TECHNICAL REPORT
R-56
June, 2020

Efficient Identification in Linear Structural Causal Models with Auxiliary Cutsets

Daniel Kumor[1] **Carlos Cinelli**[2] **Elias Bareinboim**[3]

Abstract

We develop a a new polynomial-time algorithm for identification of structural coefficients in linear causal models that subsumes previous state-of-the-art methods, unifying several disparate approaches to identification in this setting. Building on these results, we develop a procedure for identifying total causal effects in linear systems.

1. Introduction

Regression analysis is one of the most popular methods used to understand the relationships across multiple variables throughout the empirical sciences. The most common type of regression is linear, where one attempts to explain the observed data by fitting a line (or hyperplane), minimizing the sum of the corresponding deviations. This method can be traced back at least to the pioneering work of Legendre and Gauss (Legendre, 1805; Gauss, 1809), in the context of astronomical observations (Stigler, 1986). Linear regression and its generalizations have been the go-to tool of a generation of data analysts, and the workhorse behind many recent breakthroughs in the sciences, in businesses, and throughout engineering. Based on modern statistics and machine learning techniques, it's feasible to handle regression instances up to thousands, sometimes even millions of variables at the same time (Hastie et al., 2009).

Despite the power entailed by this family of methods, one of its main drawbacks is that it only explains the association (or correlation) between purported variables, while remaining silent with respect to any possible cause and effect relationship. In practice, however, learning about causation is often the main goal of the exercise, sometimes, the very reason one engaged in the data collection and the subsequent analysis in the first place. For instance, a health scientist may be interested in knowing the effect of a new treatment on the survival of its patients, while an economist may attempt to understand the unintended consequences of a new policy on a nation's gross domestic product. If regression analysis doesn't allow scientists to answer their more precious questions, which framework could legitimize such inferences?

The discipline of causal inference is interested in formalizing precisely these conditions, and, more broadly, providing a principled approach to combining data and partial understanding about the underlying generating processes to support causal claims (Pearl, 2000; Spirtes et al., 2000; Bareinboim & Pearl, 2016). One popular framework used to study this family of problems is known as structural causal models (SCMs, for short). Given the pervasiveness of linear regression in data-driven disciplines, we'll focus on the class of linear structural models, following the treatment provided in Wright (1921) and as discussed more contemporaneously in Pearl (2000, Ch. 5).

In this class of SCMs, the set of observed variables are determined by a linear combination of their direct causes and latent confounders (or errors terms). Formally, this is represented as a system of linear equations $X = \Lambda^T X + \epsilon$, where X is a vector of observed variables, ϵ is a vector of latent variables, and Λ is an upper triangular matrix of *direct effects*, otherwise known as path coefficients, whose ijth element, λ_{ij} gives the magnitude of the direct causal effect of x_i on x_j. The errors terms are commonly assumed to be normally distributed, which means that the covariance matrix Σ characterizes the observational distribution. This matrix can be linked to the underlying structural parameters through the system of polynomial equations $\Sigma = XX^T = (I - \Lambda)^{-T}\Omega(I - \Lambda)^{-1}$. Identification then is reduced to finding the elements of Λ that are uniquely determined by the above system. If a structural parameter can be expressed in terms of the elements of Σ alone, it is said to be generically identifiable (Foygel et al., 2012; Drton & Weihs, 2015).

Generic identification can be fully solved using computer algebra as shown in García-Puente et al. (2010). In practice, however, this method has a doubly-exponential computational complexity (Bardet & Chyzak, 2005), becoming

[1]Dept. of Computer Science, Purdue University, West Lafayette, IN, USA [2]Dept. of Statistics, University of California, Los Angeles, CA, USA [3]Dept. of Computer Science, Columbia University, New York, NY, USA. Correspondence to: Daniel Kumor <dkumor@purdue.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

impractical for instances larger than four or five variables (Foygel et al., 2012). It is currently unknown whether the identifiability of an arbitrary structural parameter can be determined in polynomial time.

Instead, most efficient identification algorithms search for patterns in the covariance matrix known to correspond to specific, solvable subsystems of direct effects. The most well-known of such methods is known as instrumental variable (IV) (Wright, 1928). In modern terminology, an "instrument" z relative to a direct effect λ_{xy} needs to be d-separated (Koller & Friedman, 2009) from y, while it cannot be d-separated from x in the modified graph where the target edge $x \rightarrow y$ is removed (Pearl, 2000). The existence of such a variable means that $\lambda_{xy} = \frac{\sigma_{zy}}{\sigma_{zx}}$, and, therefore, is uniquely determined by the observational distribution.

The IV and its generalizations, the conditional IV (cIV), are heavily exploited in the literature, particularly in the field of econometrics (Fisher, 1966; Bowden & Turkington, 1984). Despite its success, many identifiable effects in a linear system cannot be found with IVs and cIVs. Therefore, the past two decades has witnessed a push in the development of successively more sophisticated identification methods.

Two promising avenues towards efficiently solving generic identification are conditional Auxiliary Variables (cAV) (Chen et al., 2017) and Instrumental Cutsets (IC) (Kumor et al., 2019), both of which provide poly-time algorithms encompassing previous works such as the half-trek criterion (HTC) (Foygel et al., 2012), the generalized half-trek criterion (gHTC) (Chen, 2016; Weihs et al., 2018), auxiliary variable sets (AVS) (Chen et al., 2016), and conditional instrumental variables (cIV) (Van der Zander et al., 2015).

Another class of graphical criteria has no known efficient algorithm to date. These methods currently require an exponential number of steps. One such algorithm, the generalized instrumental set (gIS) (Brito & Pearl, 2002) and its generalization, the quasi-AV set (qAVS) (Chen et al., 2017), have thus far eluded characterization. The perceived difficulty of finding gIS (Tian, 2007) is compounded by a proof that given a candidate set of instruments, finding whether conditioning sets exist to make a gIS is NP-hard (Van der Zander & Liskiewicz, 2016). This was further exasperated when Kumor et al. (2019) proved that finding simplified conditional instrumental sets (scIS) is also NP-hard.

We roughly summarize these methods in Fig. 1, even though it lies outside the scope of this paper to survey this rich literature. It can be seen that the literature is splintered among several competing methods, with the state-of-the-art in poly-time identification being IC or cAV, depending on the setting. It's not currently known how these methods compare to qAVS, which has undetermined complexity.

The main goal of this paper is to provide an unifying treatment of the threads and corresponding algorithms found in this literature, under the umbrella of a single, efficient algorithm. In particular, our contributions are:

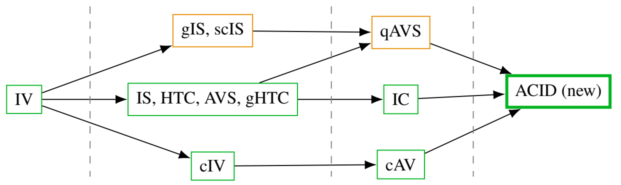


Figure 1. Summary of the discussed identification methods. $a \rightarrow b$ means all methods in b subsume all methods in a. Green boxes represent existence of polynomial-time algorithms, orange ones are undetermined or NP-hard. ACID is the newly proposed algorithm.

- We develop the Auxiliary Cutset Identification Algorithm (ACID), which runs in polynomial-time, and unifies and *strictly subsumes* existing efficient identification methods (such as IC and cAV) as well as conditioning-based methods with unknown complexity (qAVS).

- We design a strategy for identification of total effects based on the decomposition of the target query into smaller, more manageable effects that can be effectively and systematically solved by algorithms designed for direct effects.

2. Preliminaries

The causal graph of an SCM is defined as a triple $G = (V, D, B)$, representing the nodes, directed, and bidirected edges, respectively. A linear SCM has a node v_i for each variable x_i, a directed edge between v_i and v_j for each nonzero λ_{ij}, and a bidirected edge between v_i and v_j whenever there is latent confounding between the variables, i.e., nonzero $\epsilon_{ij} = \sigma_{\epsilon_i \epsilon_j}$ (Fig. 2a). When clear from the context, we will use λ_{ij} and ϵ_{ij} to refer to the corresponding directed and bidirected edges in the graph. In keeping with other works, we define $Pa(x_i)$ as the set of parents of x_i, $An(x_i)$ as ancestors of x_i, $De(x_i)$ as descendants of x_i, and $Sib(x_i)$ as variables connected to x_i with bidirected edges (i.e., variables with latent common causes).

A path in the graph is said to be "active" conditioned on a (possibly empty) set W if it contains a collider only when $b \in W \cup An(W)$, and if it does not otherwise contain vertices from W (see d-separation (Koller & Friedman, 2009)). Active paths without conditioning do not contain colliders, and are referred to as treks (Sullivant et al., 2010). The covariances of observed variables have a graphical interpretation in terms of a sum over all treks between nodes in the causal graph, namely $\sigma_{xy} = \sum \pi(x, y)$, where π is the product of structural parameters along the trek.

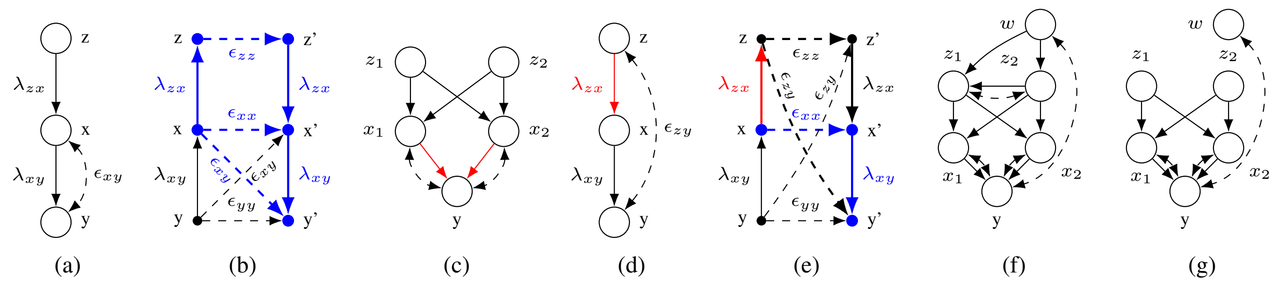


Figure 2. In (a), z is an instrument for λ_{xy}. (b) shows a directed flow graph encoding the covariances between variables, with σ_{xy} highlighted in blue. (c) λ_{x_1y} cannot be solved using a conditional instrument, but can be approached using the instrumental set $\{z_1, z_2\}$. (d) knowledge of λ_{zx} can be used to remove the backdoor path from x to y, shown in (e)'s flow graph (f) λ_{x_1y} cannot be solved by either IC nor cAV, but can be solved using gIS. (g) After applying Tian's model decomposition to the graph in (f), it becomes equivalent to the one in c, making the desired parameter efficiently solvable.

While methods such as Wright's rules of path analysis are commonly used for reading covariances directly from the causal graph, we follow the identification literature and define a "flow graph", which explicitly encodes the treks between nodes in its directed paths, allowing a direct algorithmic approach to path analysis.

Definition 2.1. *(Sullivant et al., 2010; Weihs et al., 2018; Kumor et al., 2019) The flow graph of $G = (V, D, B)$ is the graph with vertices $V \cup V'$ containing the edges:*

- *$j \to i$ and $i' \to j'$ with weight λ_{ij} if $i \to j \in D$*
- *$i \to i'$ with weight ϵ_{ii} for all $i \in V$*
- *$i \to j'$ with weight ϵ_{ij} if $i \leftrightarrow j \in B$*

This graph is referred to as G_{flow}. The nodes without $'$ are called "source", or "top" nodes, and the nodes with $'$ are called "sink" or "bottom" nodes.

As an example, consider the instrumental variable graph of Fig. 2a. Each trek corresponds to a directed path in the flow graph shown in Fig. 2b. Summing over all paths from x to y' (in blue), we obtain the covariance between these two variables,

$$\sigma_{xy} = \lambda_{zx}\epsilon_{zz}\lambda_{zx}\lambda_{xy} + \epsilon_{xx}\lambda_{xy} + \epsilon_{xy} = \sigma_{xx}\lambda_{xy} + \epsilon_{xy}$$

Reading covariances from the flow graph can help in visualizing the IV: $\frac{\sigma_{zy}}{\sigma_{zx}} = \frac{\epsilon_{zz}\lambda_{zx}\lambda_{xy}}{\epsilon_{zz}\lambda_{zx}} = \lambda_{xy}$. Finally, in Fig. 2c, there is no single instrument, but one can use the instrumental set $\{z_1, z_2\}$ to construct a solvable system of equations (Brito & Pearl, 2002),

$$\begin{aligned}\sigma_{z_1y} &= \sigma_{z_1x_1}\lambda_{x_1y} + \sigma_{z_1x_2}\lambda_{x_2y}\\ \sigma_{z_2y} &= \sigma_{z_2x_1}\lambda_{x_1y} + \sigma_{z_2x_2}\lambda_{x_2y}\end{aligned} \quad (1)$$

To simplify discussion of paths in the graph, we define the *partial effect* of a on b avoiding set C, denoted as $\delta_{ab.C}$, as the causal effect of a on b when holding all variables in C constant ($\delta_{ab.C} = \frac{\partial}{\partial a}\mathbf{E}[b \mid do(a), do(C)]$). This corresponds to the sum of all products of direct effects along the directed paths from a to b that do not cross any nodes in C. In particular, two special cases are worth noting. First, when $C = \emptyset$, then $\delta_{ab.C} = \delta_{ab}$ is the *total effect* of a on b. When $C = Pa(a)$, we recover the direct effect $\delta_{ab.C} = \lambda_{ab}$.

2.1. Auxiliary Variables

One can leverage direct effects that were previously identified to create new variables to help with the identification of further edges. For example, in Fig. 2d, λ_{zx} can be trivially identified with the regression coefficient of z on x, $\lambda_{zx} = \frac{\sigma_{zx}}{\sigma_{zz}}$. Once λ_{zx} is known, one can create an **auxiliary variable** (AV) $x^* = x - \lambda_{zx}z$ subtracting out the direct effect of z. This new variable behaves as if the edge λ_{zx} did not exist in the graph, eliminating the backdoor path from x to y, and allowing the identification of the direct effect of x on y with the regression coefficient of x^* on y, i.e, $\lambda_{xy} = \frac{\sigma_{x^*y}}{\sigma_{x^*x}}$ (Chen et al., 2016). This phenomenon can also be observed in the flow graph of Fig. 2d, shown in Fig. 2e. The covariance of x^* with y reads $\sigma_{x^*y} = \sigma_{xy} - \lambda_{zx}\sigma_{zy} = \epsilon_{xx}\lambda_{xy}$. That is, the operation to create x^* effectively removes the source edge λ_{zx} (red) from the flow graph. Next, dividing σ_{x^*y} by $\sigma_{x^*x} = \sigma_{xx} - \lambda_{zx}\sigma_{zx} = \epsilon_{xx}$ gives λ_{xy}.

