is a wildcard representing any of an arrowhead, a tail, and a circle, and remains the same after an orientation rule.

Rule 0: For every adjacent pair a and b, if $\exists E \in E_{ab}$, $nadj_E(a,b)$, and the edge $a \ast\!\!-\!\!\ast b$ is not in E, record $a \ast\!\!-\!\!\ast b$ as extraneous without removing it.

Rule 1: For every triple a, b and c, if (i) $\exists E \in E_{ac}$, $nadj_E(a,c)$, $adj_E(a,b)$, $adj_E(b,c)$, and (ii) $b \notin S^*_{ac}$, then orient $a \ast\!\!\rightarrow b \leftarrow\!\!\ast c$.

Rule 2: For every triple a, b and c, if (i) $\exists E \in E_{ac}$, $nadj_E(a,c)$, $adj_E(a,b)$, $adj_E(b,c)$, (ii) $b \in S^*_{ac}$, and (iii) $a \ast\!\!\rightarrow b \circ\!\!-\!\!\ast c$, then orient $a \ast\!\!\rightarrow b \rightarrow c$.

Rule 3: For every pair a and d, if $\exists u = \langle a, \cdots, b, c, d\rangle$, a generalized discriminating path between a and d for c, then

(i) if $c \notin S^*_{ad}$, orient $b \leftrightarrow c \leftarrow\!\!\ast d$,

(ii) if $c \in S^*_{ad}$, $b \leftrightarrow c$, and $c \circ\!\!-\!\!\ast d$, orient $c \rightarrow d$,

(iii) if $c \in S^*_{ad}$, $d \leftrightarrow c$, and $c \circ\!\!\rightarrow b$, orient $c \rightarrow b$.

Rules 0-3 describe how to use AV conditional independences found in the data to orient edges. Rule 0 is a special case of blocking paths. An edge in a pattern P is regarded extraneous with respect to the true DAG G if the two nodes on that edge in P are non-adjacent in G. Consider the example in Figure 2. Figure 2(a) is the true DAG. Figure 2(b) is the pattern learned using the FCI algorithm, where only traditional conditional independence constraints are used. Extraneous edges $c \circ\!\!\rightarrow f$, $d \circ\!\!\rightarrow e$, and $e \leftarrow\!\circ f$ that do not exist in the true DAG are learned, because there is no separating set W for c and f such that $c \perp\!\!\!\perp f | W$, and same for the other two pairs. However, we do not remove the extraneous edge,

$a \ast\!\!-\!\!\ast\, b$, immediately when it is found. This is because when performing other orientation rules, if $adj_E(a, b)$ for that E, then $a \ast\!-\!\ast\, b$ can be used the same way as if it were non-extraneous, which might help orient other edges.

Rule 1 states that b must be a collider if a and c are independent without conditioning on b but dependent when conditioning on b. Rule 2 states that the middle node cannot be a collider if a and c are independent when conditioning on b but dependent otherwise. The example of Figure 3(a) explained before is an application of Rule 1.

Rule 3 is more complicated. The intuition behind discriminating paths is to choose orientations for $b \leftarrow\!\!\ast\, c$ and $c \ast\!\!-\!\!\ast\, d$ that block the paths between a and d. If a and d are non-adjacent, there exists a conditioning set, S, that blocks all the paths between them. All the nodes between a and d on u must be in S, because otherwise there is an unblocked path $a \ast\!\!\rightarrow m \dashleftarrow\!\dashrightarrow \cdots \rightarrow d$. Therefore, u must be unblocked from a to c, and we have to block u at c. Now, we just have to check if $c \in S$, and $b \leftarrow\!\!\ast\, c$ and $c \ast\!\!\rightarrow\!\!\ast\, d$ can be oriented the same ways as Rules 1 and 2, where part (i) in Rule 3 corresponds to Rule 1 and parts (ii) and (iii) in Rule 3 correspond to Rule 2. Compared to the original definition of discriminating paths, generalized discriminating paths do not require a and d to be non-adjacent, but only require them to be generalized non-adjacent, and all the adjacent nodes to be generalized adjacent. Changing those adjacency relationships to generalized adjacencies can be understood as virtually removing E in order to analyze the paths between those nodes in P_{E-}.

Causal Identification in Patterns

Generating AVs requires either a priori knowledge of coefficient values or identification of coefficients. In this section, we show how to identify causal effects in linear patterns, which will allow us to use AVs to help learn causal structures from obserevational data. For example, in Figure 2(c), the edge $d \rightarrow f$ is identifiable using the instrumental variable (IV) method (Bowden and Turkington 1990). Although the DAG is incomplete, we can still see there is no unblocked path between a and f not through d (we can see this by enumerating all possibilities of circle marks), which makes a a valid IV. In other words, for any full DAG represented by this pattern, the coefficient on $d \rightarrow f$ is equal to σ_{af}/σ_{ad}.

The most general, efficient[7] identification algorithm in fully specified linear SCMs is the qID method (Chen, Kumor, and Bareinboim 2017). qID uses quasi-instrumental sets, which are an extension of generalized instrumental sets (Brito and Pearl 2002) for AVs. Our method can be understood as defining a stricter version of quasi-instrumental set for patterns, named *determinate quasi-instrumental set*. More formally, if Z is a determinate quasi-instrumental set for edges E in a pattern P, then Z is a quasi-instrumental set for E in any member of $[G]$ represented by P. This will enable us to identify E given P, and is guaranteed to give the same results as if we had the true DAG G_{true}, as long as G_{true} belongs to $[G]$.

To achieve this goal, we first define *determinate descendants* (De^*), *determinately unblocked paths*, *determinate non-descendants* (NDe^*), *determinately blocked paths*, and *determinately d-separated* ($dsep^*$). y is a *determinate descendant* (De^*) of x in pattern P, if x is a descendant of y in every graph represented by P. Similarly, p is a *determinately unblocked path* in P if it is an unblocked path in all graphs represented by P. Determinate non-descendant, determinately blocked path, and determinately d-separated are similarly defined. Lastly, a set of paths have *no sided intersection* if for every pair of paths, they do not share any node that has an arrow to the same direction on both paths (Foygel et al. 2012). Characterizations for each of these definitions in patterns are given in the Appendix.

Now, we describe how to find determinate quasi-instrumental sets in a pattern.

