Appendix: Differentiable Causal Discovery Under Unmeasured Confounding

The Appendix is organized as follows. In Appendix A we discuss details of the Greenery algorithm for penalizing c-trees and introduce the formalizations necessary to prove its correctness. In Appendix B we provide additional comments on the protein expression network learned by applying our method to the data from Sachs et al. (2005). In Appendix C we present formal proofs of results in our paper. In Appendix D we discuss additional implementation details and choice of hyperparameters for our experiments. Finally in Appendix E we provide additional experiments not included in the main draft of the paper.

A DETAILS OF THE GREENERY ALGORITHM

Bhattacharya et al. (2020) introduced a graphical and probabilistic operator called *primal fixing* that can be applied recursively to an ADMG and its statistical model to identify causal parameters of interest. In this section we provide the necessary background on the graphical operator and discuss how it relates to the detection of c-trees. We then show how primal fixing is codified in the steps of Algorithm 1 through an example.

A conditional ADMG (CADMG) $\mathcal{G} = (V, W, E)$ is an ADMG whose vertices can be partitioned into random vertices V and fixed vertices W, with the restriction that no arrowheads point into W (Richardson et al., 2017). A vertex V_i in a CADMG $\mathcal{G} = (V, W, E)$ is said to be primal fixable if there is no bidirected path from V_i to any of its direct children. The graphical operation of primal fixing V_i in $\mathcal{G}$, denoted by $\phi_{V_i}(\mathcal{G})$, yields a new CADMG $\mathcal{G} = (V \setminus V_i, W \cup V_i, E \setminus \{e \in E \mid e = \circ \rightarrow V_i \text{ or } \circ \leftrightarrow V_i\})$ where V_i is now "fixed" (denoted by a square box in figures shown in this Supplement) and incoming edges into V_i are deleted. This can be extended to a set of vertices as follows. A set of k vertices S is said to be primal fixable if there exists an ordering $(S_1, \ldots, S_k)$ such that S_1 is primal fixable in $\mathcal{G}$, S_2 is primal fixable in $\phi_{S_1}(\mathcal{G})$, S_3 is primal fixable in $\phi_{S_2}(\phi_{S_1}(\mathcal{G}))$, and so on. It is easy to see that any such valid ordering on S yields the same final CADMG. Hence, we can denote primal fixing a set of vertices S as simply $\phi_S(\mathcal{G})$. A vertex V_i in an ADMG $\mathcal{G}$ is said to be *reachable* if $V \setminus V_i$ is primal fixable in $\mathcal{G}$. Shpitser et al. (2018) showed that if V_i is reachable in $\mathcal{G}$, then the causal effect of the parents of V_i on V_i itself is identified, and there is no V_i rooted c-tree in $\mathcal{G}$.[1] If no valid primal fixing order exists, V_i along with the unique minimal set of vertices that could not be primal fixed form a V_i-rooted c-tree (Shpitser et al., 2018). That is, an ADMG $\mathcal{G}$ is arid if and only if every vertex $V_i \in V$ is reachable. This forms the basis of Algorithm 1.

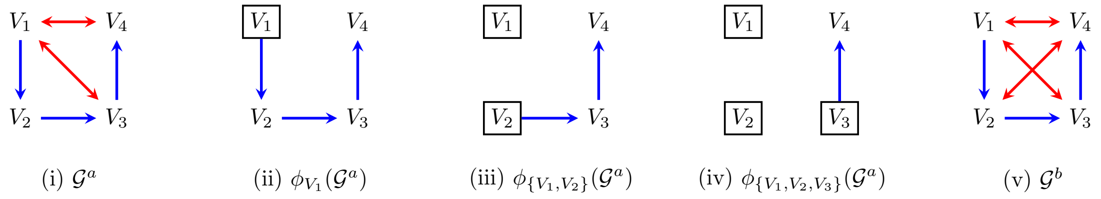
(i) $\mathcal{G}^a$
(ii) $\phi_{V_1}(\mathcal{G}^a)$
(iii) $\phi_{\{V_1,V_2\}}(\mathcal{G}^a)$
(iv) $\phi_{\{V_1,V_2,V_3\}}(\mathcal{G}^a)$
(v) $\mathcal{G}^b$

Figure A: (i) An arid ADMG; (ii) The CADMG obtained after primal fixing V_1; (iii) The CADMG obtained after primal fixing V_1 and V_2; (iv) The CADMG obtained after primal fixing V_1, V_2, and V_3; (v) A non-arid bow-free ADMG that is a super model of (i).