2.2. Trek Systems & Determinants

Instrumental variables and instrumental sets can be generalized in the flow graph by exploiting properties of determinants of the minors of the covariance matrix (Sullivant et al., 2010). Denote by $\Sigma_{z_1z_2,x_1x_2}$ the minor of the covariance matrix with columns z_1, z_2 and rows x_1x_2. This gives us $\det \Sigma_{z_1z_2,x_1x_2} = \sigma_{z_1x_1}\sigma_{z_2x_2} - \sigma_{z_2x_1}\sigma_{z_1x_2}$. As shown by Gessel & Viennot (1989), the value of this determinant can be read directly from the flow graph as the sum of non-intersecting sets of paths (i.e, paths which do not share any vertices) between z_1, z_2 and x'_1, x'_2, multiplied by the sign of the permutations (see Gessel & Viennot (1989) for details). This is due to the fact that terms that come from intersecting paths cancel out when computing the determinant, leaving only terms corresponding to non-intersecting paths. We therefore have $\det \Sigma_{z_1 z_2, x_1 x_2}$ as the product of non-intersecting paths between $z_1 \to x'_1$ and $z_2 \to x'_2$, with the non-intersecting paths between $z_1 \to x_2$ and $z_2 \to x_1$ subtracted out. If any such path set exists, the determinant can be shown to be generically non-zero.

For illustration, consider the flow graph of Fig. 2c, which is shown in Fig. 3. There is only one non-intersecting path set between $z_1 \to x_1$ and $z_2 \to x_2$, namely $\epsilon_{z_1 z_1}\lambda_{z_1 x_1}$ and $\epsilon_{z_2 z_2}\lambda_{z_2 x_2}$. Similarly, the non-intersecting path set between $z_1 \to x_2$ and $z_2 \to x_1$ consists of the paths $\epsilon_{z_1 z_1}\lambda_{z_1 x_2}$ and $\epsilon_{z_2 z_2}\lambda_{z_2 x_1}$. The determinant is given by

$$\begin{aligned}\det \Sigma_{z_1 z_2, x_1 x_2} = &\ (\epsilon_{z_1 z_1}\lambda_{z_1 x_1})(\epsilon_{z_2 z_2}\lambda_{z_2 x_2}) \\ &- (\epsilon_{z_1 z_1}\lambda_{z_1 x_2})(\epsilon_{z_2 z_2}\lambda_{z_2 x_1}) \qquad (2)\end{aligned}$$

Weihs et al. (2018) realized that this property can be exploited for the identification of structural parameters. Solving for $\det \Sigma_{z_1 z_2, y x_2}$, the same non-intersecting paths exist between the sets as shown in Fig. 3, but all paths to x'_1 are now extended to y'. This yields

$$\det \Sigma_{z_1 z_2, y x_2} = \lambda_{x_1 y} \det \Sigma_{z_1 z_2, x_1 x_2} \qquad (3)$$

allowing identification of $\lambda_{x_1 y} = \frac{\det \Sigma_{z_1 z_2, y x_2}}{\det \Sigma_{z_1 z_2, x_1 x_2}}$, which formally recovers Cramer's rule solution of Eq. (1) for $\lambda_{x_1 y}$.

2.3. Model Decomposition

One can also *decompose* a graph into smaller sub-graphs. First proposed by Tian (2005), the decomposition hinges upon the concept of a **c-component**, consisting of a set of variables connected through paths consisting solely of bidirected edges. Consider Fig. 2f. Here, x_1 has a bidirected edge to y, which in turn is connected to x_2 and w via bidirected edges. This makes $\{w, x_1, x_2, y\}$ a c-component. Likewise, $\{z_1, z_2\}$ is also a c-component.

Given a c-component, we can define a subgraph consisting of the nodes in the c-component and its direct parents, with all other variables and edges removed. This subgraph is called a "mixed component" of G.

Definition 2.2. *(Tian, 2005; Drton & Weihs, 2015) Given $G = (V, D, B)$, let $C_1, ..., C_k \subset V$ be the unique partitioning of V where $v, w \in C_i$ iff $\exists$ path from v to w composed only of bidirected edges, and let $V_i = C_i \cup Pa(C_1)$, $D_i = \{v \to w \in G | v \in V_i, w \in C_i\}$ and $B_i = \{v \leftrightarrow w \in G | v, w \in C_i\}$. Then $G_i = (V_i, D_i, B_i)$, $i = 1...k$ are the* ***mixed components*** *of G.*

The mixed component of $\{w, x_1, x_2, y\}$ in Fig. 2f is shown in Fig. 2g. Whereas existing efficient methods fail to identify

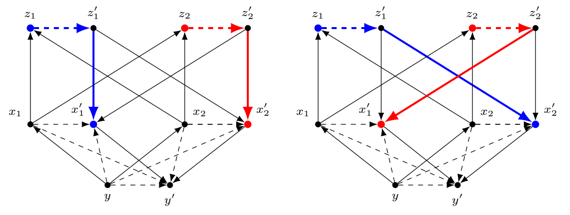


Figure 3. The flow graph of Fig. 2c, showing the two non-intersecting path sets from z_1, z_2 to x_1, x_2

$\lambda_{x_1 y}$ in Fig. 2f, it is easily identifiable in Fig. 2g. As shown by Tian (2005), this means the effect is also identifiable in the original graph.

3. Identification with the Auxiliary Cutset

In this section, we develop a polynomial-time algorithm that subsumes the state-of-the-art for efficient identification in linear SCM. We begin by defining a new type of auxiliary variable, which can help with the identification of new coefficients in the model. We then devise an identification criterion for *partial effects*, and show how to use it to efficiently create these new AVs. Finally, we show how our results can be used to recursively identify direct effects.

3.1. Auxiliary variables using total and partial effects

Standard auxiliary variables enable the use of previously identified direct effects to remove edges from the flow graph. In some cases, such as in Fig. 2d, this allows us to directly identify a target parameter, bypassing the need to search for a conditioning set. However, in many cases, auxiliary variables using only identifiable direct effects are not sufficient for this task.

An example of such a model is given in Fig. 2f. Here, although the generalized instrumental set $\{z_1, z_2\}$ *conditional* on w is sufficient for identifying $\lambda_{x_1 y}$ and $\lambda_{x_2 y}$, we cannot achieve the same result with AVs. While $z_2^* = z_2 - \lambda_{w z_2} w$ can be computed (because $\lambda_{w z_2}$ is identified), the AV $z_1^* = z_1 - \lambda_{w z_1} w - \lambda_{z_1 z_2} z_2$ cannot, since that requires the identification of both $\lambda_{w z_1}$ and $\lambda_{z_1 z_2}$. This suggests that there is something missing from AVs that rely solely on direct effects.

The source of the issue can be revealed in the mixed component of the model (Fig. 2g). Here, w is disconnected from z_1 and z_2, which becomes equivalent to the model in Fig. 2c. In the mixed component, $\lambda_{x_1 y}$ and $\lambda_{x_2 y}$ can be easily identified using an *unconditional* instrumental set $\{z_1, z_w\}$. This leads to the realization that it may not be necessary to identify the direct effects $\lambda_{w z_1}$ and $\lambda_{z_1 z_2}$ to disconnect z_1 from w—it suffices to identify the *total effect* of w on z_1. As it happens, this effect is easily computable

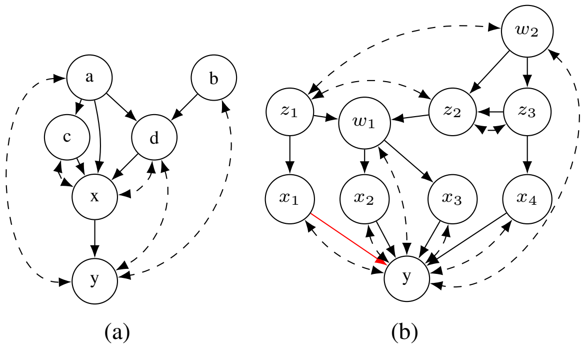


Figure 4. (a) A naïve application of total-effect AVs can't be used here. (b) Only ACID can solve for λ_{x_1y}.

with the regression coefficient of w on z_1, i.e, $\delta_{wz_1} = \frac{\sigma_{wz_1}}{\sigma_{ww}}$. We can now create a new type of AV, $z_1^\dagger = z_1 - \delta_{wz_1}w$, which indeed behaves as if z_1 had no path to w. Combining $z_1^\dagger$ with the standard AV z_2^* leads to a system of equations that can be solved for λ_{x_1y} and λ_{x_2y},

$$\sigma_{z_1^\dagger y} = \sigma_{z_1y} - \delta_{wz_1}\sigma_{wy} = \lambda_{x_1y}\sigma_{z_1^\dagger x_1} + \lambda_{x_2y}\sigma_{z_1^\dagger x_2}$$
$$\sigma_{z_2^* y} = \sigma_{z_2y} - \lambda_{wz_2}\sigma_{wy} = \lambda_{x_1y}\sigma_{z_2^* x_1} + \lambda_{x_2y}\sigma_{z_2^* x_2}$$

One needs to be careful when generalizing the results of this example—sometimes, a naïve subtraction of total effects can backfire. Consider, for instance, the model of Fig. 4a, and suppose we want to use x as an AV for the target query λ_{xy}. Note we can identify the total effects δ_{ax} and δ_{dx}. But also note that in the "naïve AV" $x^\ddagger = x - \delta_{dx}d - \delta_{ax}a$, subtracting out *both* total effects, *does not* remove the backdoor paths of x with y,

$$\sigma_{x^\ddagger y} = \sigma_{xy} - \delta_{ax}\sigma_{ay} - \delta_{dx}\sigma_{dy} = \lambda_{xy}\sigma_{x^\ddagger x} - \delta_{dx}\lambda_{ad}\epsilon_{ay}$$

This happens because the total effect of a already includes parts of the total effect of d, and thus when constructing the "naïve AV" $x^\ddagger$ we subtracted the path $\lambda_{ad}\lambda_{dx}$ twice.

One way to avoid this problem is to use only the portions of the effect of a on x that do not pass through d. That is, instead of subtracting out the total effect δ_{ax}, we subtract out the *partial effect* $\delta_{ax.d}$, leading to $x^\dagger = x - \delta_{dx}d - \delta_{ax.d}a$. Indeed, as desired, this removes all backdoor paths,

$$\sigma_{x^\dagger y} = \sigma_{xy} - \delta_{ax.d}\sigma_{ay} - \delta_{dx}\sigma_{dy} = \lambda_{xy}\sigma_{x^\dagger x}$$

In other words, by using only paths that do not intersect any other variable subtracted in the AV, we can avoid subtracting any path more than once, allowing us to effectively remove all of x's backdoor paths to both a and d at the same time. We formalize this idea in Theorem 3.1.

Theorem 3.1. *Given a variable x, and a subset $\mathbf{C}$ of the ancestors of x, the covariance of the auxiliary variable $x^\dagger = x - \sum_i \delta_{c_ix.C} \times c_i$ with variable v can be determined by the sum of paths from x to v' in G_{flow} with source edges $\lambda_{c_id_j}$ removed where $c_i \in \mathbf{C}$ and $d_j \in An(x)$.*

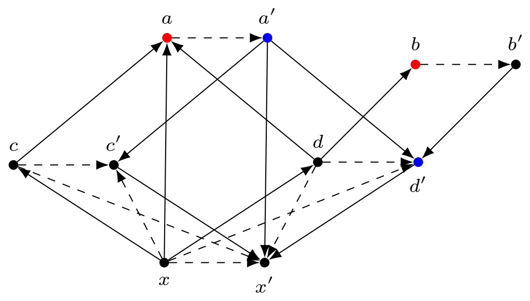


Figure 5. The flow graph of Fig. 4a, excluding y. To find the partial effects $\delta_{ax.d}$ and $\delta_{dx.a}$ (a', d' in blue), we can use the partial-effect instrumental set a, b (red).

3.2. Instrumental sets for partial-effects

Theorem 3.1 gives us a principled way to incorporate knowledge of partial effects into the flow graph in order to help existing identification algorithms. A natural question now arises: how can we identify those partial effects? In this subsection, we demonstrate how a modified version of instrumental sets can solve this task. In particular, we exploit the same property that was used to identify a direct effect in the example of Eq. (3).

Continuing with the model of Fig. 4a, we want to identify $\delta_{dx.a} = \delta_{dx}$ and $\delta_{ax.d}$. To help with understanding the general approach, we will operate on the flow graph of Fig. 4a excluding y, shown in Fig. 5. We have placed a', d' (blue) in the sink nodes. Notice that the paths from a' to x' form δ_{ax}. All paths from a' to x' that do not intersect with d' form $\delta_{ax.d}$. Likewise, the paths from d' to x' form δ_{dx}, which in this case is the same as $\delta_{dx.a}$. Our goal is to exploit the non-intersection property of paths in the determinant of a trek system to automatically find all paths from a to x that do not pass through d.

Observe that a and b are two **candidate instruments** for x, that is, they are non-descendants of $\{x\} \cup Sib(x)$. Furthermore, all paths from the source nodes a and b (red) to x' in the flow graph cross either a' or d'. Computing the determinant between a, b and a', d' gives $\det \Sigma_{ab,ad} = (\epsilon_{aa})(\epsilon_{bb}\lambda_{bd})$, since there is only one valid path set, $a \to a'$ and $b \to d'$.

Next, replace a' with x' in the determinant. That is:

$$\begin{aligned}\det \Sigma_{ab,xd} &= (\epsilon_{aa}(\lambda_{ax} + \lambda_{ac}\lambda_{cx}))(\epsilon_{bb}\lambda_{bd}) \\ &= \delta_{ax.d} \det \Sigma_{ab,ad} \qquad (4)\end{aligned}$$

We have therefore found that $\delta_{ax.d} = \frac{\det \Sigma_{ab,xd}}{\det \Sigma_{ab,ad}}$. Similarly, $\delta_{dx.a} = \frac{\det \Sigma_{ab,ax}}{\det \Sigma_{ab,ad}}$. What we have created here is the analog of a standard instrumental set, which uses the set $\{a, b\}$ as instruments to identify the partial effects of a and d on x.

The formalization of instrumental sets for identifying partial

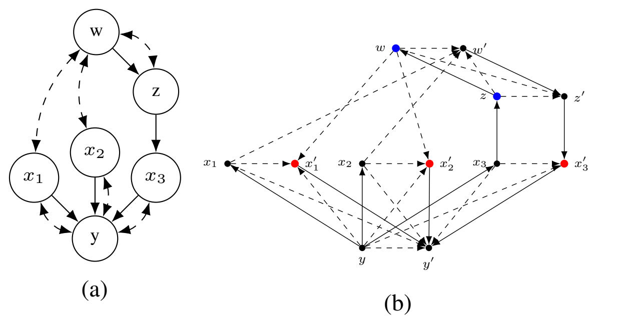


Figure 6. In this graph, w and z are both candidate instruments, but the min-cut between w, z (blue) and x_1', x_2', x_3' (red) is w, z' - with w as a source node.

effects is given in Definition 3.1 and Theorem 3.2.