Theorem 1. *Given a linear SEM with pattern P, a set of edges E_K whose coefficient values are known, and a set of structural coefficients $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$, the set $Z = \{z_1, \cdots, z_k\}$ is a* determinate quasi-instrumental set *for α if there exist triples $(z_1, W_1, \pi_1), \cdots, (z_k, W_k, \pi_k)$ such that:*

- *(i) For $i = 1, \cdots, k$, either:*
  - *(a) $W_i \in NDe^*(y)$, and $dsep^*(z_i, W_i, y)_{P_{E \cup E_y -}}$ where $E_y = E_K \cap Inc(y)$, or*
  - *(b) $W_i \in NDe^*(y) \cap NDe^*(z_i)$, and $dsep^*(z_i, W_i, y)_{P_{E \cup E_{zy} -}}$ where $E_{zy} = E_K \cap (Inc(z) \cup Inc(y))$*
- *(ii) for $i = 1, \cdots, k$, π_i is a path between z_i and x_i that is determinately unblocked by W_i in $P_{E \cup E_y -}$ if z_i satisfies (i)(a) and in $P_{E \cup E_{zy} -}$ if z_i satisfies (i)(b), where $x_i = Ta(\alpha_i)$, and*
- *(iii) the paths $\{\pi_1, \cdots, \pi_k\}$ have no sided intersection.*

Theorem 2 (Identifiability)**.** *If Z is a determinate quasi-instrumental set for E, then E is identifiable.*

In addition to enabling the usage of AVs and, therefore, the usage of Verma constraints in causal discovery, identification in patterns is also useful on its own. It allows us to compute causal effects from incomplete or even zero knowledge about the underlying causal structure.

Algorithm for Learning Patterns and Identification

In this section, we construct an algorithm for simultaneous causal discovery and identification. When learning a pattern from data and prior knowledge, we want the pattern to contain only features in the true DAG, but also be as specific as possible, i.e., we want to learn as many invariant arrowheads and tails as possible and remove as many extraneous edges as possible. As we have discussed, structure learning and causal identification can benefit each other. Learning a more precise pattern helps with identifying more edges. Identifying more edges allows us to create more AVs and learn more AV conditional independence constraints, which helps with learning a more precise structure. We construct the Linear Causal Discovery and Identification (LCDI) algorithm that implements this bootstrapping procedure to learn a pattern P and identify causal coefficients given

[7]qID is polynomial-time if the degree of the nodes are bounded.

observational data.

Linear Causal Discovery and Identification (LCDI)

Input: covariance matrix σ_V on the set of observed variables V and a set of identified edges E_{id} (can be empty)

Output: a pattern P and updated E_{id}

Step 0: Run FCI algorithm (Zhang 2008) on σ_V with Rules $\mathcal{R}1$-$\mathcal{R}4$ only, but replacing $\mathcal{R}4$ with $\mathcal{R}4-$ given below. The resulting pattern is P;

Step 1: Run the original FCI algorithm on σ_V with Rules $\mathcal{R}1$-$\mathcal{R}4$ and $\mathcal{R}8$-$\mathcal{R}10$[8] to obtain a PAG P', and merge the arrowheads in P' to P;

Step 2: Repeat the following Substeps on P until neither P nor E_{id} is updating;

- *Substep 0:* Perform causal identification on P without extraneous edges and update E_{id};
- *Substep 1:* Generate AVs using E_{id};
- *Substep 2:* Run Rules 0-3;
- *Substep 3:* Run FCI algorithm $\mathcal{R}1$ and $\mathcal{R}4+$ (given below) repeatedly until P is not updating;

Step 3: Remove all the extraneous edges marked in Rule 0 in Step 1 Substep 2 from P.

$\mathcal{R}4-$ and $\mathcal{R}4+$ below are modified from FCI. S_{ad} denotes the set of conditioning variables which makes a and d independent.

$\mathcal{R}4-$: $u = \langle a, \cdots, b, c, d\rangle$ is a discriminating path[9] between a and d for c; then

(i) if $c \notin S_{ad}$, and $c \circ\!\!-\!\!* d$, orient $b \leftrightarrow c \leftrightarrow d$;

(ii) if $c \in S_{ad}$, and $c *\!\!-\!\!\circ d$, orient $c *\!\!\rightarrow d$.

$\mathcal{R}4+$: $u = \langle a, \cdots, b, c, d\rangle$ is a discriminating path between a and d for c; then

(i) if $c \notin S_{ad}$, orient $b \leftrightarrow c \leftarrow\!\!* d$ if not done so;

(ii) if $c \in S_{ad}$, $b \leftrightarrow c$, and $c \circ\!\!-\!\!* d$, orient $c \rightarrow d$;

(iii) if $c \in S_{ad}$, $d \leftrightarrow c$, and $c \circ\!\!\rightarrow b$, orient $c \rightarrow b$;

(iv) if $c \in S_{ad}$, and $c *\!\!-\!\!\circ d$, orient $c *\!\!\rightarrow d$.

We use $\mathcal{R}4-$ and $\mathcal{R}4+$ instead of $\mathcal{R}4$ because FCI tries to recover the MAG representation for the true DAG, while our method aims to recover the true DAG directly. They make sure the resulting pattern is consistent with the true DAG instead of the MAG. We skip the tail orientation rules $\mathcal{R}8$-$\mathcal{R}10$ in the original FCI for the same reason. See the next section for a more detailed discussion of MAGs and DAGs. The correctness of LCDI is summarized in the following theorem.

Theorem 3. *P is the pattern output by LCDI, then the true DAG G that was used to generate the covariance matrix σ_V must be a member of [G] represented by P.*

[8]We skip $\mathcal{R}5$-$\mathcal{R}7$ because they are useful in dealing with selection bias, while we assume no selection bias.

[9]A discriminating path is defined as a generalized discriminating path replacing all generalized adjacency relationships with normal adjacency relationships in Definition 4.

Theorem 3 shows that any arrowhead or tail learned by LCDI must be present in the true DAG. Algorithms such as FCI that aim to recover a MAG only guarantees tail correctness regarding the MAG converted from the true DAG, but might learn tails that do not exist in the true DAG. However, correct tail orientations are an important factor for causal inference since they help distinguish between direct causation and confounded correlation, while LCDI guarantees tail soundness regarding the true DAG.

We will use the example of Figure 2 to illustrate LCDI. Figure 2(a) shows the underlying true DAG we want to recover. LCDI begins with Step 0, an iteration of modified FCI, which utilizes conditional independence constraints to learn the pattern in Figure 2(c). Extraneous edges $c \circ\!\!\rightarrow f$, $d \circ\!\!\rightarrow e$, and $e \leftarrow\!\!\circ f$ are learned, because there is no separating set that can make each pair of variables conditionally independent. In Step 1, we merge the arrowheads from the PAG learned using FCI, shown in 2(a), to the pattern from Step 0. In this specific example, no arrowhead is newly added. However, there are cases where FCI learns additional arrowheads that cannot be learned using modified FCI.