We now demonstrate usage of the primal fixing operator to establish that the ADMG $\mathcal{G}^a$ shown in Fig. A(i) is arid and the ADMG $\mathcal{G}^b$ shown in Fig. A(v) is not. These are the same graphs shown in Section 2 of the paper but we redraw and relabel them here for convenience. The reachability of vertices V_1, V_2, and V_3 in $\mathcal{G}^a$ is easily established. In every case, we can primal fix the remaining vertices in a reverse topological order starting with V_4 which has no children. The reachability of V_4 is established by noticing that V_1 is primal fixable in $\mathcal{G}^a$. In the resulting CADMG, shown in Fig. A(ii), both V_2 and V_3 are primal fixable. Primal fixing V_2 yields the CADMG

[1]Actually this was shown with respect to the ordinary fixing operator proposed in Richardson et al. (2017) which performs the same graphical operation as primal fixing but considers V_i to be fixable when there are no bidirected paths to any descendant (a vertex V_j such that there exists a directed path from V_i to V_j) of V_i. It is easy to see how primal fixing is a strict generalization of fixing by noting that the children of V_i is a subset of its descendants.

in Fig. A(iii) and finally primal fixing V_3 yields the CADMG in Fig. A(iv). Hence, all vertices in $\mathcal{G}^a$ are reachable. It then follows that $\mathcal{G}^a$ is arid. If we try to apply the same reasoning to the $\mathcal{G}^b$ in Fig. A(v), we see that V_1, V_2, and V_3 are still reachable as before. However, we cannot establish a sequence of primal fixing operations to reach V_4 as none of the other vertices are primal fixable in the original graph. Hence, there is a V_4-rooted c-tree in $\mathcal{G}^b$ comprised of the arborescence $V_1 \to V_2 \to V_3 \to V_4$ which also forms a bidirected component in $\mathcal{G}^b$.

A.1 Example Application of the Greenery Algorithm

We now demonstrate how the above primal fixing steps relate to Algorithm 1. Let the ordering of vertices of entries in the matrix be V_1, V_2, V_3, V_4. The adjacency matrices D and B for $\mathcal{G}^a$ in Fig. A(i) are as follows.

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The i^{th} iteration of the outer loop of the algorithm attempts to establish the reachability of V_i, and hence, the presence or absence of a V_i-rooted c-tree. Note that since the primal fixing operation can be applied at most $d-1$ times (where d is the number of vertices in $\mathcal{G}$) to determine the reachability of V_i, the inner loop of Algorithm 1 also executes $d-1$ times. We now focus on the final iteration of the algorithm where it tries to establish the reachability of V_4.

In the first iteration of the inner loop we have $D^f = D$ and $B^f = B$. Therefore we have,

$$e^{B^f} \circ D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.59 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 0 & 0 & 0.53 & 0.76 \end{bmatrix} \qquad F = \begin{bmatrix} 0 & 0 & 0.53 & 0.76 \\ 0 & 0 & 0.53 & 0.76 \\ 0 & 0 & 0.53 & 0.76 \\ 0 & 0 & 0.53 & 0.76 \end{bmatrix}.$$

Each entry i, j of the matrix $e^{B_f} \circ D$ is zero if and only if a bidirected path from V_i to V_j and a directed edge $V_i \to V_j$ do not co-exist in $\mathcal{G}$. The sum of the i^{th} row of this matrix then exactly characterizes the primal fixability criterion. That is, V_i is primal fixable if and only if the sum of the i^{th} row in $e^{B_f} \circ D$ is 0. The above calculations indicate that the vertices V_1, V_2, and V_4 are all primal fixable in $\mathcal{G}^a$, which can be easily confirmed by looking at the graph itself. The vector f then summarizes the primal fixability of each vertex except we add the i^{th} row of an identity matrix to ensure that we do not accidentally primal fix V_i itself when determining its reachability. The matrix F formed by tiling the f vector d times can then be used as a "mask" that implements the primal fixing operation applied to V_1 and V_2 simultaneously, yielding the following updates to D^f and B^f.