Definition 3.1. *Given target node x, a set $C \subset An(x)$, and a set Z such that $|Z| = |C|$, then, if there is a non-intersecting path-set between source nodes of Z and sink nodes of C in G_{flow}, and all paths from source nodes Z to sink node x' cross through the sink nodes of C, the set Z is said to be a* ***Partial-Effect Instrumental Set*** *(PEIS) relative to C and x.*

Theorem 3.2. *If Z is a PEIS relative to C and x, then the partial effects of C on x are identified, and given by $\delta_{c_i x.C} = \frac{\det \Sigma_{Z,C^x_{-i}}}{\det \Sigma_{Z,C}}$, where C^x_{-i} is the ordered set C with c_i replaced by x.*

Whenever there exists a PEIS Z relative to C and x, we call C a **feasible ancestral cutset** of x, because C cuts x from Z in the flow graph. Finally, we would like to emphasize that, while in this work we mainly use the PEIS for the purpose of constructing AVs, this is a general criterion for finding partial effects, and can be used independently for other purposes.

3.3. Auxiliary cutset: "best" feasible ancestral cutset

In general, given a target node x for which we want to create an auxiliary variable $x^\dagger$, there will be multiple feasible ancestral cutsets we can choose from. For instance, in Fig. 4a, d alone is a feasible ancestral cutset (using b as an instrument). Clearly, however, an auxiliary variable constructed by subtracting the effect of d alone cuts x from strictly less variables than using $\{a, d\}$. Moreover, an $x^\dagger$ constructed in this way is not independent of y (a path through a still exists), and can no longer be used as an AV to identify λ_{xy}. It is useful, therefore, to define a notion of the *best* feasible ancestral cutset C for generating an auxiliary variable $x^\dagger$. This is called the **auxiliary cutset**.

Definition 3.2. *The* ***auxiliary cutset*** *(AC) is the feasible ancestral cutset C of x such that sink nodes of C intersect all paths from sink nodes of C' to x' for all feasible ancestral cutsets C' for x in G.*

Algorithm 1 - AC: is given a graph, target vertex x, a set of candidate instruments (which can themselves be AVs), a set of identified structural parameters, and returns the Auxiliary Cutset for x

```
Input: $G, x, Z, \Lambda$
$G_{hf} \leftarrow$ AuxHalfFlowGraph$(G, x, \Lambda)$
$C \leftarrow \emptyset$
repeat
    $Z \leftarrow \{z \in Z : \text{UnremovedAncestors}(z) \cap C = \emptyset\}$
    $C \leftarrow$ vertex min-cut closest to $x$ between $Pa(x)$ and $Z$
until UnremovedAncestors$(Z) \cap C = \emptyset$
$Z' \leftarrow \{z_i \in Z : z_i$ has non-zero flow to $c_i \in C\}$
return $(Z', C)$
```

Definition 3.2 ensures that the AC of x results in an auxiliary variable $x^\dagger$ that has all the removed ancestral paths of any other possible auxiliary varible $x^{\dagger'}$ removed. In other words, for every other feasible auxiliary variable $x^{\dagger'}$, we know that the $x^\dagger$ constructed using the AC is the "best", meaning that, if a path is removed using any other feasible ancestral cutset, it is also removed using the auxiliary cutset.

To find the AC, we follow a procedure similar to the one used in Kumor et al. (2019). The core idea can once again be demonstrated on Fig. 5. In this flow graph, only a and b are source nodes whose paths to x all go through the sink nodes of $Pa(x)$. This means that a and b are both "candidate instruments"—only candidate instruments can possibly be instruments of a feasible ancestral cutset. We then run a vertex min-cut algorithm (Picard & Queyranne, 1982) from $\{a, b\}$ to the sink nodes of parents of x, namely $\{c', d'\}$, and find the min-cut that is *closest* to x. In this case, the min-cut is a', d', meaning that $\{a, d\}$ is the auxiliary cutset of x.

By using the closest vertex min-cut C between candidate instruments Z and sink nodes of parents of x, we guarantee all the conditions of Definition 3.1 are satisfied automatically by Z' and C, with $Z' \subseteq Z$, except for the requirement that C consists entirely of sink nodes. An example where this is violated is given in Fig. 6—the min-cut there includes source node w. In such cases, we know that the associated node is not part of any possible AC, so we can remove it from candidacy, and rerun the min-cut algorithm. After removing w, and then z, we are left with no possible AC in Fig. 6. The general version of this procedure (which includes concepts from Section 3.4) is implemented in Algorithm 1, and always finds the AC if it exists.

Theorem 3.3. *If there exists a feasible ancestral cutset for x in G, then the AC exists, and is found by Algorithm 1.*

3.4. Putting everything together: the *ACID* Algorithm

The final question remaining is: how can we use this newly generated AV $x^\dagger$ for identification? We want to both use it recursively in Algorithm 1, and within our identification algorithm (IC), to identify direct effects. When the set of candidate instruments consists of original variables we can easily find the AC using the standard flow graph, as it was done in the example in Fig. 5. However, once the candidate instruments have their own auxiliary cutsets, with each candidate's cutset being specific to that candidate, we run into difficulties trying to encode non-intersecting paths. To understand why, we will use the concept of "unremoved ancestors"

Definition 3.3. *Let y be a target node with auxiliary cutset C. The **unremoved ancestors** of y are all $v \in An(y)$ such that there exists a directed path from v to y in G that does not cross any $c \in C$.*

When using multiple AVs at once, e.g, $z_1^\dagger$ and $z_2^\dagger$, where each AV has source-paths in the flow graph to its own unremoved ancestors, some of which might be shared, it is unclear how to efficiently find a path set from both AVs at once that does not intersect. Theorem 3.4 shows we do not need to encode paths in the source nodes at all.

Theorem 3.4. *Let A be the unremoved ancestors of the AV $x^\dagger$. If $x^\dagger$ is a candidate instrument for y, then all elements of the AV $a^\dagger$ of $a \in A$ are also candidate instruments for y.*

Therefore, since all the unremoved ancestors of a candidate instrument are candidate instruments themselves, any candidate with a path in the source nodes can be switched with the candidate at which the path crosses over to the sink nodes. This means we only need to encode edges from source nodes to sink nodes, which can be done with the Auxiliary Half-Flow Graph (Definition 3.4).

Definition 3.4. *The **auxiliary half-flow graph** of $G = (V, D, B)$ given target node y and set of identified edges Λ is the graph with vertices $V \cup V'$ containing the edges:*

- *$i' \to j'$ with weight λ_{ij} if $i \to j \in V$ and $j \neq y$*
- *$i' \to y'$ with weight λ_{iy} if $i \to y \in V$ and $\lambda_{iy} \notin \Lambda$*
- *$i \to i'$ with weight ϵ_{ii} for all $i \in V$*
- *$i \to j'$ with weight ϵ_{ij} if $i \leftrightarrow j \in B$*

This graph is referred to as G_{hf}. The nodes without $'$ are called "source" nodes, and the nodes with $'$ are called "sink" or "bottom" nodes.

This completes the tools needed to specify the ACID algorithm (Algorithm 2), which internally uses a version of the IC algorithm (Kumor et al., 2019) adapted to make use of partial-effect AVs and Half-Flow graphs.[1]

Algorithm 2 - ACID: Given a graph, returns a set of identifiable structural parameters.

```
**Input:** $G$
$\Lambda_{id} \leftarrow \emptyset$
$v^\dagger = v\ \forall$ vertices $v \in G$
**repeat**
  **for all** vertices $v \in G$ in topological order **do**
    $Z \leftarrow \{z^\dagger : \textsc{UnremovedAncestors}(z^\dagger) \cap (Sib(y) \cup \{y\}) = \emptyset,\ \forall z \in G\}$
    $G_{hf} \leftarrow \textsc{AuxHalfFlowGraph}(G, x, \Lambda_{id})$
    $\Lambda_{id} \leftarrow \Lambda_{id} \cup IC(G_{hf}, v, Z)$
    $v^* = v - \sum_{\lambda_{av} \in \Lambda_{id}} \lambda_{av}$
    $(Z', C) \leftarrow AC(G, v, Z, \Lambda_{id})$
    $v^\dagger = v^* - \sum_{c_i \in C} \frac{\det \Sigma_{Z', C_i^{v^*}}}{\Sigma_{Z', C}} c_i$
  **end for**
**until** no change in this iteration
**return** $\Lambda_{id}$
```

ACID unifies the state-of-the-art for identification in linear SCMs. Moreover, the AC's ability to block certain ancestors turns out to be the missing piece needed for auxiliary variable methods to finally overtake methods based on conditioning. Methods built upon conditioning such as the gIS and qAVS have undetermined complexity, with several NP-Hardness results for similar methods. ACID is the first efficient identification algorithm that subsumes these approaches, obviating the discussion of their computational complexity.

Theorem 3.5. *If λ_{ab} is identifiable with either the cAV, IC, or qAVS criteria, then it is identifiable with ACID.*

4. Decomposition of total effects

The ACID algorithm identifies individual direct effects. It does not include total effects, which play an important role in virtually all causal inference tasks, such as policy-making, model testing, z-identification, and sensitivity analysis (Pearl, 2000; Bareinboim & Pearl, 2012; Chen et al., 2017; Cinelli et al., 2019; Lee et al., 2019; Cinelli & Hazlett, 2020). Unlike direct effects, the identification of total effects in linear models has not received as much attention and existing approaches fall broadly into two categories.

The first approach is to appeal to the foundational methods of identification, such as the instrumental variable, the front-door, or the back-door criterion (or, more generally, the do-calculus) (Pearl, 2000; Tian, 2004). While sound, these methods ignore recent advances in the linear identi-

[1]For details of the modified IC, refer to the appendix.

fication literature. A second approach is to decompose the total effect of x on y into the sum of structural parameters along all directed paths from x to y, and use state-of-the-art identification algorithms for direct effects to identify *all* structural parameters along these paths. This is sub-optimal, as it misses cases in which the total effect is identifiable, but some individual parameters are not.

To bridge the gap between these two extremes, we derive a decomposition of total effects that relies on the identification of only some of the direct effects of which it is composed. The method makes use of ancestor and c-component decompositions to recursively break the total effect into a set of partial effects and direct effects, which can then be attacked with current identification algorithms for linear models.

Suppose our target query is the total effect δ_{xy} in Fig. 7a. This effect is *not* identifiable non-parametrically, as there are bidirected edges between x and its children. Second, an approach based on current state-of-the-art methods that relies on identifying all direct effects of δ_{xy} would equally fail, since no existing method can identify λ_{ce}.

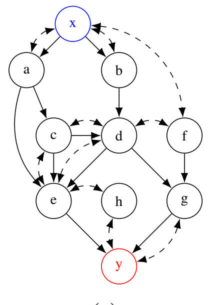
We begin by defining the "top boundary" of a mixed component as the parents of variables in a c-component that are in other c-components.

Definition 4.1. *Given graph G, with c-components $C_1, ..., C_k$, the* ***top boundary*** *$Tb(C_i)$ of the c-component C_i is defined as $Tb(C_i) = Pa(C_i) \setminus C_i$.*

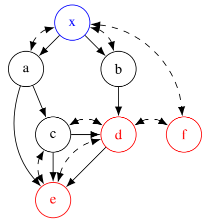
The concept of top boundary is useful for two reasons: first, we can always identify the "total effect" in the mixed component G' of its top boundary on any other node in the mixed component (this "total effect" in G' may correspond to a partial effect in the original model G); second, the total effect of x on y can be decomposed as the sum of the total effect of x in the nodes of the top boundary (in G) times the "total effect" of the top boundary nodes in y in the mixed component. This is formalized in Theorem 4.1.

Theorem 4.1. *Let $G_{An(y)}$ be the graph G with the non ancestors of y removed. Let C_y be the c-component of y in $G_{An(y)}$ and G' its corresponding mixed component. Then, if $x \in C_y$, the total effect of x on y, δ_{xy}, can be decomposed as, $\delta_{xy} = \delta'_{xy} + \sum_{b \in Tb(C_y)} \delta_{xb}\delta'_{by}$ otherwise, if $x \notin C_y$, we have, $\delta_{xy} = \sum_{b \in Tb(C_y)} \delta_{xb}\delta'_{by}$ where $Tb(C_y)$ is the top boundary of the c-component C_y and δ'_{by} is the total effect of node $b \in Tb(C_y)$ on y in the mixed component G'. Moreover, all δ'_{by} for $b \in Tb(C_y)$ are identified.*

The recursive application of this idea enables us to iteratively identify parts of the total effect via a combination of ancestral and c-component decompositions, leaving only a portion of the path to be identified using algorithms specialized for direct effects. In our example, note that h is not an ancestor of y, therefore $G_{An(y)}$ does not include h. This allows us to



(a)



(b)

Figure 7. In (a), we can reduce δ_{xy} into queries on $\delta_{xe}, \delta_{xd}, \delta_{xf}$ in (b), and then to λ_{xa} and λ_{xb} by iteratively decomposing the model.

Algorithm 3 - TED: Given a graph and a target total-effect δ_{xy}, returns a set of direct effects that suffice to identify δ_{xy}

```
Input: $G, \delta_{xy}$
$G_{An} \leftarrow An(y)$
$G' \leftarrow \text{MixedComponent}(G_{An}, y)$
$B \leftarrow \text{TopBoundary}(G') \cap An(y, G')$
if $x \in G'$ and $x \notin B$ then
    return $\{\lambda_{ab} : \forall \lambda_{ab} \in \delta'_{xy}\} \cup \bigcup_{b \in B} \text{TED}(G, \delta_{xb})$
else
    return $\bigcup_{b \in B} \text{TED}(G, \delta_{xb})$
end if
```

decompose the pruned graph into the the mixed component G' of the c-component $\{y, g\}$. Since the total effect of x on y needs to necessarily pass through the top boundary of G', we have that, as per Theorem 4.1, the total effect can be decomposed as $\delta_{xy} = \delta_{xe}\delta'_{ey} + \delta_{xd}\delta'_{dy} + \delta_{xf}\delta'_{fy}$, and all direct effects starting from the top boundary in the mixed component (δ') can be identified. The query δ_{xy} is then broken down into three smaller subqueries, δ_{xe}, δ_{xd}, δ_{xf}.