Next, in Step 2 Substep 0, the only identifiable edge in Figure 2(c) is $d \rightarrow f$, using $\{a\}$ as a determinate quasi-instrumental set. This allows the AV, $f^* = f - a{\cdot}\alpha$, where α is the coefficient on $d \rightarrow f$, to be generated in Substep 1. Next, in Substep 2, LCDI searches for conditional independences between the newly generated AVs and other variables. In Rule 0, $nadj_{\{d\rightarrow f\}}(c, f)$ since $\sigma_{cf^*\cdot\varnothing} = 0$, and $c \circ\!\!\rightarrow f$ is recorded as extraneous. Similarly, $e \leftarrow\!\!\circ f$ is recorded as extraneous. In Rule 1, $nadj_{\{d\rightarrow f\}}(c, f)$ and $b \notin S^*_{cf}$ give orientations $c \leftrightarrow b \leftrightarrow f$, and $nadj_{\{e\rightarrow f\}}(c, f)$ and $b \notin S^*_{cf}$ give orientation $e \leftrightarrow b$. In Rule 3, we can find a generalized discriminating path, $u = \langle c, d, b, f\rangle$ between c and f for b, and condition (iii) gives $b \rightarrow d$.

In the next iteration of Step 2, we find $b \rightarrow d$ is now identifiable using $\{b\}$ as a determinate quasi-instrumental set, and as before, $d \circ\!\!\rightarrow e$ is marked extraneous. In Step 2, we orient $d \leftrightarrow a \leftrightarrow e$, $c \rightarrow e$, and $e \leftrightarrow f$.

In the third iteration of Step 2, we find $c \rightarrow e$ is identifiable using $\{c\}$ as a determinate quasi-instrumental set. No more edge orientations can be deduced. Lastly, in Step 3, all the three extraneous edges are removed, and we obtain the final pattern, Figure 2(d).

Compared to the pattern learned by FCI in Figure 2(b), the pattern learned by LCDI was much more informative. First, LCDI removed all the extraneous edges, while FCI had three of them. Second, LCDI learned more edge orientations (in this specific example, LCDI was even able to recover all the edge orientations!) while FCI had quite a few circle marks. Third, LCDI guaranteed tail soundness regarding the true DAG, while FCI oriented $b \rightarrow f$, which was in the MAG representation of the true DAG, but was inconsistent with the true DAG itself.

The runtime of LCDI is composed of two parts, identification and structure update. Denote the runtime of qID in (Chen, Kumor, and Bareinboim 2017) as q, the runtime of FCI in (Zhang 2008) as f, the number of iterations run as r, then the runtime of LCDI is $\mathcal{O}(r(q + f))$. r is bounded by

TECHNICAL REPORT
R-490
December 2019

d \ n	6	7	8	9	10	11
$(1.5, 2]$	1.0	4.0	2.0	1.0	1.5	2.0
$(2.75, 3.25]$	5.5	8.0	15.5	18.5	23.5	25.5
$(4, 4.5]$	0	8.0	15.0	32.5	36.5	45.0

Table 1: percentage of graphs where LCDI learns more arrowheads than FCI

d \ n	6	7	8	9	10	11
$(1.5, 2]$	17.4	19.2	13.0	12.5	13.8	11.6
$(2.75, 3.25]$	11.1	8.7	12.7	9.6	10.4	7.4
$(4, 4.5]$	0	8.7	7.4	8.0	7.0	6.8

Table 2: percentage more of arrowheads LCDI learns than FCI in graphs where LCDI learns more arrowheads

the number of edges in the initial pattern, but is likely to be much smaller.

Simulation Results

To illustrate the advantages of LCDI, we compare it with FCI, which is considered to be the current state of the art constraint-based causal discovery algorithm without additional assumptions on the data distribution. FCI was first proposed by Spirtes et al. (2000), and the improved version by Zhang (2008) achieved arrowhead and tail completeness, i.e., it can learn every invariant arrowhead and tail for the equivalence class of MAGs. However, FCI might recover more tails than there are in the true DAG, because the MAG itself might have more tails. The PAG in Figure 2(b) has a directed edge, $b \rightarrow f$, which is in fact a bidirected edge, $b \leftrightarrow f$, in the true DAG (2(a)). However, FCI does not recover more arrowheads than there are in the true DAG. The following theorem shows the power of orienting arrowheads in LCDI.

Theorem 4. *Under the linear setting and given the covariance matrix of the data, if an invariant arrowhead can be recovered by FCI, then it can be recovered by LCDI.*

Theorem 4 results directly from how LCDI is constructed, and it implies that LCDI always recovers equal or more correct arrowheads compared to FCI.

To quantify this improvement, we implemented LCDI and the version of FCI by Zhang (2008). We randomly generate DAGs with number of nodes (n) from 6 to 11 with various average node degrees (d), and an edge being directed and bidirected both have probability 0.5. We then compare the patterns that would be learned on the generated DAG by each method assuming faithfulness. More specifically, we compare the number of invariant arrowheads and extraneous edges learned. Each data entry in Tables 1 and 2 was averaged over 200 random DAGs.

Table 1 shows for DAGs of different node numbers, the percentages of DAGs where LCDI learns at least one more arrowhead than FCI, for different d ranges ($(1.5, 2]$, $(2.75, 3.25]$, $(4, 4.5]$). As we can see, the benefit of LCDI generally increases with the number of nodes in the DAG. In

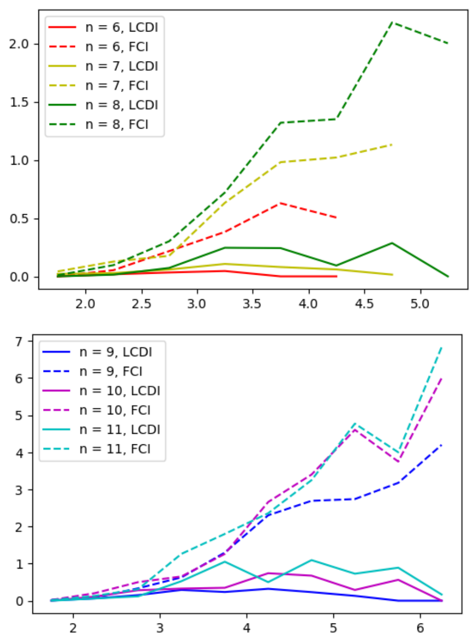

Figure 5: numbers of extraneous edges learned by FCI vs. numbers of extraneous edges learned by LCDI

over 45% of the DAGs with $n = 11$ and large d, LCDI learns more arrowheads, which is a significant improvement.

Table 2 shows for the DAGs where LCDI learns more arrowheads, how much more can LCDI learn compared to FCI. For any n, it can recover 10% to 20% of total arrowheads more than FCI when d is small.