$$D^f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.53 & 0 \\ 0 & 0 & 0 & 0.76 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B^f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

It is easy to confirm that the induced ADMG $\mathcal{G}(D^f, B^f)$ corresponds to the CADMG shown in Fig. A(iii). Note that a constant positive scaling factor can also be applied to the hyperbolic tangent function to improve the sharpness of the approximation of the primal fixing operator. In the second iteration of the loop, we apply the same process again and obtain,

$$e^{B^f} \circ D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 0 & 0 & 0 & 0.76 \end{bmatrix} \qquad F = \begin{bmatrix} 0 & 0 & 0 & 0.76 \\ 0 & 0 & 0 & 0.76 \\ 0 & 0 & 0 & 0.76 \\ 0 & 0 & 0 & 0.76 \end{bmatrix}.$$

That is, in the second iteration of the algorithm, V_3 becomes primal fixable. Applying the primal fixing operator yields the adjacency matrices,

$$D^f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.58 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B^f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

which induce the CADMG shown in Fig. A(iv) corresponding to primal fixing V_3. Thus, in this case, reachability of V_4 is established in 2 steps. However, the algorithm will still perform a third step that does not result in any additional primal fixing and does not change the conclusion of reachability of V_4. As there are no vertices that have both a bidirected path and directed path to V_4 in the final CADMG and corresponding adjacency matrices, $C = e^{B^f} \circ e^{D^f}$ is simply the identity matrix. Taking the i^{th} column sum then evaluates to 1 which is subtracted off later in the final "return" step of the algorithm. A similar argument holds for vertices V_1, V_2, and V_3. Thus, applying Algorithm 1 to $\mathcal{G}^a$ in Fig. A(i) returns a value of 0 confirming that $\mathcal{G}^a$ is arid.

We now consider application of the algorithm to the ADMG $\mathcal{G}^b$ shown in Fig. A(v). We will apply a scaling constant of 10 to the hyperbolic tangent function, i.e., we use $\tanh(10x)$, so that the values are large enough to illustrate the main concept. We again focus on the reachability of V_4. The adjacency matrices for $\mathcal{G}^b$ are:

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

In the first iteration of the inner loop we have,

$$e^{B^f} \circ D = \begin{bmatrix} 0 & 0.64 & 0 & 0 \\ 0 & 0 & 0.19 & 0 \\ 0 & 0 & 0 & 0.64 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 1 & 0.96 & 1 & 1 \end{bmatrix} \qquad F = \begin{bmatrix} 1 & 0.96 & 1 & 1 \\ 1 & 0.96 & 1 & 1 \\ 1 & 0.96 & 1 & 1 \\ 1 & 0.96 & 1 & 1 \end{bmatrix}.$$

That is, we see that none of the vertices in $\mathcal{G}^b$ are primal fixable. Therefore applying the primal fixable operator through the matrix F results in adjacency matrices,

$$D^f = \begin{bmatrix} 0 & 0.96 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B^f = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0.96 \\ 1 & 0 & 0 & 0 \\ 1 & 0.96 & 0 & 0 \end{bmatrix},$$

which induce a "CADMG" that has the same edges as the original graph $\mathcal{G}^b$. Repeated applications of this in the second and third iterations do not change the structure of the induced graph. Therefore, upon termination of the inner loop, there remains a directed path from every vertex in $V \setminus V_4$ to V_4 and the vertices still form a bidirected connected component. That is, there is a V_4-rooted c-tree in $\mathcal{G}^b$. This is confirmed when we evaluate the sum of the i^{th} column of $C = e^{D^f} \circ e^{B^f}$ to 2.34. The other vertices V_1, V_2, and V_3 are still reachable and their respective column sums upon termination of the inner loop yield a value of 1 each. Subtracting d at the end of the algorithm still leaves a positive remainder of 1.34. Hence, Algorithm 1 returns a positive quantity when applied to $\mathcal{G}^b$, confirming that it is not arid.