Now we can apply Theorem 4.1 for each of the remaining queries (shown in Fig. 7b). Pruning the non-ancestors of f leaves us with just f, and $\delta_{xf} = 0$. Looking at e, note that f is not its ancestor, and the c-component decomposition allows identification of δ'_{ae} and δ'_{be}, with resulting subqueries δ_{xa} and δ_{xb}. Applying the same logic to d leads us to identify δ'_{ad} and δ'_{bd}, with identical remaining subqueries. This leaves only two directed edges left to be identified $\delta_{xa} = \lambda_{xa}$ and $\delta_{xb} = \lambda_{xb}$. We have thus reduced the identification of the total effect of δ_{xy} to the identification of λ_{xa} and λ_{xb} only, both of which can be solved using f as an instrumental variable. The full algorithm for the total effect decomposition is given in Algorithm 3, which given a target total-effect δ_{xy}, returns a set of direct effects that are sufficient for identifying the desired quantity.

5. Conclusion

We developed an efficient algorithm for linear identification that subsumes the current state-of-the-art, unifying disparate approaches found in the literature. In doing so, we also introduced a new method for identification of partial effects, as well as a method for exploiting those partial effects via auxiliary variables. Finally, we devised a novel decomposition of total effects allowing previously incompatible methods to be combined, leading to strictly more powerful results.

Acknowledgements

Kumor and Bareinboim are supported in parts by grants from NSF IIS-1704352 and IIS-1750807 (CAREER).

References

Bardet, M. and Chyzak, F. On the complexity of a gröbner basis algorithm. In *Algorithms Seminar*, pp. 85–92, 2005.

Bareinboim, E. and Pearl, J. Causal inference by surrogate experiments: z-identifiability. *Uncertainty in Artificial Intelligence*, 2012.

Bareinboim, E. and Pearl, J. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113:7345–7352, 2016.

Bowden, R. J. and Turkington, D. A. *Instrumental Variables*. Number no. 8 in Econometric Society Monographs in Quantitative Economics. Cambridge University Press, Cambridge [Cambridgeshire] ; New York, 1984. ISBN 978-0-521-26241-5.

Brito, C. and Pearl, J. Generalized instrumental variables. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pp. 85–93. Morgan Kaufmann Publishers Inc., 2002.

Chen, B. Identification and Overidentification of Linear Structural Equation Models. *Advances in Neural Information Processing Systems 29*, pp. 1579–1587, 2016.

Chen, B., Pearl, J., and Bareinboim, E. Incorporating Knowledge into Structural Equation Models Using Auxiliary Variables. *IJCAI 2016, Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 7, 2016.

Chen, B., Kumor, D., and Bareinboim, E. Identification and model testing in linear structural equation models using auxiliary variables. *International Conference on Machine Learning*, pp. 757–766, 2017.

Cinelli, C. and Hazlett, C. Making sense of sensitivity: extending omitted variable bias. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82 (1):39–67, 2020.

Cinelli, C., Kumor, D., Chen, B., Pearl, J., and Bareinboim, E. Sensitivity analysis of linear structural causal models. *International Conference on Machine Learning*, 2019.

Drton, M. and Weihs, L. Generic Identifiability of Linear Structural Equation Models by Ancestor Decomposition. *arXiv:1504.02992 [stat]*, April 2015.

Fisher, F. M. *The Identification Problem in Econometrics*. McGraw-Hill, 1966.

Foygel, R., Draisma, J., and Drton, M. Half-trek criterion for generic identifiability of linear structural equation models. *The Annals of Statistics*, pp. 1682–1713, 2012.

García-Puente, L. D., Spielvogel, S., and Sullivant, S. Identifying causal effects with computer algebra. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 193–200, 2010.

Gauss, C. Theoria motus, corporum coelesium, lib. 2, sec. iii, perthes u. *Besser Publ*, pp. 205–224, 1809.

Gessel, I. M. and Viennot, X. Determinants, paths, and plane partitions. *preprint*, 1989.

Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Kumor, D., Chen, B., and Bareinboim, E. Efficient identification in linear structural causal models with instrumental cutsets. In *Advances in Neural Information Processing Systems*, pp. 12477–12486, 2019.

Lee, S., Correa, J. D., and Bareinboim, E. General identifiability with arbitrary surrogate experiments. In *Proceedings of the Thirty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence*, Corvallis, OR, 2019. AUAI Press.

Legendre, A. M. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

Picard, J.-C. and Queyranne, M. On the structure of all minimum cuts in a network and applications. *Mathematical Programming*, 22(1):121–121, December 1982. ISSN 1436-4646. doi: 10.1007/BF01581031.

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

Stigler, S. M. *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1986.

Sullivant, S., Talaska, K., and Draisma, J. Trek separation for Gaussian graphical models. *The Annals of Statistics*, pp. 1665–1685, 2010.

Tian, J. Identifying linear causal effects. In *AAAI*, pp. 104–111, 2004.

Tian, J. Identifying direct causal effects in linear models. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, pp. 346. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Tian, J. A Criterion for Parameter Identification in Structural Equation Models. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 8, Vancouver, BC, Canada, 2007.

Van der Zander, B. and Liskiewicz, M. On Searching for Generalized Instrumental Variables. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS-16)*, 2016.

Van der Zander, B., Textor, J., and Liskiewicz, M. Efficiently Finding Conditional Instruments for Causal Inference. *IJCAI 2015, Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015.

Weihs, L., Robinson, B., Dufresne, E., Kenkel, J., McGee II, K. K. R., Reginald, M. I., Nguyen, N., Robeva, E., and Drton, M. Determinantal generalizations of instrumental variables. *Journal of Causal Inference*, 6(1), 2018.

Wright, P. G. *Tariff on Animal and Vegetable Oils*. Macmillan Company, New York, 1928.

Wright, S. Correlation and causation. *J. agric. Res.*, 20: 557–580, 1921.

A. Appendix

This appendix consists of 3 sections. The first section, A.1, presents the proofs for the theorems of Section 3 in the paper. The second section, A.2, contains the proof for the total-effect decomposition, as well as a method for easily solving for the covariance matrix of a mixed component using only the covariance matrix of the original model. Finally, the last Section, A.3, shows that generalizing ancestral decomposition as defined by Drton & Weihs (2015) is NP-Hard, making decomposition-based methods sub-optimal as a general strategy for identification.

A.1. Auxiliary Cutsets

In this section, we prove the theorems of Section 3. For simplicity of exposition, hereafter, unless otherwise specified (such as in Theorem 3.1), for any variable v, we use $v^\dagger$ to denote the variable

$$v^\dagger = v - \sum_{c_i \in C} \delta_{c_i v.C} c_i$$

where C is the (possibly empty) *Auxiliary Cutset* (Definition 3.2) for v constructed by the ACID algorithm. In this sense, whenever certain properties of $v^\dagger$ are stated, this means that, when the ACID algorithm constructs the AV $v^\dagger$ for the variable v, then $v^\dagger$ constructed in this way must have those properties.

Theorem 3.1. *Given a variable x, and a subset $\mathbf{C}$ of the ancestors of x, the covariance of the auxiliary variable $x^\dagger = x - \sum_i \delta_{c_i x.C} \times c_i$ with variable v can be determined by the sum of paths from x to v' in G_{flow} with source edges $\lambda_{c_i d_j}$ removed where $c_i \in \mathbf{C}$ and $d_j \in An(x)$.*

Proof. Let $c_1, ..., c_n$ be the set C ordered in topological order over G_{flow}. By definition of topological order, any paths from x to c_i cannot pass through any node c_j where $j > i$. Defining $C_i = \{c_1, ..., c_i\}$, we therefore have $\delta_{c_i x.C} = \delta_{c_x.C_{i-1}}$ (remember that paths in the source nodes of G_{flow} go in the *reverse* direction of arrows in the original graph - so δ_{cx} in the original graph is reflected by source-paths from x to c in the flow graph).

Define $x_i^\dagger = x - \sum_{j=1}^{i} \delta_{c_i x.C_{i-1}} c_i$ and G_{flow}^i as the flow graph encoding the covariances of $x_i^\dagger$. We will proceed by induction on i.

In the base case, $i = 0$, and $x_0^\dagger = x$, making $G_{flow}^0 = G_{flow}$. This holds by the definition of the flow graph.

Now, suppose that the covariances of $x_i^\dagger$ are encoded by G_{flow}^i, which is G_{flow} with source edges $\lambda_{c_j d_i}$ removed for $j < i$. We now observe $x_i^\dagger = x_{i-1}^\dagger - \delta_{c_i x.C_{i-1}} c_i$. The variable's covariance with v is therefore

$$\sigma_{x_i^\dagger v} = \sigma_{x_{i-1}^\dagger v} - \delta_{c_i x.C_{i-1}} \sigma_{c_i v}$$

Here, $\sigma_{x_{i-1}^\dagger v}$ consists of all paths between x and v' in G_{flow}^{i-1} by the inductive hypothesis. Likewise, $\delta_{c_i x.C_{i-1}}$ consists of all paths from x to c_i in G_{flow}^{i-1}, since all paths from x to c_i crossing elements of C_{i-1} have had their edges to C_{i-1} removed. Likewise, $\sigma_{c_i v}$ can be computed in G_{flow}, however, by the topological ordering, none of C_{i-1} are reachable from c_i in G_{flow}, meaning that the paths from c_i to v in G_{flow}^{i-1} also encode this covariance (none of the edges in the descendants of c_i were removed).

Finally, we observe that $\delta_{c_i x.C_{i-1}} \sigma_{c_i v}$ can therefore be seen as all the paths from x to v' that cross through c_i in G_{flow}^{i-1}. The covariance $\sigma_{x_i^\dagger v}$, therefore, consists of all paths in G_{flow}^{i-1} that *don't cross* c_i. This can be achieved by defining G_{flow}^i by removing the source edges incoming to c_i in G_{flow}^{i-1}.

We have therefore shown by induction that G_{flow}^n of $x^\dagger = x_n^\dagger$ encodes the covariance as described in the theorem. □

Theorem 3.2. *If Z is a PEIS relative to C and x, then the partial effects of C on x are identified, and given by $\delta_{c_i x.C} = \frac{\det \Sigma_{Z,C_{-i}^x}}{\det \Sigma_{Z,C}}$, where C_{-i}^x is the ordered set C with c_i replaced by x.*

Proof. Please note that this theorem refers to Z as variables, which are not themselves ACs. A simple modification to make this proof work for recursive application of ACs is given in Corollary A.1.

We will start by looking at Σ_{Z,C_{-i}^x}, specifically the column that was replaced with x. The jth row of this column is $\sigma_{z_j x}$. We will decompose this covariance into treks, and split it into paths crossing the elements of C. In particular:

$$\sigma_{z_j x} = \sum \text{treks from } z_j \text{ to } x'$$

We now group the treks for each c_k in C, such that the k'th trek set contains all the treks from z_j to x' where c_k is the last element of C crossed by the trek. We can do this, because by the definition of PEIS, all paths from z to x are intersected by C' (the source nodes of C in G_{flow}).

$$\sigma_{z_j x} = \sum_{c_k \in C} \text{sum treks from } z_j \text{ to } x' \text{ crossing } c_k' \textit{ last} \text{ of } \forall c' \in C'$$

We can focus the group of treks for c_k. We can decompose this into the paths *to* c_i, and the paths *from* c_i:

$$(\text{treks from } z_j \text{ to } c_k') \times (\text{paths from } c_k' \text{ to } x' \text{ avoiding all } c \in C)$$

The last term was because c_k was the last element of C on the treks from z_j to x. This corresponds to:

$$\sigma_{z_j x} = \sum_{c_k \in C} \sigma_{z_j c_k} \delta_{c_k x.C} \tag{5}$$

Finally, plugging this into the determinant $\det \Sigma_{Z,C^x_{-i}}$

$$\det \begin{bmatrix} \sigma_{z_1c_1} & \cdots & \sigma_{z_1c_{i-1}} & \sigma_{z_1x} & \sigma_{z_1c_{i+1}} & \cdots & \sigma_{z_1c_n} \\ \sigma_{z_2c_1} & \cdots & \sigma_{z_1c_{i-1}} & \sigma_{z_2x} & \sigma_{z_2c_{i+1}} & \cdots & \sigma_{z_2c_n} \\ \vdots & \cdots & & \vdots & & \cdots & \vdots \\ \sigma_{z_nc_1} & \cdots & \sigma_{z_nc_{i-1}} & \sigma_{z_nx} & \sigma_{z_nc_{i+1}} & \cdots & \sigma_{z_nc_n} \end{bmatrix}$$

$$= \det \begin{bmatrix} \sigma_{z_1c_1} & \cdots & \sum_{c_k\in C} \sigma_{z_1c_k}\delta_{c_kx.C} & \cdots & \sigma_{z_1c_n} \\ \sigma_{z_2c_1} & \cdots & \sum_{c_k\in C} \sigma_{z_2c_k}\delta_{c_kx.C} & \cdots & \sigma_{z_2c_n} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \sigma_{z_nc_1} & \cdots & \sum_{c_k\in C} \sigma_{z_nc_k}\delta_{c_kx.C} & \cdots & \sigma_{z_nc_n} \end{bmatrix}$$

Using the multilinearity of the determinant, this is equivalent to:

$$= \sum_{c_k\in C} \delta_{c_kx.C} \det \begin{bmatrix} \sigma_{z_1c_1} & \cdots & \sigma_{z_1c_k} & \cdots & \sigma_{z_1c_n} \\ \sigma_{z_2c_1} & \cdots & \sigma_{z_2c_k} & \cdots & \sigma_{z_2c_n} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \sigma_{z_nc_1} & \cdots & \sigma_{z_nc_k} & \cdots & \sigma_{z_nc_n} \end{bmatrix}$$

All terms C_k where $k \neq i$ have columns already in the matrix, meaning that these determinants are 0. This means:

$$= \delta_{c_ix.C} \det \begin{bmatrix} \sigma_{z_1c_1} & \cdots & \sigma_{z_1c_i} & \cdots & \sigma_{z_1c_n} \\ \sigma_{z_2c_1} & \cdots & \sigma_{z_2c_i} & \cdots & \sigma_{z_2c_n} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \sigma_{z_nc_1} & \cdots & \sigma_{z_nc_i} & \cdots & \sigma_{z_nc_n} \end{bmatrix}$$

Finally, we divide by the given determinant to get the desired result:

$$\det \Sigma_{Z,C^x_{-i}} = \delta_{c_ix.C} \det \Sigma_{Z,C}$$
$$\frac{\det \Sigma_{Z,C^x_{-i}}}{\det \Sigma_{Z,C}} = \delta_{c_ix.C}$$

Since Z and C have a full path-set to each-other, $\Sigma_{Z,C}$ is full-rank, so its determinant is generically non-zero, completing the proof. □

Corollary A.1. *If $Z^\dagger$ is a PEIS relative to C and x, then the partial effects of C on x are identified, and given by $\delta_{c_ix.C} = \frac{\det \Sigma_{Z^\dagger,C^x_{-i}}}{\det \Sigma_{Z^\dagger,C}}$, where C^x_{-i} is the ordered set C with c_i replaced by x.*

Proof. We will perform a tiny modification of the proof of Theorem 3.2. In particular, the decomposition given in Eq. (5) needs to be modified to reference paths in the flow graph for $z^\dagger_i \in Z$ as defined in Theorem 3.1:

$$\sigma_{z^\dagger_j x} = \sum_{c_k\in C} \sigma_{z^\dagger_j c_k} \delta_{c_kx.C}$$

Note that $\delta_{c_kx.C}$ is identical in all graphs of $Z^\dagger$, since no paths in the sink nodes are modified. Here, $\sigma_{z^\dagger_j c_k}$ is found in the flow graph of $z^\dagger_j$. However, notice that other than using covariances of $v^\dagger$, the proof of Theorem 3.1 works without modification. □

Theorem A.1. *Let C be the vertex-min-cut between a set Z and set P, closest to P in DAG G. If C' is the vertex min-cut between a set $Z' = Z \cup \{z'\}$ and P closest to P, then*

1. *$|C| \le |C'|$*
2. *if $|C| = |C'|$ then $C = C'$*
3. *all paths from C to P intersect C'*

Proof. (1) can be proved by simply noticing that if $|C'| < |C|$, since C' also cuts Z from P, then C' is a valid cut that is smaller than C - a contradiction of min-cut property of C.