Figure 5 shows the numbers of extraneous edges learned by FCI and LCDI. The different colors indicate DAGs of different node numbers. On average, LCDI learns less than 1 extraneous edge for any n and d, while the number of extraneous edges FCI learns increases as n and d increases.

We can see LCDI provides decent improvements in a large percentage of random DAGs–it learns more arrowheads and less extraneous edges. Furthermore, these improvements do not sacrifice correctness. All the arrowheads and tails LCDI learns and all the extraneous edges it removes are guaranteed to be in the true DAG.

Related Work

Shpitser, Richardson, and Robins (2009) introduced a method to test extraneous edges using Verma constraints under the non-parametric setting. Their work is limited to full DAGs and is not generalized to partial DAGs.

Jaber, Zhang, and Bareinboim (2018) introduced an identification method for PAGs. Their method works in the non-parametric setting. In comparison, our method can identify some causal effects that cannot be identified without assuming linearity. In addition, our method is applied to patterns, which are consistent with the true DAG.

Shpitser et al. (2012) introduced a score-based causal discovery method. Their method incorporates Verma constraints in a different way: their Q-FIT algorithm fits parameters such that if two graphs are equivalent in terms of Verma constraints, they have the same score. Their method searches for graphs with highest likelihood score based on data. However, the resulting graph is a full DAG. Therefore, even though that DAG is Verma-constraint-equivalent to the true DAG, we still might not be able to infer what structures the true DAG has, since it is in general impossible to list all equivalent DAGs and summarize their characteristics. In comparison, our method is constraint-based, and learns an equivalent class that is guaranteed to represent the true DAG.

Shimizu et al. (2006) introduced a linear causal discovery method. It assumes non-faithfulness, no latent confounders, and non-Gaussian errors. In contrast, we assume faithfulness and relax the other two assumptions.

Conclusion

In this paper, we developed a symbiotic approach to causal discovery and identification in linear models. We first formally defined the type of partially specified DAGs, patterns, that are useful for both causal discovery and identification. We then devised a method of incorporating Verma constraints using auxiliary variables, and method of identification on patterns. Finally, we developed an algorithm that performs causal discovery and identification simultaneously, for mutual benefit. We showed that the combined algorithm performs better than doing each task separately. In addition, our algorithm can learn more complete structures than previously reported algorithms.

Acknowledgements

Zhang and Pearl are supported in parts by grants from International Business Machines Corporation (IBM) [#A1771928], National Science Foundation [#IIS-1527490 and #IIS1704932], and Office of Naval Research [#N00014-17-S-B001]. The authors would like to thank Yujia Shen, Elias Bareinboim, and Carlos Cinelli for helpful discussions.

References

Bowden, R. J., and Turkington, D. A. 1990. *Instrumental variables*, volume 8. Cambridge University Press.

Brito, C., and Pearl, J. 2002. Generalized instrumental variables. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 85–93. Morgan Kaufmann Publishers Inc.

Chen, B.; Kumor, D.; and Bareinboim, E. 2017. Identification and model testing in linear structural equation models using auxiliary variables. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 757–766. JMLR. org.

Chen, B.; Pearl, J.; and Bareinboim, E. 2015. Incorporating knowledge into structural equation models using auxiliary variables. *arXiv preprint arXiv:1511.02995*.

Chen, B. 2016. Identification and overidentification of linear structural equation models. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 1579–1587.

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

Foygel, R.; Draisma, J.; Drton, M.; et al. 2012. Half-trek criterion for generic identifiability of linear structural equation models. *The Annals of Statistics* 40(3):1682–1713.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning* 20(3):197–243.

Jaber, A.; Zhang, J.; and Bareinboim, E. 2018. Causal identification under markov equivalence. *arXiv preprint arXiv:1812.06209*.

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

Ramsey, J.; Glymour, M.; Sanchez-Romero, R.; and Glymour, C. 2017. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics* 3(2):121–129.

Richardson, T., and Spirtes, P. 2002. Ancestral graph markov models. *Ann. Statist.* 30(4):962–1030.

Richardson, T. 1996. A discovery algorithm for directed cyclic graphs. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, 454–461. Morgan Kaufmann Publishers Inc.

Shimizu, S.; Hoyer, P. O.; Hyvärinen, A.; and Kerminen, A. 2006. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7(Oct):2003–2030.

Shpitser, I., and Pearl, J. 2008. Dormant independence. Technical Report R-340L, <http://ftp.cs.ucla.edu/pub/stat_ser/r340-L.pdf>, Department of Computer Science, University of California, Los Angeles, CA. Extended version of paper that appeared in AAAI-08.

Shpitser, I.; Richardson, T. S.; Robins, J. M.; and Evans, R. 2012. Parameter and structure learning in nested markov models. *arXiv preprint arXiv:1207.5058*.

Shpitser, I.; Richardson, T. S.; and Robins, J. M. 2009. Testing edges by truncations. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Spirtes, P.; Glymour, C. N.; Scheines, R.; Heckerman, D.; Meek, C.; Cooper, G.; and Richardson, T. 2000. *Causation, prediction, and search.* MIT press.

Tian, J., and Pearl, J. 2002. On the testable implications of causal models with hidden variables. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 519–527. Morgan Kaufmann Publishers Inc.

Van der Zander, B., and Liskiewicz, M. 2016. On searching for generalized instrumental variables. In *AISTATS*, 1214–1222.

Verma, T., and Pearl, J. 1991. *Equivalence and synthesis of causal models*. UCLA, Computer Science Department.

Zhang, J. 2008. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence* 172(16-17):1873–1896.

TECHNICAL REPORT
R-490
December 2019

Appendix

Supplemental Definitions

For the following sections, we use $Anc(y)$ to denote the ancestors of y, and $De(y)$ to denote the descendants of y. $Nbr(v)$ denotes the set of nodes adjacent to v. $dsep(X, S_{XY}, Y)$ denotes the sets of variables X and Y are d-separated by the set of variables S_{XY}.

Definition 5. *In a pattern P, for three nodes $a, b,$ and c where $b \in Nbr(a) \cap Nbr(c)$, we have the following definitions.*

(i) b is a colliding node *of a and c if $a \ast\!\!\rightarrow b \leftarrow\!\!\ast\, c$, denoted $b \in Col(a, c)$.*

(ii) b is a blocking node *of a and c if there is at least one of $a \leftarrow b$ and $b \rightarrow c$ holds, denoted $b \in Blo(a, c)$.*

Definition 6. *In a pattern P, if we can reach a node b from a node a through connected edges regardless of the directions, while not passing through any node twice, then that path is defined as a* simple path.