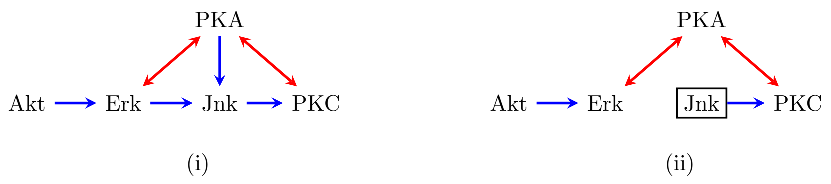

Figure B: (i) A subgraph of the protein network in Fig. 2 that we use to highlight the Verma constraint between Akt and PKC; (ii) A CADMG corresponding to the post-intervention distribution that would be obtained by intervening on Jnk.

B COMMENTS ON PROTEIN EXPRESSION ANALYSIS

In this section we discuss the Verma restriction that allows us to establish that Erk is *not* a cause of PKA. The importance of this relation stems from manipulation of Erk by the authors of Sachs et al. (2005) and establishing that no downstream change was observed in PKA.

We first point out that there is no ordinary conditional independence constraint between Akt and PKC in the learned structure shown in the right panel of Fig. 2, despite the absence of an edge between the two. This can be confirmed by noting the presence of an *inducing path* between Akt and PKC. An inducing path between V_i and V_j is a path from V_i to V_j where every non-endpoint is both a collider ($\rightarrow \circ \leftarrow, \leftrightarrow \circ \leftarrow$, or $\leftrightarrow \circ \leftrightarrow$) and has a directed path to either V_i or V_j. It is well-known that the presence of such a path precludes the possibility of an ordinary conditional independence of the form $V_i \perp\!\!\!\perp V_j \mid Z$ for any $Z \subseteq V \setminus \{V_i, V_j\}$ (Verma and Pearl, 1990). In our analysis it can be confirmed that Akt $\rightarrow$ Erk $\leftrightarrow$ PKA $\leftrightarrow$ PKC is an inducing path between Akt and PKC. Thus, there is no ordinary conditional independence between these two proteins under our learned model. However, under the faithfulness assumption, the absence of the edge between Akt and PKC implies an equality restriction. We now provide the non-parametric form of the corresponding Verma constraint.

Consider the ADMG and corresponding distribution obtained by recursively marginalizing out all vertices (except PKC) with no outgoing directed edges in Fig. 2. In performing this graphical operation, none of the variables removed act as a latent confounder for the remaining variables in the problem. Therefore, by rules of latent projection described in Verma and Pearl (1990), we simply obtain a subgraph of the original network as shown in Fig. B(i). Note that the inducing path between Akt and PKC is still preserved. Let $p(V^s)$ be the corresponding marginal distribution on the remaining subset of variables. The Verma constraint is then given by,

$$\text{Akt} \perp\!\!\!\perp \text{PKC in } \frac{p(V^s)}{p(\text{Jnk} \mid \text{Erk}, \text{PKA})}.$$

Intuitively, one can view the independence between Akt and PKC as manifesting in a post-intervention distribution obtained after intervening on Jnk, resulting in the CADMG (or truncated ADMG) shown in Fig. B(ii) where incoming edges to Jnk are removed. The resulting independence is then easily read off from the CADMG via the m-separation criterion (Richardson, 2003). See Tian and Pearl (2002) and Richardson et al. (2017) for more details on how to derive such constraints in general. Orienting the Erk $\leftrightarrow$ PKA edge as either Erk $\leftarrow$ PKA or Erk $\rightarrow$ PKA breaks the inducing path between Akt and PKC, meaning that either orientation produces a different independence model implying an ordinary independence constraint instead of the Verma restriction. We evaluated the BIC scores with either orientation and confirm that they both yield an increase in the score. This indicates that our learned model which posits that Erk is correlated with PKA through unmeasured confounding is the preferred causal explanation. This explanation is consistent with experiments performed in Sachs et al. (2005), and we are able to arrive at the same conclusion from purely observational data. Moreover, this explanation was differentiated from others via the Verma restriction between Akt and PKC, highlighting the value of considering general equality restrictions beyond ordinary conditional independence.

C PROOFS

Theorem 1 *The constraints shown in Table 1 are satisfied if and only if the adjacency matrices satisfy the relevant property of ancestrality, aridity, and bow-freeness respectively.*

Proof. We use the following facts for all of our proofs. The matrix exponential of a square matrix A is defined as the infinite Taylor series,

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k. \tag{1}$$

For a binary square matrix A, corresponding to a directed/bidirected adjacency matrix, the entry A^k_{ij} counts the number of directed/bidirected walks of length k from vertex i to vertex j; see for example (Butler, 2008).