(2) Suppose $C \neq C'$, then $\exists v \in C'$ such that $v \notin C$. There are two possibilities - either $v \notin De(Z)$, in which case $C' \setminus \{v\}$ is a min-cut for Z - a contradiction, since this cut is smaller than the min-cut, or $v \in De(Z)$, in which case C is already the min-cut of $De(Z)$ closest to P, meaning that C' cannot be the closest min-cut.

(3) If $|C| = |C'|$, $C = C'$, so this holds. If $|C| < |C'|$, we can construct a cut C^* for Z' to P by defining $C^* = C \cup \{z'\}$. The closest-min-cut C' must be either C^* or closer to P. This means that all paths from C^* must intersect the closest min-cut, and $C \subset C^*$, so all paths from C to P must intersect C'. □

Corollary A.2. *Given a set Z which has a full flow to set C, where C intersects all paths from Z to P in DAG G, then all paths from C to P intersect the min-cut C' between Z and P closest to P.*

Proof. By the fact that Z has a full flow to C, it means that any min-cut between Z and C must be of size $|C|$. If the min-cut C' closest to P is of size $|C|$, then all paths from any other min-cut to P must pass through C'. If C' is smaller than C, we know that it must be in descendants of C, since Z has a full flow to C. □

Lemma A.1. *Given a set of candidate instruments Z, and a target node x, then if the vertex min-cut C from Z to the sink nodes of $Pa(x)$ has an element $c \in C$ in the source nodes, then if a candidate instrument $z_i \in Z$ has a path to c in G_{flow}, it is not part of any PEIS.*

Proof. By contradiction. Suppose there exists a PEIS Z' relative to C' and x, where $z_i \in Z'$. By definition of C', the sink nodes of C' cut Z' from the sink nodes of $Pa(x)$. Using Corollary A.2, and the fact that sink nodes do not have directed paths to any source nodes in G_{flow}, we know that there is a closest-min-cut C^* between Z and $Pa(x)$, where C^* is made up of sink nodes, and all paths from C' cross C^*. Now, using Theorem A.1, by adding all other candidate nodes Z to Z', we get the closest vertex-min-cut C between Z and $Pa(x)$, and we know that all paths from C^* are intersected by $C_s \subseteq C$. This means that all paths from z_i in C are cut using only sink nodes C_s - meaning that the element c is redundant, as it does not block any previously unblocked paths. This is a contradiction to the min-cut property of C. □

Theorem 3.3. *If there exists a feasible ancestral cutset for x in G, then the AC exists, and is found by Algorithm 1.*

Proof. Using Corollary A.3, we know that we can find the closest min-cut from all candidate instruments Z, some of which can have ACs by using the half-flow graph G_{hf}.

In the first steps of the algorithm, we exploit Lemma A.1 to eliminate all z_i from candidacy which cannot be part of any PEIS. After running the loop, we can construct the min-vertex-cut C closest to $Pa(x)$ from the remaining candidates $Z' \subseteq Z$, where all nodes in the cut are sink nodes. At this point, any PEIS must have as its instrumental set $Z^* \subseteq Z$, and feasible ancestral cutset C^*, which by Corollary A.2 and Theorem A.1 (3) must have all its paths to $Pa(x)$ intersect C. Next, by running a max-flow between Z' and C (by definition of vertex min-cut, the flow will be of size $|C|$), we get a set Z'' of elements of Z with flow 1 through them, which is a valid PEIS for C, making C the Auxiliary Cutset. □

Theorem 3.4. *Let A be the unremoved ancestors of the AV $x^\dagger$. If $x^\dagger$ is a candidate instrument for y, then all elements of the AV $a^\dagger$ of $a \in A$ are also candidate instruments for y.*

Proof. If $a \in A$ has no path to $\{y\} \cup Sib(y)$, then it is already a candidate instrument by definition. The only case that requires consideration is when $\{y\} \cup Sib(y)$ is an ancestor of a in G. Since a is an unremoved ancestor of x, it means that there is a path from x to a in G_{flow} with edges incoming into the auxiliary cutset C removed. This means that the cutset of $x^\dagger$ must include a subset $C' = C \cap An(a)$ that cuts a from $\{y\} \cup Sib(y)$ (otherwise, since there is a path from a to x, the path from $\{y\} \cup Sib(y)$ to a combined with the path from a to x makes a path from $\{y\} \cup Sib(y)$ to x, which would mean that $x^\dagger$ is not a candidate instrument - a contradiction).

Our goal here is to show that C' is a feasible ancestral cutset for a, meaning that using Corollary A.2 and Theorem A.1, $a^\dagger$ will also have all paths to $\{y\} \cup Sib(y)$ subtracted out. Let Z' be the instruments for C' in the original PEIS that was created for $x^\dagger$. We show that Z' are also candidate instruments for a, and that C' cuts them from a, which will complete the proof.

Suppose that $z \in Z'$ is not a candidate instrument for a. This means that z has $\{a\} \cup Sib(a)$ in its unremoved ancestors. This is equivalent to saying that C' does not cut all paths from z to a, since the path from z, to $\{a\} \cup Sib(a)$ and then to a exists. But any such path can be extended to x, since C does not block all paths from x to a, since a is an unremoved ancestor. But C was defined as a full cutset for the instruments - a contradiction.

Now, since all Z' are candidate instruments, and C' is a valid cutset, we know that the AC created for a will cut away all paths through C', including those through $\{y\} \cup Sib(y)$. □

Corollary A.3. *Given all candidate instruments Z for x, a vertex min-cut between Z and the sink nodes of $Pa(x)$ in G_{hf} is the same as the vertex min-cut between Z and the sink nodes of $Pa(x)$ in G_{flow} where each candidate has its subtracted paths removed.*

Proof. Using Theorem 3.4, we know that all source nodes reachable by $z^\dagger \in Z$ are also candidate instruments. This means that the closest min-cut must cut *all* edges to sink nodes from each of these variables in both G_{hf} and G_{flow}. That is, suppose that $z^\dagger$ has a path to $Pa(x)$ which goes up in the source nodes to z', crosses down to sink nodes, and continues to $Pa(x)$. Since z' must be a candidate instrument too, the path must be blocked *at or after* z' - blocking it in the source nodes before z' would not block the path from z' itself. However, the path from z' exists in G_{hf}, so all such source paths will be blocked in G_{hf} in the same way as they would be blocked in G_{flow} - meaning that the two graphs have identical closest min-cuts for Z. □

For the next theorem, we need to recall the definition of a *match-block* given in Kumor et al. (2019).

Definition A.1. *(Kumor et al., 2019) Given a directed acyclic graph $G = (V, D)$, a set of source nodes S and sink nodes T, the sets $S_f \subseteq S$ and $T_f \subseteq T$, with $|S_f| = |T_f| = k$, are called* ***match-blocked*** *iff for each $s_i \in S_f$, all elements of T reachable from s_i are in the set T_f, and the max flow between S_f and T_f is k in G where each vertex has capacity 1.*

Theorem A.2. *Any edges returned by the IC algorithm when run over G_{hf}, when given a set $Z^\dagger$ of candidate instruments are identifiable. That is, the modified IC algorithm (Algorithm 4) is sound.*

Proof. First, we use the fact that the closest min-cut as used by the IC is found with the half-flow graph (Corollary A.3).

Algorithm 4 - IC: The IC algorithm from Kumor et al. (2019), modified to accept a set of candidate instruments Z. See Theorem A.2 for a proof of the modifications' correctness.

```
Input: $G, y, \Lambda^*, Z$
$G_{hf} \leftarrow$ AuxHalfFlowGraph($G$,$y$,$\Lambda^*$)
$T \leftarrow$ all sink-node parents of $y'$ in $G_{hf}$
$C \leftarrow$ ClosestMinVertexCut($G_{hf}, Z, T$)
$S_f \leftarrow$ elements of $S$ that have a full flow to $C$
$G_c \leftarrow G_{hf}$ with edges to $C$ removed
$(C_m, T_m) \leftarrow$ MaxMatchBlock($G_c, C, T$)
$T_f \leftarrow$ elements of $T$ that are part of a full flow between $C \setminus C_m$ and $T \setminus T_m$
$(S_f, T_f \cup T_m, T_m)$
```

This means that all paths from $v^\dagger$ in each $v^\dagger$'s flow graph must cross C. There also exists a flow of size $|C|$ from the candidate instruments Z to C (by definition of min-cut). We can therefore always decompose the paths for a candidate instrument $v_i^\dagger$ into paths that cross the cutset. Our proof will therefore be a near-verbatim repeat of Theorem 3.2. The main difference between the two is that the IC allows elements of the cutset to be in the source nodes, meaning that we can no longer easily interpret parts of paths as partial effects. We will also show that the $v^\dagger$ each having different paths in the source nodes does not affect our conclusions.

Decompose the paths from $v_i^\dagger$ to a parent node x_j of y into paths crossing each element $c_k \in C$ of the cutset. Like in the proof of Theorem 3.2, the decomposition is such that a path from $v_i^\dagger$ to x_j is assigned to c_k if c_k is the *last* element of C along the path.

$$\sigma_{v_i^\dagger x_i} = \sum_{c_k \in C} p_{v_i c_k | v_i^\dagger} p^{-C}_{c_k x_j | v_i^\dagger}$$

In the above equation, we used the notation p^{-C} to represent all paths that do not cross any elements of C, and the subscript $|v_i^\dagger$ to represent the paths of the flow graph for $v_i^\dagger$.

We now make the claim that the paths which don't intersect C are identical for all elements of $V^\dagger$. That is,

$$p^{-C}_{c_k x | v_i^\dagger} = p^{-C}_{c_k x | v_j^\dagger} = p^{-C}_{c_k x}$$

This means that we can drop the subscript $|v_i^\dagger$. This claim is trivial when c_k is in the sink nodes, as it is in the PEIS, since the flow graphs for all $V^\dagger$ are identical in the sink nodes.

If c_k is in the source nodes, however, we no longer have this guarantee. Nevertheless, we can exploit the results of Theorem 3.4 to claim that any path that crosses from c_k in the source nodes to any other source node, means that the other source node is also a candidate instrument, and C intersects all paths from it - meaning that c_k is not the last element of C that intersects the path. This is a contradiction to the definition of p^{-C}, so we can conclude that all paths from p^-C must cross directly into the sink nodes, meaning that they are identical for all elements of $V^\dagger$.

We therefore have:

$$\sigma_{v_i^\dagger x_i} = \sum_{c_k \in C} p_{v_i c_k | v_i^\dagger} p^{-C}_{c_k x_j}$$

Now, suppose that the IC given is between sets V and T, with t_n being x, and usable to identify edge λ_{xy}. We observe the determinant $\Sigma_{V^\dagger, T^y_{-i}}$:

$$\Sigma_{V^\dagger, T^y_{-i}} = \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ \sigma_{v_1^\dagger y} & \cdots & \sigma_{v_n^\dagger y} \end{bmatrix}$$

$$= \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ \sum_{c_k \in C} p_{v_1 c_k | v_i^\dagger} p^{-C}_{c_k y} & \cdots & \sum_{c_k \in C} p_{v_n c_k | v_i^\dagger} p^{-C}_{c_k y} \end{bmatrix}$$

By the multilinearity of determinant, we can move the sum outside the matrix:

$$= \sum_{c_k \in C} \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ p_{v_1 c_k | v_i^\dagger} p^{-C}_{c_k y} & \cdots & p_{v_n c_k | v_i^\dagger} p^{-C}_{c_k y} \end{bmatrix}$$

Now, we can decompose $p^{-C}_{c_k x}$ into a sum over the T, in the same way as was done above:

$$p^{-C}_{c_k x} = \sum_{t_i \in T} p^{-C}_{c_k t_i} p^{-C-T}_{t_i y}$$

Note that all the T are in the sink nodes (The T are a subset of the sink nodes of $Pa(y)$ as defined by IC algorithm), so we can use the notation of total effects:

$$p^{-C}_{c_k x} = \sum_{t_i \in T} p^{-C}_{c_k t_i} \delta_{t_i y, CT}$$

Replacing the above in the determinant, and once again exploiting the multilinearity property, we get:

$$= \sum_{c_k \in C} \sum_{t_i \in T} \delta_{t_i y, CT} \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ p_{v_1 c_k | v_i^\dagger} p^{-C}_{c_k t_i} & \cdots & p_{v_n c_k | v_i^\dagger} p^{-C}_{c_k t_i} \end{bmatrix}$$

$$= \sum_{t_i \in T} \delta_{t_i y, CT} \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ \sigma_{v_1^\dagger t_i} & \cdots & \sigma_{v_n^\dagger t_i} \end{bmatrix}$$

Just like in Theorem 3.2, all $i \neq n$ makes the determinant have a repeated row, and be equal to 0, giving:

$$= \delta_{t_n y, CT} \det \begin{bmatrix} \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_1} \\ \sigma_{v_1^\dagger t_1} & \cdots & \sigma_{v_n^\dagger t_2} \\ \vdots & \cdots & \vdots \\ \sigma_{v_1^\dagger t_{n-1}} & \cdots & \sigma_{v_n^\dagger t_{n-1}} \\ \sigma_{v_1^\dagger t_n} & \cdots & \sigma_{v_n^\dagger t_n} \end{bmatrix}$$

This gives us the equation:

$$\det \Sigma_{V^\dagger, T^y_{-i}} = \delta_{t_n y, CT} \det \Sigma_{V^\dagger, T}$$

allowing us to solve: $\delta_{t_n y, CT} = \frac{\det \Sigma_{V^\dagger, T^y_{-i}}}{\det \Sigma_{V^\dagger, T}}$.

Finally, we realize that Kumor et al. (2019) proved that whenever there is a match-block for x, we have that $\delta_{t_n y, CT} = \lambda_{xy}$.