Definition 7. *$Right'(\pi)$ are the set of nodes, if any, that has a directed edge leaving it in the direction of y in addition to y. $Left'(\pi)$ are the set of nodes, if any, that has a directed edge leaving it in the direction of x in addition to x.*

Definition 8. *In a pattern P, a set of determinate unblocked paths, $\pi_1, \cdots, \pi_n$, has* no sided intersection *if for all $\pi_i, \pi_j \in \{\pi_1, \cdots, \pi_n\}$ such that $\pi_i \neq \pi_j$, $Left'(\pi_i) \cap Left'(\pi_j) = Right'(\pi_i) \cap Right'(\pi_j) = \varnothing$.*

Lemma 2. *A simple path $\pi_s = \{x, v_1, \cdots, v_k, y\}$ between two nodes x and y is determinately blocked by a set of nodes S_{xy} if there exists v_i where $i \in \{1, \cdots, k\}$*

(i) $v_i \in Col(v_{i-1}, v_{i+1})$ and $v_i \notin S_{xy}$ and $\forall v \notin NDe^*(b), v \notin S_{xy}$, or

(ii) $v_i \in Bol(v_{i-1}, v_{i+1})$ and $v_i \in S_{xy}$.

Definition 9. *(Chen, Kumor, and Bareinboim 2017) Given a linear SEM with graph G, a set of edges E_K whose coefficient values are known, and a set of structural coefficients $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$, the set $Z = \{z_1, \cdots, z_k\}$ is a* quasi-instrumental set *if there exist triples $(z_1, W_1, \pi_1), \cdots, (z_k, W_k, \pi_k)$ such that:*

(i) For $i = 1, \cdots, k$, either:

(a) $W_i \notin De(y)$, and $(z_i \perp\!\!\!\perp y | W_i)_{G_{E \cup E_y -}}$ where $E_y = E_K \cap Inc(y)$, or

(b) $W_i \notin De(y) \cup De(z_i)$, and $(z_i \perp\!\!\!\perp y | W_i)_{G_{E \cup E_{zy} -}}$ where $E_{zy} = E_K \cap (Inc(z) \cup Inc(y))$

(ii) for $i = 1, \cdots, k$, π_i is a path between z_i and x_i that is not blocked by W_i in $G_{E \cup E_y -}$ if z_i satisfies (i)(a) and in $G_{E \cup E_{zy} -}$ if z_i satisfies (i)(b), where $x_i = Ta(\alpha_i)$, and

(iii) the set of paths $\{\pi_1, \cdots, \pi_k\}$ has no sided intersection.

Proof of Identifiability

Lemma 3. *In a pattern P, for two nodes a and b, $a \in De^*(b)$ if there exists a directed path composed solely of directed edges from b to a.*

Proof. Since each directed edge must be invariant in $[G]$ represented by P, that path in P must be a directed path in any member in $[G]$, which matches the definition of descendants in a full DAG. □

Lemma 4. *A simple path $\pi_s = \{x, v_1, \cdots, v_k, y\}$ between two nodes x and y is* determinately unblocked *by a set of nodes S_{xy} if for each v_i where $i \in \{1, \cdots, k\}$*

(i) $v_i \in Col(v_{i-1}, v_{i+1})$, and $v_i \in S_{xy}$ or $\exists v \in De^*(v_i), v \in S_{xy}$, or

(ii) $v_i \in Bol(v_{i-1}, v_{i+1})$ and $v_i \notin S_{xy}$.

Proof. We look at the two cases. We prove that if a node on the path satisfying either condition, it does not block the path. So if every node on the path satisfies a condition, then the entire path is not blocked.

(i) $v_{i-1} \ast\!\!\rightarrow v_i \leftarrow\!\!\ast\, v_{i+1}$. The two arrowheads are invariant in $[G]$, hence v_i is a collider in any member of $[G]$. So if v_i or any of v_i's determinate descendants is conditioned on, the path is unblocked at v_i.

(ii) The existence of at least one directed edge away from v_i guarantees that there cannot be arrows pointing to v_i from both v_{i-1} and v_{i+1} in any G, which means not conditioning on v_i makes the path unblocked at v_i in any G.

□

Lemma 5. *In a pattern P, for two nodes a and b, $a \in NDe^*(b)$ if*

(i) there exists a directed path composed solely of directed edges from a to b, or

(ii) for each simple path π_s from a to b, there exists an edge on π_s with an arrow in the direction to b.

Proof. We look at the two cases and prove either case gives $a \notin De(b)$ in G.

(i) The directed path in P can only be a directed path in $[G]$, which means $a \in Anc(b)$ in any G of $[G]$.

(ii) If $a \in De(b)$ in some G in $[G]$, then there must exist a directed path from b to a in G, which cannot have an arrow in the direction to b, leading to contradiction.

□

Lemma 6. *Given a pattern P, $dsep^*(X, S_{XY}, Y)$ if every simple path π_{s_i} between a node $x_i \in X$ and a node $y_i \in Y$ is determinately blocked by S_{XY}.*

Proof. We first prove that if a path π_s is determinately blocked by S_{xy} in P, then it is blocked in every G in $[G]$ represented by P. We look at the two cases in Definition 2.

(i) $v_{i-1} \ast\!\!\rightarrow v_i \leftarrow\!\!\ast\, v_{i+1}$. The two arrowheads are invariant in $[G]$, hence v_i is a collider in any member of $[G]$. So if none of v_i and any of v_i's possible descendants is conditioned on, the path is blocked.

(ii) The existence of at least one directed edge away from v_i guarantees that there cannot be arrows pointing to v_i from both v_{i-1} and v_{i+1} in any G, which means which means conditioning on v_i must block the path in any G.

By Definition 2, for any path in G, we will be able to find a corresponding simple path in P. So if every simple path in P between X and Y is blocked by S_{XY}, then every path in any G between X and Y must be blocked by S_{XY}, which means $dsep(X, S_{XY}, Y)$. □

Lemma 7. *If in a pattern P, a set of determinately unblocked paths $\pi_1, \cdots, \pi_n$ has no sided intersection, then it has no sided intersection in any fully specified DAG G represented by P.*

Proof. For any $\pi_i \in \{\pi_1, \cdots, \pi_n\}$, we only have to prove $Left(\pi_i) = Left'(\pi_i)$ and $Right(\pi_i) = Right'(\pi_i)$. This holds because the definitions are the same as in fully specified DAGs. □

Next, we prove our main identifiability theorems.