Ancestral ADMGs

Consider the constraint shown in Table 1. That is,

$$\text{trace}(e^D) - d + \text{sum}(e^D \circ B) = 0.$$

It is easy to see from results in (Zheng et al., 2018) that the constraint $\text{trace}(e^D) - d = 0$ is satisfied if and only if the induced graph $\mathcal{G}(D, B)$ is acyclic. We now show that $\text{sum}(e^D \circ B) = 0$ if and only if $\mathcal{G}$ is ancestral.

By definition of the matrix exponential,

$$\begin{aligned} \text{sum}(e^D \circ B) &= \text{sum}\Big(I \circ B + \sum_{k=1}^{\infty} \frac{1}{k!} D^k \circ B\Big) \\ &= \text{sum}\Big(I \circ B\Big) + \sum_{k=1}^{\infty} \frac{1}{k!} \text{sum}\Big(D^k \circ B\Big), \end{aligned}$$

where the second equality follows from basic matrix properties.

The first term in the series, $\text{sum}(I \circ B)$, counts the number of self bidirected edges $V_i \leftrightarrow V_i$ which is a special-case violation of ancestrality. This term is zero if no such edges exist. An entry i, j in the matrix $D^k \circ B$ counts the number of occurences of directed paths from V_i to V_j of length k such that V_i and V_j are also connected via a bidirected edge. Therefore, all remaining terms of the form $\frac{1}{k!}\text{sum}(D^k \circ B)$ count the number of directed paths of length k that violate the ancestrality property rescaled by a positive factor of $\frac{1}{k!}$. That is, these terms are all ≥ 0 and equal to zero only when no such paths exist, i.e., $\mathcal{G}$ is ancestral.

Arid ADMGs

Consider the constraint shown in Table 1. That is,

$$\text{trace}(e^D) - d + \textsc{Greenery}(D, B) = 0.$$

The terms $\text{trace}(e^D) - d$ capture the acyclicity constraint as before. We now show that the output of Algorithm 1 is zero if and only if $\mathcal{G}$ satisfies the arid property. That is, $\textsc{Greenery}(D, B) = 0$ is satisfied if and only if $\mathcal{G}$ is arid. The background required for this proof was laid out in Appendix A.

The outer loop of Algorithm 1 iterates over each vertex V_i in order to evaluate its reachability, or equivalently, the presence/absence of a V_i-rooted c-tree (Shpitser et al., 2018). The inner loop achieves this as follows.

Reachability of V_i can be determined in at most $d-1$ primal fixing operations. Therefore, the inner loop executes $d - 1$ times. On each iteration, the algorithm considers the primal fixability of vertices by effectively treating the matrices D^f and B^f as adjacency matrices of a CADMG. In the first iteration, D^f and B^f are initialized with values from the directed and bidirected adjacency matrices respectively. The sum of the j^{th} row in the matrix $e^{B^f} \circ D^f$ evaluates to zero if and only if there are no bidirected paths from V_j to any of its direct children V_k, which exactly corresponds to the graphical criterion for determining primal fixability of V_j. The addition of the i^{th} row of an identity matrix to t ensures that V_i itself is not treated as primal fixable when evaluating its reachability. Therefore, in the first iteration, the vector f encodes a smoothened version (due to the application of the hyperbolic tangent function) of the usual primal fixability criterion for all vertices $V \setminus V_i$ in the original graph $\mathcal{G}$. Tiling the vector f to form the $d \times d$ matrix F allows us to apply the softened version of primal fixing to the adjacency matrices, which is performed in lines 6-8 of the algorithm. On the next iteration, the matrices D^f and B^f can then be treated as adjacency matrices of a CADMG obtained by primal fixing a set of vertices, say S_1, that satisfied the primal fixability criterion in $\mathcal{G}$. The same logic can be applied to subsequent iterations of the algorithm where we determine the primal fixability of a set of vertices $V \setminus (S_1 \cup V_i)$ in $\phi_{S_1}(\mathcal{G})$, denote the primal fixable vertices as S_2, and then proceed to do the same for $V \setminus (S_1 \cup S_2 \cup V_i)$ in $\phi_{S_1 \cup S_2}(\mathcal{G})$, and so on.