□

Theorem A.3. *ACID runs in Polynomial-time*

Proof. The ACID algorithm runs in a loop while "anything changed". We show that this loop is executed at most V^2 times (V is the number of vertices), where each iteration is polynomial. First, we use the result from Theorem A.1, to claim that the auxiliary cutset of a vertex can only increase in size over iterations. This is because if a new cutset is the same size as a previous one, it is identical, meaning that there is "no change". It it is different, it means that it is strictly larger. A cutset cannot get larger than the number of vertices in the graph, so each node will have a new AC (i.e. a change) at most V times. At worst, only one AV will change in each iteration, meaning that the loop will need to run V times for each V variables.

Each iteration of the loop is polynomial, since it consists only of max-flows, set operations, and an invocation of the IC algorithm (which is polynomial). □

Theorem A.4. *Let C_v be the c-component of v, and $Tb(C_v)$ be the nodes on the top boundary of mixed component of C_v. Then $v^\dagger$'s paths don't cross the source nodes of $Tb(C_v)$ (i.e. there is an AC for v that cuts away $Tb(C_v)$).*

Proof. We will prove this by induction on the nodes of G in *any* topological order, including the one where G is just a graph of the ancestors of v - allowing us to make C_v the ancestral c-component.

Let V_i be the first i nodes in a topological order over G. In the base case, with V_1, there is a single node v_1, where $C_{v_1} = \{v_1\}$, which does not have a top boundary, meaning that the theorem holds.

Suppose that the theorem holds for all vertices V_{n-1}. Observe the nodes in $Tb(C_{v_n})$. All of these nodes are in different c-components than C_{v_n}. Each $t \in Tb(C_{v_n})$ has a corresponding $t^\dagger$, for which Theorem A.4 holds by the inductive hypothesis. This means that all paths from $t^\dagger$ to v_n must cross the bottom boundary of C_t, and therefore must cross through the sink nodes of $Tb(C_{v_n})$ in G_{flow}. This means that the set of sink nodes $Tb(C_{v_n})'$ is a feasible ancestral cutset for PEIS $Tb(C_{v_n})$ to v_n. By Corollary A.2 and Theorem A.1, the AC for $v_n^\dagger$ cuts all the paths to $Tb(C_{v_n})$. □

Corollary A.4. *If there exists a conditioning set W that d-separates y from v in a graph where all edges $x_i \to y$ are removed, then $v^\dagger$ is a candidate instrument for y.*

Proof. It is sufficient to prove that any $\{y\} \cup Sib(y)$ that are ancestors of v are in a different c-component from v, which allows us to invoke Theorem A.4 to prove that $v^\dagger$ has all ancestral paths to $\{y\} \cup Sib(y)$ cut, meaning that it is a candidate instrument.

Suppose v is in the same c-component as $s \in \{y\} \cup Sib(y)$, and s is an ancestor of v. We note that Theorem A.4 was proven for any topological ordering of nodes, so we choose the ordering such that v is in the same *ancestral* c-component. This means that $Sib(s) \subset An(v)$ in the graph of v's ancestors. Since s is an ancestor of v, there must be at least one conditioning node blocking the backdoor paths from v to s, meaning that s can be used as a collider, so all $Sib(s)$ must also have their paths blocked by conditioning. Continuing this recursively, we need to condition on the entire ancestral c-component, which will create an unblocked path from s to y using only bidirected edges. This shows that s must be in a different c-component, completing the proof. □

Definition A.2. *Let the **v-subset** of conditioning set W from instrument z be the subset $W_v \subseteq W$ such that each $w \in W_v$ has an unblocked path to z starting from an edge incident to w, and optionally crossing v-structures of W. That is, there exists a path $w \leftrightarrow a_1 \leftrightarrow ... \leftrightarrow a_k \leftrightarrow z$, where*

$a_1, ..., a_k \in W$, and $\leftrightarrow$ represents a trek that doesn't cross any element of W except at its endpoints.

Lemma A.2. *Given conditioning set W that d-separates y from z in a graph where all edges $x_i \to y$ are removed, each element of the v-subset of z is a candidate instrument.*

Proof. We show that each element v of the v-subset of z must have a conditioning set W_v that d-separates y from v in the graph with all directed edges incident to y are removed. Suppose not. That means that there does not exist a conditioning set blocking v from y. If a conditioning set exists, then a set exists that is made up of ancestors of v (y's incident edges were removed) (Van der Zander et al., 2015). Since such a set does not exist, there is a path into v's ancestors that leads to y, but by definition of v-subset, we can then create a path to y from z, a contradiction.

We then invoke Corollary A.4 to show that indeed, the element is a candidate instrument. □

Theorem A.5. *The Conditional Instrumental Variable (cIV) method is strictly subsumed by ACID.*

Proof. Suppose we have a cIV z conditioned on W for identifying edge $\lambda_{x_1 y}$, with an unblocked path p from z to x_1. Our proof consists of two parts: (i) we first show that the existence of such W, z and p allows the construction of auxiliary variables that can be used as candidate instruments. And, then (ii) we show how one can use those AVs to construct an instrumental cutset which can be used to identify $\lambda_{x_1 y}$ in the ACID algorithm.

To begin, we claim that all elements of $V^\dagger$, constructed from the v-subset V of z, are candidate instruments using Lemma A.2.

Next, we look at p. It consists of a series of treks, starting at z, and ending at x_1, with all intermediate trek endpoints on elements of W, and with no elements of W otherwise in p (i.e. an unblocked path can cross multiple v-structures at ws, but is not otherwise blocked). Decompose p into the series of treks $p = z \leftrightarrow w_1 \leftrightarrow w_2 ... \leftrightarrow x_1 = [t_1, t_2, ..., t_k]$ Let $HT(t_i)$ be the *last* element of $Left(t_i)$ along the trek (i.e. the last element in the source nodes when viewing the trek in a flow graph). Now, let $P_{ht} = \{HT(t_1), HT(t_2), ..., HT(t_k)\}$. We claim that set of variables $P_{ht}^\dagger$ are also candidate instruments. Note that all $HT(t_i)$ are part of a active backdoor path from either z or a node w that is d-connected to z. That means that a subset of W must block each $HT(t_i)$ from $Sib(y) \cup \{y\}$ in the graph of $HT(t_i)$'s ancestors (otherwise, there would be an unblocked path between z and y, a contradiction to the requirements of a cIV). We can therefore use Corollary A.4 to claim that all elements of $P_{ht}^\dagger$ are candidate instruments.

Finally, we use Theorem 3.4 to claim that all unremoved ancestors $A^\dagger$ of $P_{ht}^\dagger \cup V^\dagger$ are also candidate instruments.

We have therefore showed that at a certain iteration of the ACID algorithm, the set $Z = P_{ht}^\dagger \cup V^\dagger \cup A^\dagger$ are all candidate instruments. We will next show that there exists a cutset C where $|C| \leq |Z|$, there is a flow of size $|C|$ from Z to C, and C cuts Z from the sink nodes of $Pa(y)$. In this cutset, there will be a match-block for x_1, meaning that the closest min-cut as found by the IC algorithm will also have a match-block to x_1, showing that $\lambda_{x_1 y}$ can be identified (Kumor et al., 2019).

We construct the cut C between source nodes of Z and the sink nodes of $Pa(y)$ by including the following nodes in C:

- All the sink nodes of V
- All the source nodes of $A \setminus (V \cup P_{ht})$
- The sink node of x_1

With the set C constructed as above, we now prove that it cuts Z from the sink nodes of $Pa(y)$. We will proceed by showing this for each component of $Z = P_{ht}^\dagger \cup V^\dagger \cup A^\dagger$ individually.

Let $a \in A^\dagger \setminus (V^\dagger \cup P_{ht}^\dagger)$. The source node is part of the cut, which automatically blocks all paths from $a^\dagger$.

Let $v \in V^\dagger$. All paths from v to x_1' are cut by x_1', so we focus on the paths from v to the sink nodes of other parents of y. Furthermore, v' is part of the cut, so all directed paths from v to $Pa(y)$ are cut by v'. This leaves us with paths from v to $Pa(y)$ starting either with a bidirected edge to a sibling of v, or an edge into a parent of v that was an unremoved ancestor of $v^\dagger$.

Suppose that the path starts from a bidirected edge to a sibling of v. Each such path to $Pa(y) \setminus \{x_1\}$ must be intersected by an element of V, because v was part of the v-subset, so if a path from v to a sibling of v, and down to $Pa(y) \setminus \{x_1\}$ was not intersected by W, it means that there is an active path to another parent of y from z, violating the cIV condition. Each element of $v_i \in V$ has the sink node $v_i' \in C$, so all such paths are blocked by C.

Next, suppose that the path from v starts with an edge into an unremoved parent of v. All such parents are elements of A, so if the parent is an element of $A \setminus (V \cup P_{ht})$, the path is blocked by the source node of the parent, which is part of the cutset.

Finally, this leaves us with the paths from v starting with an edge into an unremoved parent which is an element of $V \cup P_{ht}$, at which point the arguments outlined above can be repeated recursively if the parent is an element of V, and the arguments claiming that elements of P_{ht} are cut by C (below) can be used otherwise.

This leaves us with the claim that each $h \in P_{ht}$ is also cut by C from the sink nodes of the parents of y. We know that all paths using outgoing edges or bidirected edges to siblings of h must intersect elements of V' or x_1', meaning that they are cut (otherwise we would have an unblocked path from z to y, violating the cIV assumption). Just like with v, this leaves paths into the unremoved parents of h, which are either elements of $A \setminus (V \cup P_{ht})$, in which case the paths are blocked directly, or $V \cup P_{ht}$, where the argument can be repeated recursively for the parent.

This completes the proof that all elements of the candidate instruments Z are cut by the cutset C. We now show that there is a flow of size $|C|$ from Z to C. We will match the variables as follows:

1. each element of $A \setminus (V \cup P_{ht})$ with itself
2. The elements of $v \in V$ that are endpoints of a trek in p are *not* matched - instead, their sink nodes that are part of the cutset, are matched with the element of P_{ht} that is part of the trek ending at the sink element of v,
3. The element x_1' is matched with the element of P_{ht} whose trek ends at x_1'.
4. All other $v \in V$ are matched to their own sink nodes.

None of these matched paths intersect, giving a flow of size $|C|$. Finally, by definition of instrumental variable, x_1' has no unblocked (by C) directed path to $Pa(y) \setminus \{x_1\}$, since otherwise z would have a path to that variable also. This means that x_1' is match-blocked with x_1', showing that $\lambda_{x_1 y}$ is identifiable with IC.

Finally, the graph in Fig. 2c is not identifiable with cIV, but can be identified by ACID, showing strict subsumption. □

Corollary A.5. *The Conditional Auxiliary Variable (cAV) is strictly subsumed by ACID.*

Proof. Consider the first iteration of the algorithm. As shown by the previous theorem (Theorem A.5), we know that ACID identifies at least as many coefficients as cIV. In the next iteration, both the cIV and ACID will construct AVs with the previously identified coefficients, which will be used as candidate instruments. Again, ACID identifies at least as many coefficients as cIV. Since this holds for every step of the algorithm, any instrument enabled for cAV is also enabled for ACID.

Finally, Fig. 2c can also be used here to show strict subsumption. □

Theorem A.6. *The Generalized Instrumental Set (gIS) is strictly subsumed by ACID*

Proof. We will follow the same procedure as was done for Theorem A.5. It is recommended that readers first understand the proof of that theorem, since this proof is an extension of the arguments made for cIV.

Suppose we have a generalized instrumental set $(z_1, W_1, p_1), ..., (z_k, W_k, p_k)$. Chen et al. (2017) showed that the paths of a gIS can be reduced to half-treks, so each p_i is a half-trek from z_i to x_i, meaning the trek takes its first edge from a source node to the sink nodes in the flow graph (not taking any backdoor paths).

To begin, we construct a set of candidate instruments for y similar to the one constructed in the proof of Theorem A.5. First, we use Corollary A.4 to claim that each $z_i^\dagger$ is a candidate instrument, Lemma A.2 to claim that all elements of the v-subset V_i of z_i are also candidates, and Theorem 3.4 to claim that the ACs of all unremoved ancestors A_i of $\{z_i^\dagger\} \cup V_i^\dagger$ are candidates.

Next, we define P as the set of nodes along the paths $p_1, ..., p_k$ excluding the paths' start/endpoints. Define $V = (\bigcup_i V_i) \setminus (P \cup \{z_1, ..., z_k\})$, and $A = (\bigcup_i A_i) \setminus (V \cup \{z_1, ..., z_k\})$

We now use the set $Z = \{z_1^\dagger, ..., z_k^\dagger\} \cup V^\dagger \cup A^\dagger$ as the set of candidate instruments, and show that the min-cut from Z to $Pa(y)$ closest to y' contains the elements $x_1', ..., x_k'$, which are match-blocked, allowing to identify the edges with IC (Kumor et al., 2019).

Define the cut C as follows:

1. Sink nodes $x_1', ..., x_k'$ are part of the cut
2. Sink nodes of all elements of V are part of the cut,
3. Source nodes of all elements of A are part of the cut.

We have a full flow by matching each z_i to x_i', each V to its own sink node, and each A to itself. By the definitions of A and V, none of these paths intersect.

Next, we show that C intersects all paths from Z. We make the same arguments here as were made in the proof of Theorem A.5, with the major caveat that V has elements of P removed. Therefore, we know that any unblocked path must pass through an element of V_i that was removed as part of P. However, suppose that the element e is part of p_j, and was part of V_i. Any paths from the instrument's V_i or z_i that pass through the sink node e' are no longer blocked by e'. However, all such paths are blocked by elements of V_j, since p_j is not blocked for the instrument z_j. If any such elements of V_j are on yet another path, we can continue the same argument recursively.

Finally, by definition of gIS, x_i' has no unblocked (by C) directed path to $Pa(y) \setminus \{x_1, ..., x_k\}$, since otherwise z

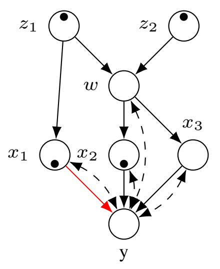


Figure 8. Taken from Kumor et al. (2019), λ_{x_1y} is not identifiable with qAVS, but can be identified with IC (and therefore ACID)

would have a path to that variable also. Since x'_i has a path to x'_i (they are the same node), the set $x'_1, ..., x'_k$ are match-blocked, showing that $\lambda_{x_1y}, ..., \lambda_{x_ky}$ is identifiable with IC. □

Corollary A.6. *The qAVS is strictly subsumed by ACID*

Proof. The qAVS is identical to the gIS, with the addition of auxiliary variables to the instruments and to y. The IC algorithm and AC algorithms both explicitly include both types of AVs, meaning that any instrument usable for qAVS is also enabled for ACID.