Theorem 1. *Given a linear SEM with pattern P, a set of edges E_K whose coefficient values are known, and a set of structural coefficients $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$, the set $Z = \{z_1, \cdots, z_k\}$ is a* determinate quasi-instrumental set *for α if there exist triples $(z_1, W_1, \pi_1), \cdots, (z_k, W_k, \pi_k)$ such that:*

(i) For $i = 1, \cdots, k$, either:

(a) $W_i \in NDe^*(y)$, and $dsep^*(z_i, W_i, y)_{P_{E \cup E_y -}}$ where $E_y = E_K \cap Inc(y)$, or

(b) $W_i \in NDe^*(y) \cap NDe^*(z_i)$, and $dsep^*(z_i, W_i, y)_{P_{E \cup E_{zy} -}}$ where $E_{zy} = E_K \cap (Inc(z) \cup Inc(y))$

(ii) for $i = 1, \cdots, k$, π_i is a path between z_i and x_i that is determinately unblocked by W_i in $P_{E \cup E_y -}$ if z_i satisfies (i)(a) and in $P_{E \cup E_{zy} -}$ if z_i satisfies (i)(b), where $x_i = Ta(\alpha_i)$, and

(iii) the paths $\{\pi_1, \cdots, \pi_k\}$ have no sided intersection.

Proof. We need to prove that if Z is a determinate quasi-instrumental set in pattern P, then Z is a quasi-instrumental set in any full DAG G in $[G]$ represented by P. We look at the three requirements respectively.

(i) (a) By Lemma 5, in any graph G, nodes in W_i are non-descendants of y. By Lemma 6, in any graph $G_{E \cup E_y -}$ represented by $P_{E \cup E_y -}$, W_i d-separates z_i and y.

(b) By Lemma 5, in any graph G, nodes in W_i are non-descendants of y or z_i. By Lemma 6, in any graph $G_{E \cup E_{zy} -}$ represented by $P_{E \cup E_{zy} -}$, W_i d-separates z_i and y.

(ii) By Lemma 4, π_i is an unblocked path in the corresponding modified graph.

(iii) By Lemma 7, the set of paths has no sided intersection in any G.

Those prove that Z satisfy the requirements of quasi-instrumental set for E. □

Theorem 2 (Identifiability)**.** *If Z is a determinate quasi-instrumental set for E, then E is identifiable.*

Proof. By the definition of determinate quasi-instrumental set, Z is a quasi-instrumental set for E in any full DAG G in $[G]$ represented by P. We know that if Z is a quasi-instrumental set for E in G, then E is identifiable. The solution is unique for any G, because the sets of linear equations generated by Z to solve for the coefficients of E for different G are exactly the same. □

Identification Algorithms

Algorithm 1 Find a separating set S_{YZ} such that $dsep^*(Y, S_{YZ}, Z)$ in pattern P

```
1: function FINDSEP($P, Y, Z$)
2:     $S_{YZ} = \varnothing$
3:     while $\exists \pi_s = Y, V_1, \cdots, V_k, Z$, a simple path between $Y$ and $Z$ s.t. $k \geq 1$, $\pi_s$ not determinately blocked by $S_{YZ}$ do
4:         blocked = $false$
5:         for $i = 1, \cdots, k$ do
6:             if $\{V_i\}$ determinately blocks $\pi_s \wedge V_i \in NDe^*(Y)$ then
7:                 $S_{YZ} \leftarrow S_{YZ} \cup \{V_i\}$
8:                 blocked = $true$
9:                 Break
10:        if blocked = $false$ then
11:            return $\perp$
12:    return $S_{YZ}$
```

Algorithm 2 Modified version of TestQIS from Chen, Kumor, and Bareinboim (2017) for a pattern P

```
1: function TESTQISP($P, X, Y, Z, E, E_{id}, AUX$)
2:     for i in $1, \cdots, |Z|$ do
3:         if $Aux_i$==1 then
4:             $W_i \leftarrow$ FindSep($P_{E \cup E_{Z_i} \cup E_y -}, Y, Z_i$), where $E_{Z_i} = E_{id} \cap Inc(z_i)$
5:             if $W_i = \perp \vee (W_i \cap (V \setminus NDe^*(y))) \neq \varnothing \vee (W_i \cap (V \setminus NDe^*(z_i))) \neq \varnothing$ then
6:                 return $\perp$
7:         else
8:             $W_i \leftarrow$ FindSep($P_{E \cup E_y -}, Y, Z_i$)
9:             if $W_i = \perp \vee (W_i \cap (V \setminus NDe^*(y))) \neq \varnothing$ then
10:                return $\perp$
11:    continue algorithm TestGeneralIVs from Van der Zander and Liskiewicz (2016) using the modified graph for each $Z_i$ starting from second for loop.
12:    Instead of returning $False$, return $\perp$,
13:    and instead of returning $True$, return $W$.
```

TECHNICAL REPORT
R-490
December 2019

Algorithm 3 Finds a determinate quasi-instrumental set for a set of edges E in a pattern P, given a set E_{id} of identified edges

```
1: function FINDQISP($E, P, E_{id}$)
2:     for all $Z \subset V \setminus \{y\}$ of size $|E|$ do
3:         for all $Aux \in \{0,1\}^{|E|}$ do
4:             $W \leftarrow$ TestQISP($P, Ta(E), He(E), Z, E_{id}, Aux$)
5:             if $W \neq \bot$ then
6:                 return $(Z, W)$
7:     return $\bot$
```

Algorithm 4 Identify as many edges as possible in pattern P given σ_V and identified edges E_{id} (Modified from Chen, Kumor, and Bareinboim (2017) qID Algorithm)

```
1: function QIDP($P, \sigma_V, E_{id}$)
2:     Initialize $EdgeSets \leftarrow$ all connected marked edge sets in $P$
3:     repeat
4:         for all $ES$ in $EdgeSets$ such that $ES \nsubseteq E_{id}$ do
5:             $y \leftarrow He(ES)$
6:             for all $E \subseteq ES$ such that $E \nsubseteq E_{id}$ do
7:                 $(Z, W) \leftarrow FindQISP(E, P, E_{id})$
8:                 if $(Z, W) \neq \bot$ then
9:                     Identify $E$ using $Z^*$ as an auxiliary instrumental set in $G^{E_{id} \cap Inc(Z)+}$
10:                    $E_{id} \leftarrow E_{id} \cup E$
11:    until All coefficients have been identified or no coefficients have been identified in the last iteration
12:    return $\bot$
```

Component Algorithms for LCDI

We use $TaArr(v)$ to denote the set of nodes connected to node v by any edge that has an arrow pointing to v.