On termination of the inner loop, we have that $S_1 \cup S_2, \dots, \cup S_{d-1} \subseteq V \setminus V_i$. We first consider the case when equality holds. In this case, V_i is reachable, from which it follows that there is no V_i-rooted c-tree in $\mathcal{G}$ (Shpitser et al., 2018). The final matrices D^f and B^f then correspond to a CADMG where all vertices except V_i have been primal fixed. In such a CADMG the only edges that may be present are directed edges into V_i due to the removal of incoming edges to all other vertices in the graph. Thus, e^{B^f} evaluates to an identity matrix as there are no bidirected edges. Assuming $\mathcal{G}$ is a graph with no directed cycles (which is already enforced by the first two terms in the arid constraint), the Hadamard product $C = e^{B^f} \circ e^{D^f}$ is then also an identity matrix. Taking the sum of the i^{th} column of C then simply evaluates to 1. If every vertex $V_i \in V$ is reachable in this manner, it implies that the graph is arid, and the greenery quantity will then evaluate to d. The subtraction of d in the "return" statement of Algorithm 1 then returns a value of 0 for arid graphs. Now we consider the case when equality does not hold, i.e., there exists a set of vertices $X = V \setminus V_i \setminus (S_1 \cup S_2 \cdots \cup S_{d-1})$ that could not be primal fixed. This implies that V_i is not reachable and there exists a V_i-rooted c-tree. By definition, the structure of this c-tree comprises of directed and bidirected paths from vertices in X to V_i. The sum of the i^{th} column in $C = e^{B^f} \circ e^{D^f}$ then provides a weighted count of these paths. Subtracting off d in the final "return" statement then yields a positive quantity that provides a weight for each V_i-rooted c-tree detected in a non-arid graph $\mathcal{G}$.

Bow-free ADMGs

Consider the constraint shown in Table 1. That is,

$$\text{trace}(e^D) - d + \text{sum}(D \circ B) = 0.$$

The terms $\text{trace}(e^D) - d$ capture the acyclicity constraint as before. It is easy to see that the term $\text{sum}(D \circ B)$ counts the number of bows in the induced graph $\mathcal{G}$. Hence, $\text{sum}(D \circ B)$ is zero if and only if $\mathcal{G}$ is bow-free.

□

Theorem 2 *Let $p(V; \theta^*)$ be a distribution in the curved exponential family that is Markov and faithful with respect to an arid ADMG $\mathcal{G}^*$. Finding the global optimum of the continuous program in display (1) with $f \equiv BIC$ yields an ADMG $\mathcal{G}(\theta)$ that implies the same equality restrictions as $\mathcal{G}^*$.*

Proof. This follows immediately from the validity of the constraints in Theorem 1 and the consistency of the BIC score for model selection in curved exponential families (Haughton, 1988).

□

Corollary 1.2 *The results in Theorem 1 and Corollary 1.1 hold if every occurrence of a matrix exponential e^A is replaced with the matrix power $(I + cA)^d$ for any $c > 0$, where I is the identity matrix.*

Proof. The proof is straightforward by noting that the binomial expansion of $(I + cA)^d = I + \sum_{k=1}^{d} \binom{d}{k} c^k A^k$ which is similar to the infinite series expansion of the matrix exponential truncated to d terms. As paths greater than length d are irrelevant in a system with d vertices, these terms are sufficient.

□

Hyperparameter	Setting	Justification
Tolerance for $h(\theta)$	10^{-8}	Numerically close enough to 0 – the lower the better.
Max dual ascent iterations	100	Same value as in Zheng et al. (2018); convergence is typically achieved within 10 iterations.
RICF increment s	1	RICF often converges in 10 steps (Drton et al., 2009; Nowzohour et al., 2017). Higher values should be used for larger graphs.
Regularization strength λ	0.05	Obtained through manual testing on held-out data derived from Fig. 1(b,c).
Progress rate r	0.25	Same value as in Zheng et al. (2018); Yu et al. (2019).
Tolerance for RICF	10^{-4}	Numerically close enough to 0 – the lower the better.

Table A: Hyperparameter settings used for our experiments.

D IMPLEMENTATION DETAILS

In this section we discuss implementation details of our procedure that were not included in the main paper.