Finally, Fig. 8 can be used here to show strict subsumption. □

Theorem A.7. *The Instrumental Cutset is strictly subsumed by ACID*

Proof. The IC algorithm is run as a sub-component of ACID, with the only difference being additional candidate instruments. This means that any candidates present for IC, are also available for ACID, making the same edges identifiable.

Strict subsumption can be seen with Fig. 2f, which is not captured by IC (Kumor et al., 2019), but can be computed with ACID. □

Theorem 3.5. *If λ_{ab} is identifiable with either the cAV, IC, or qAVS criteria, then it is identifiable with ACID.*

Proof. See Corollary A.5, Corollary A.6, and Theorem A.7.

An example parameter that is only identifiable with ACID, but cannot be identified with *any* of the previous algorithms is given in Fig. 9. To see why this is the case, we first show how ACID identifies the edge,and then show that IC, cAV, and qAVS all fail.

When ACID starts, it uses w_2 to identify $\lambda_{w_2z_3}$, creating $z_3^\dagger$, and uses w_2 as a PEIS for w_2, giving $z_2^\dagger = z_2 - \delta_{w_2z_2}w$. Then, ACID uses these three AVs to construct the IC with cutset x'_1, w'_1, x'_4, identifying λ_{x_1y} and λ_{x_4y}.

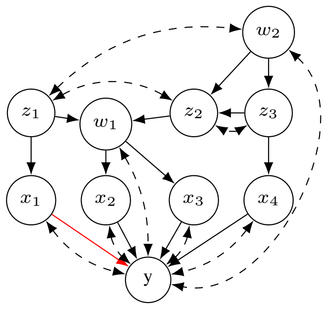


Figure 9. λ_{x_1y} is only identifiable with ACID

The cAV works in a similar way, identifying $\lambda_{w_2z_3}, \lambda_{z_3x_4}, \lambda_{z_2w_1}, \lambda_{z_1w_1}, \lambda_{z_1x_1}, \lambda_{w_1z_2}, \lambda_{w_1z_3}$. None of the AVs resulting from these identified edges can be used to identify any additional parameters, since w_1 cannot be conditioned.

Likewise, qAVS identify all the same parameters as cAV here, but no conditioning exists that is able to cut an instrument for x_1 from x_2 and x_3, for which no instrumental set exists.

Finally, the IC fails here, since it cannot construct $z_2^\dagger$, leaving only 2 candidate instruments, z_1 and z_3^* - for which there is no match-block to any x.

Also note that the entire graph is a single c-component, so no form of c-component decomposition can be used here to aid in identification. □

A.2. Total-Effect Decomposition

Theorem 4.1. *Let $G_{An(y)}$ be the graph G with the non ancestors of y removed. Let C_y be the c-component of y in $G_{An(y)}$ and G' its corresponding mixed component. Then, if $x \in C_y$, the total effect of x on y, δ_{xy}, can be decomposed as, $\delta_{xy} = \delta'_{xy} + \sum_{b \in Tb(C_y)} \delta_{xb}\delta'_{by}$ otherwise, if $x \notin C_y$, we have, $\delta_{xy} = \sum_{b \in Tb(C_y)} \delta_{xb}\delta'_{by}$ where $Tb(C_y)$ is the top boundary of the c-component C_y and δ'_{by} is the total effect of node $b \in Tb(C_y)$ on y in the mixed component G'. Moreover, all δ'_{by} for $b \in Tb(C_y)$ are identified.*

Proof. Given δ_{xy}, take the mixed component of y, G', which has top boundary B. We know by the definition of paths in the graph that δ_{xy} is the sum of all possible directed paths starting from x, and ending at y.

There are two cases:

1. x is not in y's c-component. All paths in δ_{xy} cross elements of B at least once, since they must cross into y's c-component. Decompose the paths of δ_{xy} such that δ_{xy}^b is the sum of all paths where b is the LAST

element of B on the path. We then have:

$$\delta_{xy} = \sum_{b \in B} \delta_{xy}^b$$

Next, we split each path in δ_{xy}^b into the portion from x to b, and the portion from b to y.

Looking at the portion from b to y, we know that none of the paths cross any boundary nodes (because b was the last boundary node along the path), and that these are ALL such paths. This corresponds to the sum of paths from b to y in the mixed component of y.

Finally, observing the portion from x to b, there are no restrictions on these paths - they are all paths in G that start at x, and end in b, possibly crossing other nodes in B before getting to b. This corresponds to δ_{xb}. Therefore, we have:

$$\delta_{xy} = \sum_{b \in B} \delta_{by}^{G'} \delta_{xb}$$

2. x is in y's c-component, first take out the δ'_{xy} as all paths that do not cross any boundary. Then split the remaining paths as was done in part 1. We get:

$$\delta_{xy} = \delta_{xy}^{G'} + \sum_{b \in B} \delta_{by}^{G'} \delta_{xb}$$

□

A.2.1. Solving for the Decomposed Covariance Matrix

In this section, we develop a decomposition algorithm which is equivalent to the decomposition described by Tian (2005). The main difference of our method is that it operates directly upon the covariance matrix, and has an intuitive interpretation as selecting paths in the flow graph. That is, we define a system of linear equations whose solution is the covariance matrix of the desired mixed component. This covariance matrix can then be used to compute the terms required for the total-effect decomposition described in the text.

We begin by defining the "boundary" of a mixed component as the variables that are also present in other mixed components:

Definition A.3. *Given graph G, with c-components $C_1, ..., C_k$, the **top boundary** of C_i is $Tb(C_i) = Pa(C_i) \setminus C_i$, and the **bottom boundary** is $Bb(C_i) = Pa(\bigcup_{j \neq i} C_j) \cap C_i$. Their union is called the **boundary set**.*

In Fig. 2f, $\{z_1, z_2\}$ is the top boundary, and $\{w\}$ is the bottom boundary for the c-component $\{w, x_1, x_2, y\}$. The boundary corresponds to the nodes at which treks need to be modified to create the decomposed graph (Fig. 2g). Our algorithm hinges on keeping track of the "missing paths" between the top boundary and all other boundary nodes in the decomposed graph, which are kept in the **boundary matrix**. Specifically, given two boundary nodes, v, w, where w is on the top boundary, B_{vw} is the sum of treks that start at v, take a directed or bidirected edge that is not in G', and end at w' (without passing v' in the flow graph if v is a top node).

With the boundary matrix, and the covariances of the mixed component, we can create a system of equations reconstructing the original model's covariance matrix:

Definition A.4. *Given the following:*

1. *G' is a mixed component of G corresponding to c-component C,*
2. *Σ is the (known) covariance matrix of G,*
3. *$v_1, ..., v_n$ is a topological ordering of nodes in G', according to the graph of G*
4. *$Tb_i = Tb(C) \cap \{v_1, ..., v_i\}$ and $Bb_i = Bb(C) \cap \{v_1, ..., v_i\}$ according to the topological ordering.*
5. *B is the boundary matrix of C, Σ' is the mixed component's covariance matrix. Both are unknown.*

*Then the **decomposition system** is a system of equations defined over Σ', B: for each v_i, with $h = 1...i$,*

1. *if $v_i \in Tb_i$ and $h = i$, let $\sigma_{ii} = \sigma'_{ii}$.*
2. *if $v_i \in Tb_i$, $h \neq i$ and $v_h \in Tb_i \cup Bb_i$, define:*

$$\sigma_{hi} = \sum_{t \in Tb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{ti} + \sum_{b \in Bb_{i-1}} \sigma'_{hb} B_{bi}$$

   Furthermore, let $\sigma'_{hi} = 0$, and let all $B_{ji} = 0$ where $j \in Bb(C) \setminus Bb_i$.

3. *otherwise, define*

$$\begin{aligned}\sigma_{hi} = \sigma'_{hi} &+ \sum_{t \in Tb_{i-1}} \sum_{u \in Tb_{i-1} \setminus \{t\}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tu} \frac{\sigma'_{ui}}{\sigma_{uu}} \\ &+ \sum_{t \in Tb_{i-1}} \sum_{b \in Bb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tb} \sigma'_{bi} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{t \in Tb_{i-1}} \sigma'_{hb} B_{bt} \frac{\sigma'_{ti}}{\sigma_{tt}}\end{aligned}$$

We will demonstrate the intuition behind the decomposition system by manually decomposing the model in Fig. 10. In particular, given Σ, we want to find the values of Σ' such that they are consistent with the model in Figs. 10b and 10d.

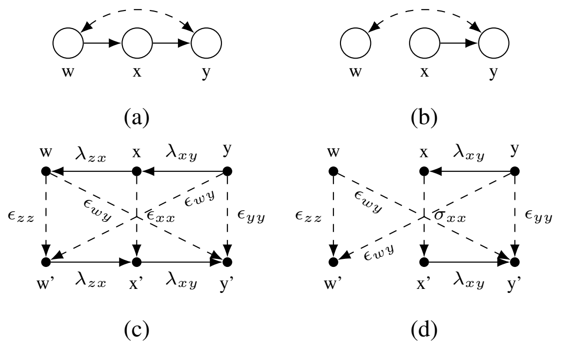


Figure 10. (a) shows the original graph, with the flow graph in (c). The decomposed model is in (b), with its flow graph in (d).

Going in topological order, we first have w, which is not on the top boundary (it is on the bottom boundary), and with no other boundary nodes thus far, condition 3 gives $\sigma_{ww} = \sigma'_{ww}$, which is identical in both Fig. 10c and Fig. 10d. Next in topological order comes x, which is on the top boundary. Here, condition 1 gives $\sigma_{xx} = \sigma'_{xx}$, and since w was a bottom boundary, condition 2 gives $\sigma'_{wx} = 0$ and $\sigma_{wx} = \sigma'_{ww} B_{wx}$. This results in $B_{wx} = \lambda_{wx}$.

Finally, we are left with y. Since it is not on the top boundary, all of its equations are constructed using condition 3:

$$\sigma_{wy} = \sigma'_{wy} + \frac{\sigma'_{wx}}{\sigma_{xx}} B_{wx}\sigma'_{wy} + \sigma'_{ww} B_{wx}\frac{\sigma'_{xy}}{\sigma_{xx}}$$
$$\sigma_{xy} = \sigma'_{xy} + \frac{\sigma'_{xx}}{\sigma_{xx}} B_{wx}\sigma'_{wy} + \sigma'_{xw} B_{wx}\frac{\sigma'_{xy}}{\sigma_{xx}}$$
$$\sigma_{yy} = \sigma'_{yy} + \frac{\sigma'_{yx}}{\sigma_{xx}} B_{wx}\sigma'_{wy} + \sigma'_{yw} B_{wx}\frac{\sigma'_{xy}}{\sigma_{xx}}$$

Focusing on the first two equations, we can simplify them using values that were previously solved:

$$\sigma_{wy} = \sigma'_{wy} + \epsilon_{ww}\lambda_{wx}\frac{\sigma'_{xy}}{\sigma_{xx}}$$
$$\sigma_{xy} = \lambda_{wx}\sigma'_{wy} + \sigma'_{xy}$$

Notice that this is a system of two equations with 2 unknowns. In both equations, the σ' add in all treks that make up G', additionally including treks that make up σ_{xx}. The boundary terms complete the system by adding paths that move across the boundaries, in particular, in σ_{wy}, the trek $\epsilon_{ww}\lambda_{wx}\lambda_{xy}$, and in σ_{xy}, the trek $\lambda_{wx}\epsilon_{wy}$. Finally, having solved this system, the equation for σ_{yy} has only a single unknown variable, σ'_{yy}, giving the desired mixed component's covariance matrix.

This procedure can always be performed, and leads to a unique covariance matrix Σ' for the mixed component:

Theorem A.8. *Given a decomposition system, the equations can be iteratively solved for all B and Σ' as a series of linear systems of equations. Each of these systems is full-rank.*

Proof. By generating the equations in topological order, we can define consecutive full rank systems of equations. Suppose that all systems up to node $i-1$ were solved for B and σ'. We have two cases in the generator, one when i is a top boundary, and h is a boundary, and one "otherwise". We show that both can be turned into full rank systems.

In the case where i is a top node and h is a boundary node, it suffices to look at the system of equations for *all* boundary nodes h with i:

$$\begin{bmatrix} \sigma_{t_1 i} \\ \vdots \\ \sigma_{t_n i} \\ \sigma_{b_1 i} \\ \vdots \\ \sigma_{b_m i} \end{bmatrix} = \begin{bmatrix} \frac{\sigma'_{t_1 t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{t_1 t_n}}{\sigma_{t_n t_n}} & \sigma'_{t_1 b_1} & \cdots & \sigma'_{t_1 b_m} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\sigma'_{t_n t_1}}{\sigma_{t_n t_1}} & \cdots & \frac{\sigma'_{t_n t_n}}{\sigma_{t_n t_n}} & \sigma'_{t_n b_1} & \cdots & \sigma'_{t_1 b_m} \\ \frac{\sigma'_{b_1 t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{b_1 t_n}}{\sigma_{t_n t_n}} & \sigma'_{b_1 b_1} & \cdots & \sigma'_{b_1 b_m} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\sigma'_{b_m t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{b_m t_n}}{\sigma_{t_n t_n}} & \sigma'_{b_m b_1} & \cdots & \sigma'_{b_m b_m} \end{bmatrix} \begin{bmatrix} B_{t_1 i} \\ \vdots \\ B_{t_n i} \\ B_{b_1 i} \\ \vdots \\ B_{b_m i} \end{bmatrix}$$

To show that the system is full rank, it suffices to show that the determinant of the center matrix is generically non-zero:

$$\det \begin{bmatrix} \frac{\sigma'_{t_1 t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{t_1 t_n}}{\sigma_{t_n t_n}} & \sigma'_{t_1 b_1} & \cdots & \sigma'_{t_1 b_m} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\sigma'_{t_n t_1}}{\sigma_{t_n t_1}} & \cdots & \frac{\sigma'_{t_n t_n}}{\sigma_{t_n t_n}} & \sigma'_{t_n b_1} & \cdots & \sigma'_{t_1 b_m} \\ \frac{\sigma'_{b_1 t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{b_1 t_n}}{\sigma_{t_n t_n}} & \sigma'_{b_1 b_1} & \cdots & \sigma'_{b_1 b_m} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\sigma'_{b_m t_1}}{\sigma_{t_1 t_1}} & \cdots & \frac{\sigma'_{b_m t_n}}{\sigma_{t_n t_n}} & \sigma'_{b_m b_1} & \cdots & \sigma'_{b_m b_m} \end{bmatrix}$$

Using the fact that multiplying a column by a constant is equivalent to multiplying the entire determinant by the same constant, we can extract all factors of $\frac{1}{\sigma_{t_i t_i}}$, giving us:

$$= \prod_{i=1}^{n} \frac{1}{\sigma_{t_i t_i}} \det \Sigma'_{Tb_{i-1}\cup Bb_{i-1}, Tb_{i-1}\cup Bb_{i-1}}$$

However, here Σ' corresponds to the covaraince of variables with themselves, which by Sullivant et al. (2010) is generically non-zero, because we have a path in the flow graph from each variable's top node to its own bottom node. Therefore the system is full rank, and can be *uniquely* solved for the B_{hi}.