Algorithm 5 Find a set of variables S such that $a \perp\!\!\!\perp h^* | S$, given σ_V

```
1: function FINDINDEP($a, h^*, \sigma_V$)
2:     if $a$ is an AV then
3:         $a_o \leftarrow$ the variable in $V$ used to generate $a$
4:     else
5:         $a_o \leftarrow a$
6:     for $i = |V| - 2, |V| - 1, \cdots, 0$ do
7:         for all $|S| = i$ and $S \subset V \setminus \{a_o, h\}$ do
8:             if $\sigma_{a_o h^* \cdot S} = 0$ then
9:                 return $S$
10:    return $\bot$
```

Algorithm 6 Generate Auxiliary Variable h^* for variable h given a pattern P and E_{id}, return h^* and the variables used to generate h^*

```
1: function GENAV($h, P, E_{id}$)
2:     Initialize $T \leftarrow \varnothing$
3:     for all $e_i \in Inc(h) \cap E_{id}$ do
4:         $\alpha_i \leftarrow$ the coefficient on $e_i$
5:         $T \leftarrow T \cup (\alpha_i, Ta(e_i))$
6:     Generate AV $h^* = h - \sum_{(\alpha_j, t_j) \in T} \alpha_j t_j$
7:     $T_{var} \leftarrow \cup_{(\alpha_j, t_j) \in T} \{t_j\}$
8:     return $h^*, T_{var}$
```

Algorithm 7 Learn a pattern given σ_V and a set of identified edges, E_{id} (can be empty)

```
1: function LCDI($\sigma_V, E_{id}$)
2:     $P \leftarrow$ FCI steps $\mathcal{R}1$-$\mathcal{R}3$, $\mathcal{R}4-$, with input $\sigma_V$
3:     $P' \leftarrow$ FCI with input $\sigma_V$
4:     $P \leftarrow P$+arrowheads from $P'$
5:     Initialize $E_{ex} \leftarrow \varnothing$
6:     repeat
7:         $qIDP(P_{E_{ex}-}, \sigma_V, E_{id})$
8:         for all pairs $a, b$ such that $a \in Nbr(b)$ do
9:             if $\exists E \in E_{ab}$, $nadj_E(a, b)$ then
10:                if $a *\!\!-\!\!* b \notin E$ then
11:                    $E_{ex} \leftarrow E_{ex} \cap \{a *\!\!-\!\!* b\}$
                       $P \leftarrow$ Rules 1-3 on $a$ and $b$ $P \leftarrow$ FCI steps $\mathcal{R}1$ and $\mathcal{R}4+$
12:    until $P$ and $E_{id}$ are both the same as the previous iteration
13:    $P \leftarrow P_{E_{ex}-}$
14:    return $P, E_{id}$
```

Proof of Correctness of LCDI

Corollary 1. *Given a linear pattern P representing $[G]$, where $E \subset Inc(z)$ is a set of edges whose coefficient values are known, if $(W \cup \{y\}) \cap (V \setminus NDe^*(z)) = \varnothing$, and G_{E-} represents the graph G with the edges for E removed, then $\sigma_{z^* y \cdot W} = 0$ only if $(z \perp\!\!\!\perp y | W)_{G_{E-}}$ for all G in $[G]$.*

Proof. By the definition of NDe^*, $NDe^*(z)_P = (V \setminus De(z))_G$. Hence, we have $(V \setminus NDe^*(z))_G = De(z)_P$. Together with $\sigma_{z^* y \cdot W} = 0$, by Theorem 1 in Chen, Kumor, and Bareinboim (2017), we have $(z \perp\!\!\!\perp y | W)_{G_{E-}}$. □

Lemma 1. *Given an AV, $z^* = z - \Sigma_i e_i t_i$, the conditional independence constraint, $\sigma_{az^* \cdot S} = 0$, is equivalent to the Verma constraint, $\sigma_{az \cdot S} - \Sigma_i e_i \sigma_{at_i \cdot S} = 0$, where S is a set of variables. Furthermore, this Verma constraint cannot, in general, be represented as a conditional independence constraint over the original model variables, V.*

Proof. Expanding z^* to $z - \Sigma_i e_i t_i$ in $\sigma_{az^* \cdot S}$ gives $\sigma_{az \cdot S} - \Sigma_i e_i \sigma_{at_i \cdot S}$ due to the linearity property of the covariance function. So the two constraints are equivalent. Since each e_i is identified, $\sigma_{az \cdot S} - \Sigma_i e_i \sigma_{at_i \cdot S} = 0$ is over only covariances, and is thus a valid Verma constraint. As a result, it cannot be equivalent to a conditional independence constraint over the original model variables, V. □

Lemma 8. *If an edge $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ in P is removed by LCDI, then $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ must be extraneous, i.e., in any G in $[G]$ represented by P, a and b are non-adjacent.*

Proof. $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ in P is marked extraneous if $\exists E, nadj_E(a, b)$, which implies a^* and b^* can be made conditionally independent. If in G, a and b are adjacent, $a^* \leftarrow a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b \rightarrow b^*$ is an unblocked path between a^* and b^*, which makes a^* and b^* always dependent, contradiction. □

Lemma 9. *If LCDI learns an edge $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ in P that is extraneous with respect to the true dag G, then $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ can be used in any single orientation step in LCDI as if it were not extraneous. In other words, treating $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ as a non-extraneous edge gives the same result as it were truly non-extraneous.*

Proof. A detailed proof is relatively complicated. We will only give a brief sketch of the idea. However, the reader can list all the steps involved and check all the cases one by one.

Extraneous edges are learned due to inducing paths between two nodes. We only need to prove having an inducing path between two nodes is the same as having a non-extraneous edge between two nodes, where the imaginary non-extraneous edge has the same orientation as the extraneous edge learned.

We only need to discuss the rules not included in FCI, namely, $\mathcal{R}4+$ and Step 2. In all those rules, what matters is whether conditioning on a node blocks the path between two non-adjacent or generalized non-adjacent nodes. There are two cases where being an extraneous edge might affect the result (due to symmetry, we only discuss a): 1) we are orienting an edge connected to a, and $a \leftarrow\!\!\!/\!\!\!-\!\!\ast b$ 2) we are not orienting an edge connected to a, but $a \ast\!\!-\!\!\!/\!\!\!-\!\!\ast b$ appears in the antecedent of the rule.

For case 1), the arrowhead at a must have been learned by one of previous steps to block the path to b. First, we prove it is not possible that there are only directed inducing paths from a to b, i.e., all inducing paths from a to b starts with a directed edge from a. If so, we will for example see in the pattern that conditioning on a unblocks the path to b, while in the inducing path a is not a collider. Thus, if there is an arrowhead at a, there must be at least one inducing path between a and b that starts with an arrowhead at a.