Implementation of Constraints

As mentioned in the main paper, we use the representation of constraints in Table 1 obtained by replacing each matrix exponential e^A with $(I + cA)^d$. We have two primary reasons for doing so. First, as pointed out by Yu et al. (2019), the latter representation is numerically more stable. Second, by evaluating the binomial expansion $(I + cA)^d = I + \sum_{k=1}^{d} \binom{d}{k} c^k A^k$ explicitly, we are able to obtain analytic gradients for our constraints automatically via the HIPS Autograd package (Maclaurin et al., 2015; Maclaurin, 2016). Analytic gradients for the matrix exponential on the other hand are not easily obtained and the function itself is not implemented in many popular computing libraries. In our implementation we use a value of $c = 1$ when computing portions of the constraint related to directed edges and a value of $c = 2$ when computing portions of the constraint related to bidirected edges. As the constraints in Theorem 1 are valid for any $c > 0$, these values were chosen only to make values of $h(\theta)$ under violations of ancestrality, aridity, and bow-freeness to be larger than the tolerance level (10^{-8}) of the augmented Lagrangian procedure. As mentioned in Section A, a scaling factor applied to the hyperbolic tangent function controls the sharpness of approximation of the primal fixing operator. In our experiments we use a scaling factor of $\ln(5000)$, but any sufficiently large value suffices as long as the penalty $h(\theta)$ computed for c-trees is above the tolerance level of the augmented Lagrangian procedure. Finally symmetry of the matrix β is enforced by requiring each off-diagonal entry β_{ij} and β_{ji} are tied to a single free parameter. Positive-definiteness of β is guaranteed by construction in the RICF procedure (Drton et al., 2009).

Choice of Hyperparameters

We summarize our choice of hyperparameters and justification for these choices in Table A. Choice of some hyperparameters, such as tolerance levels for RICF and increments in RICF iterations, require little justification as lower tolerance and more iterations can only improve approximation. We set specific values only to cap the run time of our procedure. Choices for most other hyperparameters are based on prior literature.

Converting Estimates of θ to an ADMG $\mathcal{G}(\theta)$

The final step of Algorithm 3 returns an ADMG $\mathcal{G}(\theta)$ as follows. We first derive the matrices δ and β from θ. The structure of the induced ADMG is then given by: $V_i \rightarrow V_j$ exists in $\mathcal{G}$ if $|\delta_{ij}| > \omega$ and $V_i \leftrightarrow V_j$ exists in $\mathcal{G}$ if $|\beta_{ij}| > \omega$ for all $i \neq j$. Such thresholding is standard in similar continuous optimization structure learning methods, such as Zheng et al. (2018) and Yu et al. (2019), and the threshold can be made arbitrarily small as long as tolerance to $h(\theta)$ is also small. In our experiments we use $\omega = 0.05$.

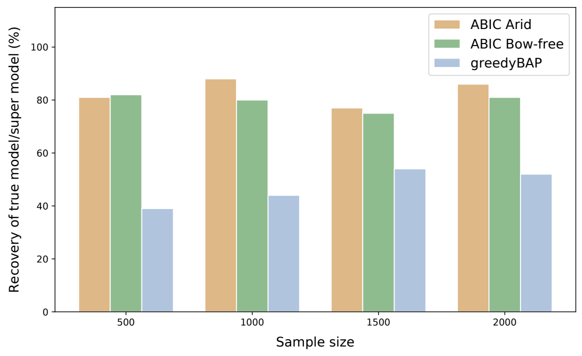

Figure C: Bar plots showing rate of recovery of the true equivalence class *or* a super model of the true equivalence class of ADMGs with a Verma constraint as a function of sample size. The underlying data is the same as the one used to generate the plots in Fig. 2.

λ	True model	Super model	Wrong model
5e-4	0.20	0.80	0.00
5e-3	0.25	0.70	0.05
$\underline{\text{5e-2}}$	$\underline{0.39}$	$\underline{0.41}$	$\underline{0.20}$
5e-1	0.01	0.00	0.99
5e0	0.00	0.00	1.00

Table B: Analysis of different settings of L_0-regularization parameter λ in the ABIC bow-free procedure. We report the fraction of times the procedure recovered the true model (or one that is equivalent to it), a super model of the true model, or an incorrect independence model. The underlying data for the experiment is the same as the one used to generate the bar plots in Fig. 2 for $n = 1000$. We use the underlined $\lambda = $ 5e-2 for all experiments.