Next, we show that we can always solve the second case. Here, we have a system where i is any non-top node, and h is any node. Observe the subsystem of equations where

h is the set of boundary nodes. All terms except σ'_{bi} where b is a boundary node were previously solved. This once again describes a system of linear equations. We will show that this system has a non-zero determinant by showing that its value is $1 + ...$, where all other terms contain at least one structural parameter (our models are not assumed to be normalized, so treks won't generically add up to 1).

To show this, we will observe that the matrix has ones on its diagonal, and any non-diagonal term contains structural parameters. Notice that σ'_{hi} has a factor of 1. We show that none of the summations contain σ'_{hi}, this making the matrix diagonal 1:

$$\begin{aligned}\sigma_{hi} = \sigma'_{hi} &+ \sum_{t \in Tb_{i-1}} \sum_{u \in Tb_{i-1} \setminus \{t\}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tu} \frac{\sigma'_{ui}}{\sigma_{uu}} \\ &+ \sum_{t \in Tb_{i-1}} \sum_{b \in Bb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tb} \sigma'_{bi} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{t \in Tb_{i-1}} \sigma'_{hb} B_{bt} \frac{\sigma'_{ti}}{\sigma_{tt}}\end{aligned}$$

Suppose h is in the top boundary. Then, to match σ'_{hi}, in the first summation, u must be h:

$$\sum_{t \in Tb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{th} \frac{\sigma'_{hi}}{\sigma_{hh}}$$

However, σ'_{ht} is 0 between nodes in the top boundary, and $t \neq h$, since $u \neq t$ by the definition of the summation over u. This term is therefore 0.

Similarly, the only other way to get a σ'_{hi} term is in the last summation, which would be:

$$\sum_{b \in Bb_{i-1}} \sigma'_{hb} B_{bh} \frac{\sigma'_{hi}}{\sigma_{hh}}$$

σ'_{hb} can be non-zero only if b is later than h in topological order. However, B_{bh} is 0 for b later in topological order, making this term also 0.

Next, suppose that h is in the bottom boundary. The only term that can match σ'_{hi} in this case is the middle summation.

$$\sum_{t \in Tb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{th} \sigma'_{hi}$$

Once again, either σ'_{ht} or B_{th} is 0, since $\sigma'_{ht} \neq 0$ means t is an ancestor of h, but B_{th} is 0, since boundary matrix element between bottom and top boundary can only be nonzero when t is later in topological order.

We therefore know that the matrix has 1s on its diagonal. Next, we know that wherever there is a non-zero term in an off-diagonal element, it must contain a B_{ij}, which are guaranteed to contain structural parameters that are not present in σ' (so they cannot cancel when dividing σ_{tt}).

This is sufficient to prove that the above defined system of equations is full rank, and has a unique solution for all σ'_{hi} where h is on the boundary. Finally, we show that we can exploit the solutions for boundary nodes to solve the system where h is not on the boundary:

$$\begin{aligned}\sigma_{hi} = \sigma'_{hi} &+ \sum_{t \in Tb_{i-1}} \sum_{u \in Tb_{i-1} \setminus \{t\}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tu} \frac{\sigma'_{ui}}{\sigma_{uu}} \\ &+ \sum_{t \in Tb_{i-1}} \sum_{b \in Bb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tb} \sigma'_{bi} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{t \in Tb_{i-1}} \sigma'_{hb} B_{bt} \frac{\sigma'_{ti}}{\sigma_{tt}}\end{aligned}$$

Here, notice that only σ'_{hi} has not yet been solved - all other terms have been solved either in the system of equations with boundary nodes, or in previous systems. This means that we can solve for this single variable.

This completes the proof - we have demonstrated an order in which solving the system leads to full rank systems of linear equations, giving unique solutions to all of the unknowns. □

Theorem A.9. *The covariance matrix Σ' found using the decomposition system for c-component C corresponds to the covariance matrix of the mixed component of C defined such that each top boundary node t has variance σ_{tt}, and all other edges in G''s flow graph are identical to their matched values in G.*

Proof. To achieve this, we will show that the equations in Definition A.4 correspond directly to a sum over sets of treks, which, due to the uniqueness and linearity proved in Theorem A.8, must match the decomposition's values. We will go through the 3 cases individually.

In case 1, $\sigma_{tt} = \sigma'_{tt}$ by definition. This allows the t node to include all treks starting from t, going into its ancestors, and continuing back down to t (and into descendants).

In case 2, i is a top boundary node (which has no parents in G'), and h all come before i in topological order, so there is no trek between h and i in G', making $\sigma'_{hi} = 0$. Likewise, $B_{ij} = 0$ for all $j \in Bb$ later than i in topological order, because all paths from j not in G' must pass down into j's descendants using directed edges, making none of these treks able to return to i.

Finally, since i is a top node, all treks from h to i will cross through the boundary, meaning that:

$$\sigma_{hi} = \sum_b \begin{matrix}\text{treks } t \in T \text{ from } h \text{ to } i \text{ with } b \text{ as first} \\ \text{boundary node where } b' \notin t \text{ if } b \in Tb_{i-1}\end{matrix}$$

We exclude treks that cross top boundary b, and then cross b' (the bottom node of b in the flow graph), since such treks are already accounted for when we defined σ_{bb}. Since such treks move back to G', we instead use the *next* boundary node they cross as b in the above summation.

In the above equation, if b is a top boundary, the portion of the trek from h to b can be written as $\frac{\sigma'_{hb}}{\sigma_{bb}}$, since σ_{bb} is the b node's variance, and needs to be removed. The rest of the trek to i crosses the boundary, making it part of the boundary matrix element B_{bi}.

Similarly, if b is a bottom boundary, the treks from h to b can be writted as σ'_{hb}, and from b, the paths are from the boundary matrix. This gives the desired equation:

$$\sigma_{hi} = \sum_{t \in Tb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{ti} + \sum_{b \in Bb_{i-1}} \sigma'_{hb} B_{bi}$$

Finally, in case 3, we use the same reasoning, but simply need to add paths from the boundary nodes, to the target node, as well as the possibility of certain paths not crossing boundary nodes at all:

$$\begin{aligned}\sigma_{hi} = \sigma'_{hi} &+ \sum_{t \in Tb_{i-1}} \sum_{u \in Tb_{i-1} \setminus \{t\}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tu} \frac{\sigma'_{ui}}{\sigma_{uu}} \\ &+ \sum_{t \in Tb_{i-1}} \sum_{b \in Bb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tb} \sigma'_{bi} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{t \in Tb_{i-1}} \sigma'_{hb} B_{bt} \frac{\sigma'_{ti}}{\sigma_{tt}} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{u \in Bb_{i-1}} \sigma'_{hb} B_{bu} \sigma'_{ui}\end{aligned}$$

Finally, by observing that B_{bu} must be 0 when both b and u are boundary nodes, we get the desired equation:

$$\begin{aligned}\sigma_{hi} = \sigma'_{hi} &+ \sum_{t \in Tb_{i-1}} \sum_{u \in Tb_{i-1} \setminus \{t\}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tu} \frac{\sigma'_{ui}}{\sigma_{uu}} \\ &+ \sum_{t \in Tb_{i-1}} \sum_{b \in Bb_{i-1}} \frac{\sigma'_{ht}}{\sigma_{tt}} B_{tb} \sigma'_{bi} \\ &+ \sum_{b \in Bb_{i-1}} \sum_{t \in Tb_{i-1}} \sigma'_{hb} B_{bt} \frac{\sigma'_{ti}}{\sigma_{tt}}\end{aligned}$$

We have therefore shown that the decomposition given in Definition A.4 is a valid decomposition according to Σ',

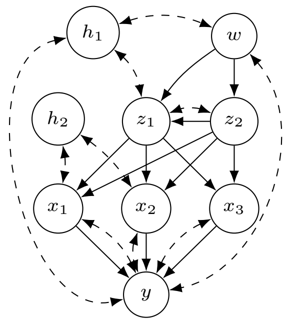


Figure 11. In order to identify $\lambda_{x_1 y}$ with an IS combined with Tian's decomposition, one would need to marginalize **only** h_1, and leave h_2.

and since we showed that the solutions to the unknowns are unique in Theorem A.8, we can conclude that the covariance matrix Σ' solved using these systems must match the mixed component's graph with identical structural parameters.

□

A.3. Generalizing Ancestral Decomposition

One question that might come up is whether one can simply combine existing efficient methods with a variant of ancestral decomposition (Drton & Weihs, 2015) to gain an efficient identification algorithm. While this algorithm would still miss examples such as the one in Fig. 4b, we go further, by showing that efficiently extending Ancestral Decomposition to work in a generalized context is NP-Hard.

To demonstrate the source of complexity, refer to Fig. 11, which is an extension of Fig. 2f. Notice that h_1 and h_2 are non-ancestors of y. The ancestral decomposition algorithm of Drton & Weihs (2015) would first attempt to identify $\lambda_{x_1 y}, \lambda_{x_2 y}, \lambda_{x_3 y}$ in the original graph, which is a single c-component. It would then marginalize all non-ancestors of y, giving the graph where h_1 and h_2 are removed. In this case, z_1 and z_2 are no longer in y's c-component, allowing a model decomposition. However, z_1, z_2 are insufficient to operate as an instrumental set for any edges incident to y - it is only when h_2 is also available that the IS h_2, z_1, z_2 can be used to identifiy $\lambda_{x_1 y}, \lambda_{x_2 y}, \lambda_{x_3 y}$.

This means that Ancestral decomposition as defined by Drton & Weihs (2015) is insufficient here. One would need to extend their algorithm to both marginalize over h_1, and leave h_2 to successfully identify the given edges. This section demonstrates that deciding which non-ancestors of y to marginalize to enable identification with the IS algorithm is NP-Hard.

Theorem A.10. *Given a boolean formula F in conjunctive normal form, if a graph G is constructed as follows, starting with a target node y:*

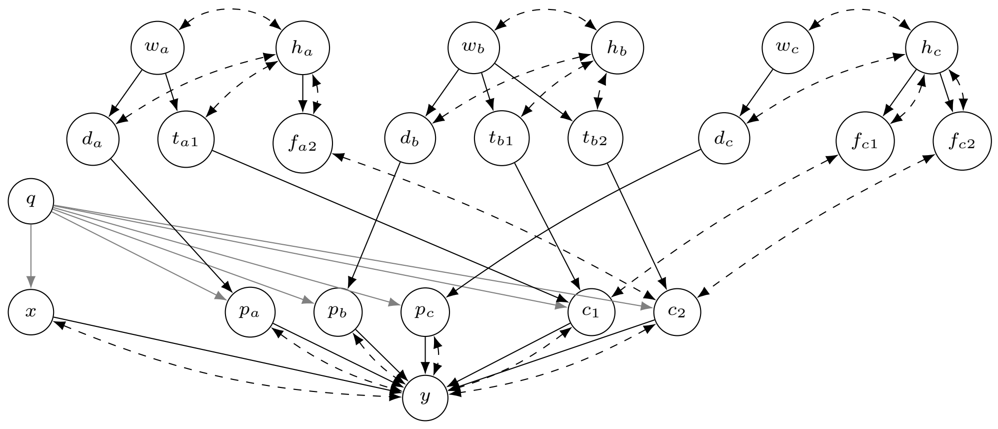


Figure 12. Encoding of $(a \vee b \vee \neg c) \wedge (\neg a \vee \neg c \vee b)$. Bidirected edges from w_a, w_b and w_c to y were left out for clarity.

1. *For each clause $c_i \in F$, a node c_i is added with edges $c_i \to y$ and $c_i \leftrightarrow y$*

2. *For each variable $v_i \in F$, create nodes w_i, h_i, d_i, p_i, with edges $w_i \leftrightarrow y$, $w_i \leftrightarrow h_i$, $h_i \leftrightarrow d_i$, $w_i \to d_i$, $d_i \to p_i$, $p_i \to y$, $p_i \leftrightarrow y$.*

3. *For each variable $v_i \in F$, if clause $c_j \in F$ contains:*

   - *v_i: Add node t_{ij}, with edges $w_i \to t_{ij}$, $t_{ij} \to c_j$, $t_{ij} \leftrightarrow h_i$*
   - *$\neg v_i$: Add node f_{ij}, with edges $h_i \to f_{ij}$, $h_i \leftrightarrow f_{ij}$, $c_j \leftrightarrow f_{ij}$*

4. *Add nodes x and q, with edges $x \to y$, $x \leftrightarrow y$, $q \to x$, and $q \to p_i$, $q \to c_j$, $\forall p_i, c_j$.*

Then there exists a set of nodes $\notin An(y)$ that can be marginalized such that the resulting mixed component for y has an instrumental set that can be used to solve for λ_{xy} if and only if F has a satisfying assignment.

An example graph for the formula $(a \vee b \vee \neg c) \wedge (\neg a \vee \neg c \vee b)$ is given in Fig. 12. In particular, looking at h_a, if both f_{a2} and h_a are marginalized, then d_a and t_{a1} are in a separate mixed component, cutting their back-paths through w_a to y, and allowing d_a to match with p_a, and t_{a1} to match with c_1 (i.e. the value being true (marginalized) allows satisfying clause c_1). If h_a is not marginalized, then h_a is matched to p_a, and f_{a2} can match to c_2, meaning that $\neg a$ satisfies clause c_2.

This general procedure works for all 3SATs

Proof. $\to$: If there is a satisfying assignment to the variables, then we can sum out h_k for all variables k which are true, and leave h_j and f_j for all variables j that are false in the assignment. We can then construct the instrumental set by choosing a variable that satisfies each clause - suppose c_i is satisfied by v. Then t_{vi} is a valid instrument for c_i, since it has no path to w in the mixed component. Similarly, f_{vi} is a valid instrument if the clause is satisfied by $\neg v$. Each of the p_i has either d_i or h_i as valid instruments, and q can be matched to x, which is a full IS for all the parents of y.

$\leftarrow$: Suppose there is no satisfying assignment to the variables. We can only marginalize over the f and h, since all other variables are ancestors of y. Marginalizing over a variable f does not break any c-components, and removes a candidate instrument, so it cannot help. Let $h_1, ..., h_k$ be a set of h that is marginalized. We know that the 3SAT has no satisfying assignment, so at least one clause c_j is not satisfied by setting all variables corresponding to the marginalized h to true, and all others to false. c_j has no possible instrument, since all t that are ancestors are in y's c-component, and therefore have path to y (and their h were not marginalized, because otherwise the clause would have been satisfied). Similarly, all f that have a path to c_j were marginalized, since otherwise they'd be instruments, and setting that variable to false would satisfy the clause. Therefore, there is no instrumental set here. □