In LCDI, we always generate AVs that remove all known incoming edges. This implies that as the algorithm proceeds, the set of inducing paths we can "cut" is always superset of previous iterations. We describe it as "cut" because independence among AVs is the same as independence among original model variables with the generating edges removed. Therefore, if there are both directed inducing paths and bidirected inducing paths, the extraneous edge can only possibly "inherit" the orientation of the type of the inducing paths that remain uncut first. Once a type of inducing paths can be cut, there is no way to use the orientation of the cut inducing paths anymore, because they will always be cut later on. It can be understood as the "cut level" keeps increasing. If in the end both types of inducing paths are cut, the extraneous edge should be regarded as non-existing because the two nodes are generalized non-adjacent. Note there cannot be directed inducing paths and reversed directed inducing paths at the same time due to acyclicity.

Case 2) is mainly used for Rule 3 and $\mathcal{R}4+$. If $a \rightarrow b$, then there cannot be inducing paths with an arrowhead to a, because those inducing paths should have already been cut in order to learn $a \rightarrow b$. Same reasoning applies to $a \leftarrow \ast\, b$. □

Lemma 10. *If an arrowhead exists in a MAG M, then it must exist in any DAG D whose MAG representation is M (the arrowheads on the extraneous edges in M are not included).*

Proof. See Zhang (2008) for the rules to convert a DAG to its MAG representation. We just have to prove every time the rules orient an arrowhead in M, the arrowhead must be in D.

(i) When $a \rightarrow b$ is oriented in M, if there is no arrow at a in D and $a \rightarrow b$ is not extraneous, then it must be $a \leftarrow b$ in D. However, if $a \leftarrow b$ in D, a cannot be b's ancestor, which contradicts the MAG orientation rules.

(ii) When $a \leftrightarrow b$ is oriented in M, if there is no arrow at a or b in D and $a \leftrightarrow b$ is not extraneous, then either $a \rightarrow b$ or $a \leftarrow b$ in D. However, if $a \rightarrow b$ in D, we would have $a \rightarrow b$ in M, contradiction. Same reasoning applies to $a \leftarrow b$.

□

Lemma 11. *Denote the pattern Steps 0 and 1 of LCDI learn as P, then the true DAG G must be in the class $[G]$ that P represents.*

Proof. We first look at Step 0.

By Lemma 10, the arrowheads in the pattern P resulting from FCI $\mathcal{R}1$-$\mathcal{R}4$ should all be in G because it is a PAG. We only need to prove replacing $\mathcal{R}4$ with $\mathcal{R}4-$, the resulting P does not have any arrowhead or tail that does not exist in G. The only change $\mathcal{R}4-$ made is, instead of orienting $c \rightarrow d$, it does not change the mark at c to a tail. If $c \circ\!\!\rightarrow d$ is kept as a circle at c instead of being oriented as a tail, if we check each of $\mathcal{R}1$-$\mathcal{R}4-$, we can see it does not give additional orientations. In other words, if an edge is oriented due to the circle at c as the antecedent of a rule, then that edge would have also been oriented the same way if it were a tail at c.

We then prove the tails oriented by Step 0 must exist in G. Since we modified $\mathcal{R}4$, the only step that learns a tail is $\mathcal{R}1$. $\mathcal{R}1$ orients $a \ast\!\!\rightarrow b \circ\!\!-\!\!\ast c$ to $a \ast\!\!\rightarrow b \rightarrow c$ if a and c are non-adjacent. It must be $b \rightarrow c$ in G because otherwise b would be a collider, and b cannot be in the separating set of a and c. If that is the case, then $\mathcal{R}0$ would have oriented $a \ast\!\!\rightarrow b \leftarrow c$ before $\mathcal{R}1$ is triggered.

Next, we look at Step 1. By Lemma 10, the arrowheads in the pattern P resulting from FCI should all be in G because it is a PAG. Hence we can merge all those arrowheads to P learned from Step 0. After merging, all the arrowheads exist in G, and we just proved all the tails exist in G. □

Lemma 12. *Rules 0-3 in Step 2 Substep 2 of LCDI are sound, i.e., the true DAG G must be in the class $[G]$ that the resulting pattern P represents.*

Proof. Rule 0 does not change P directly, and the correctness will be proved later.

In Rule 1, if b is not a collider, then $a^* \leftarrow a \ast\!\!\leftrightarrow\!\!\ast\, b \ast\!\!\leftrightarrow\!\!\ast\, c \rightarrow c^*$ must be an unblocked path not conditioning on b. However, that gives $\nexists S$, $\sigma_{a^* c^* \cdot S} = 0$, so $adj_E(a, c)$, contradiction. Note we assume both a^* and c^* are generated, but the same reasoning applies to the case where only one AV is generated. Rule 2 can be proved analogously.

In Rule 3, for a^* and d^* to be conditionally independent, all the nodes between a and c need to be conditioned on. If there is a node n, the first node from a to c that is not conditioned on, there will be an unblocked path $a^* \leftarrow a \leftrightarrow \cdots \leftrightarrow n \rightarrow d \rightarrow d^*$, which makes a and d generalized adjacent, contradiction. Therefore, there must be a path from a^* to d^* through b and c. The same reasoning that we used to prove Rules 1 and 2 applies: if c needs to be conditioned on, i.e., $c \in S^*(a, d)$, then it must be a non-collider; otherwise c must be a collider. □

Lemma 13. *Step 2 Substep 3 of LCDI are sound, i.e., the true DAG G must be in the class $[G]$ that the resulting pattern P represents.*

Proof. The correctness of $\mathcal{R}1$ has been discussed. In $\mathcal{R}1$, if the middle node has not been oriented as a collider by previous steps, then it must be a non-collider, because it must be conditioned on to block the path between the two side nodes.

$\mathcal{R}4+$ is a weaker version of the original $\mathcal{R}4$ or $\mathcal{R}4-$. We take out some of the orientations because those orientations are guaranteed for a MAG, but not guaranteed for a DAG. However, the correctness of $\mathcal{R}4+$ can be proved the same way as in Lemma 12. □

Lemma 14. *Step 3 of LCDI are sound, i.e., the true DAG G must be in the class $[G]$ that the resulting pattern P represents.*

Proof. We just have to prove every extraneous edge we marked using Rule 0 must be extraneous in G. Suppose $a \ast\!\!\leftrightarrow\!\!\ast\, b$ is marked. If there is an edge $a \ast\!\!-\!\!\ast\, b$ in G, then there is an unblocked path $a^* \leftarrow a \ast\!\!\leftrightarrow\!\!\ast\, b \rightarrow b^*$ in G, which makes a^* and b^* not conditionally independent, contradiction. □

Theorem 3. *P is the pattern output by LCDI, then the true DAG G that was used to generate the covariance matrix σ_V must be a member of $[G]$ represented by P.*

Proof. The correctness of LCDI results from Lemmas 8-14. □