E ADDITIONAL RESULTS AND EXPERIMENTS

In this section we provide additional results and experiments that were excluded from the main draft due to space constraints.

Fig. C provides additional insight into the modes of failure for each algorithm used to recover Verma constraints in the experiments corresponding to the bar plots in Fig. 2 of the main draft. It is easy to see from Fig. C that more often than not, the arid and bow-free ABIC methods yield an equivalent model or a super model of the true ADMG while the greedyBAP method more often returns an incorrect model.

Table B shows the results obtained from the ABIC bow-free procedure for different settings of regularization stength λ. Results are shown for the same task as in Fig. 2 of recovering ADMGs with a Verma constraint for sample size $n = 1000$. As expected, for low values of λ, the procedure is more likely to return a denser ADMG corresponding to a super model of the true model. As λ increases, the procedure recovers the true model more often, and finally for relatively large values of λ the procedure almost always returns a sparser ADMG corresponding to an incorrect independence model.

Finally, we present results for 15 variable ADMGs in Table C to supplement the 10 variable experiments in Table 2 of the main paper. We observe similar trends showing that our method performs favorably in comparison to baselines for recovery of both arid and ancestral ADMGs.

	Skeleton		Arrowhead		Tail	
Method	tpr ↑	fdr ↓	tpr ↑	fdr↓	tpr ↑	fdr ↓
gBAP (Nowzohour et al., 2017)	0.80	0.27	0.28	0.53	0.02	0.42
ABIC (bow-free)	**0.83**	**0.15**	**0.69**	**0.23**	**0.26**	**0.41**

	Skeleton		Arrowhead		Tail	
Method	tpr ↑	fdr ↓	tpr ↑	fdr↓	tpr ↑	fdr ↓
FCI (Spirtes et al., 2000)	0.29	0.11	0.24	0.56	0.05	0.74
gSPo (Bernstein et al., 2020)	**0.87**	0.23	0.41	0.62	0.31	0.88
ABIC (ancestral)	0.77	**0.09**	**0.66**	**0.24**	**0.62**	**0.68**

Table C: Comparison of our method to greedyBAP (left) and FCI (right) for recovering 15 variable arid and ancestral ADMGs respectively. The metrics reported are analogous to Table 2 in the main text. (↑/↓ indicates higher/lower is better.)

References

Bernstein, D., Saeed, B., Squires, C., and Uhler, C. (2020). Ordering-based causal structure learning in the presence of latent variables. In *International Conference on Artificial Intelligence and Statistics*, pages 4098–4108. PMLR.

Bhattacharya, R., Nabi, R., and Shpitser, I. (2020). Semiparametric inference for causal effects in graphical models with hidden variables. *arXiv preprint arXiv:2003.12659*.

Butler, S. K. (2008). *Eigenvalues and structures of graphs*. PhD thesis, UC San Diego.

Drton, M., Eichler, M., and Richardson, T. S. (2009). Computing maximum likelihood estimates in recursive linear models with correlated errors. *Journal of Machine Learning Research*, 10:2329–2348.

Haughton, D. M. (1988). On the choice of a model to fit data from an exponential family. *Annals of Statistics*, 16(1):342–355.

Maclaurin, D. (2016). *Modeling, inference, and optimization with composable differentiable procedures*. PhD thesis.

Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Autograd: Effortless gradients in Numpy. In *ICML 2015 AutoML Workshop*, volume 238, page 5.

Nowzohour, C., Maathuis, M. H., Evans, R. J., Bühlmann, P., et al. (2017). Distributional equivalence and structure learning for bow-free acyclic path diagrams. *Electronic Journal of Statistics*, 11(2):5342–5374.

Richardson, T. S. (2003). Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30(1):145–157.

Richardson, T. S., Evans, R. J., Robins, J. M., and Shpitser, I. (2017). Nested Markov properties for acyclic directed mixed graphs. Working paper.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Shpitser, I., Evans, R. J., and Richardson, T. S. (2018). Acyclic linear SEMs obey the nested Markov property. In *Proceedings of the 34th Annual Conference on Uncertainty in Artificial Intelligence*.

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

Tian, J. and Pearl, J. (2002). On the testable implications of causal models with hidden variables. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 519–527.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7154–7163.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pages 9472–9483.