

**Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596**

**AN ALGORITHM FOR DECIDING IF A SET OF OBSERVED
INDEPENDENCIES HAS A CAUSAL EXPLANATION**

**T. S. Verma
J. Pearl**

**February 1992
CSD-920018**

An Algorithm for Deciding if a Set of Observed Independencies Has a Causal Explanation

TS Verma* Judea Pearl
<verma@cs.ucla.edu> <judea@cs.ucla.edu>
Cognitive Systems Laboratory
Computer Science Department
University of California
Los Angeles, CA 90024

February 20, 1992

Abstract

In a previous paper [8] we presented an algorithm for extracting causal influences from independence information, where a causal influence was defined as the existence of a directed arc in all minimal causal models consistent with the data. In this paper we address the question of deciding whether there exists a causal model that explains ALL the observed dependencies and independencies. Formally, given a list M of conditional independence statements, it is required to decide whether there exists a directed acyclic graph D that is perfectly consistent with M , namely, every statement in M , and no other, is reflected via d -separation in D . We present and analyze an effective algorithm that tests for the existence of such a dag, and produces one, if it exists.

Key words: Causal modeling, graphoids, conditional independence.

*TS Verma is a PhD student at UCLA

1 Introduction

Directed acyclic graphs (dags) have been widely used for modeling statistical data. Starting with the pioneering work of Sewal Wright [14] who introduced path analysis to statistics, through the more recent development of Bayesian networks and influence diagrams, dag structures have served primarily for encoding causal influences between variables as well as between actions and variables.

Even statisticians who usually treat causality with extreme caution, have found the structure of dags to be an advantageous model for explanatory purposes. N. Wermuth [13], for example, mentions several such advantages. First, the dag describes a stepwise stochastic process by which the data *could have been* generated and in this case it may even “prove the basis for developing causal explanations” [1]. Second, each parameter in the dag has a well understood meaning since it is a conditional probability, i.e., it measures the probability of the response variable given a particular configuration of the explanatory (parents) variables and all other variables being unspecified. Third, the task of estimating the parameters in the dag model can be decomposed into a sequence of local estimation analyses, each involving a variable and its parent set in the dag. Fourth, general results are available for reading all implied independencies directly off the dag [12], [6], [5] and for deciding from the topology of two given dags whether they are equivalent, i.e., whether they specify the same set of independence-restrictions on the joint distribution [2], [11], and whether one dag specifies all the restrictions specified by the other [7]¹.

This paper adds a fifth advantage to the list above. It presents an algorithm which decides for an arbitrary list of conditional independence statements whether it defines a dag and, if it does, a corresponding dag is drawn. The algorithm we present has its basis in the “Inferred-Causation” (IC) algorithm described in [8] and in Lemmas 1 and 2 of [11]. However, whereas in [8] we were interested in detecting local relationships that we called “genuine causal influences”, we now consider an entire dag as one unit which ought to fit the data at hand.

¹The criterion for dag equivalence is given in Lemma 3.1. It follows from Frydenberg’s analysis of chain graphs, which applies to strictly positive distributions. The more direct analysis of Verma and Pearl [11] renders the criterion applicable to arbitrary distributions, as well as to non-probabilistic dependencies of the graphoid type [9].

1.1 Problem

Given a list M of conditional independence statements² ranging over a set of variables U it is required to decide whether there exists a directed acyclic graph (dag) D that is consistent with M .

1.2 Definitions

A *dependency model* is a list of conditional independence statements of the form $I(A, B|C)$, where A , B and C are disjoint subsets of some set of variables U . A dag D is *consistent* with a dependency model M if every statement in M and no statement outside M follows from the topology of D . In this case, M is said to be *dag-isomorphic*. A statement I follows from the topology of a dag D , if I holds in every probability distribution that is compatible with D ³. A probability distribution P is compatible with D if it can be decomposed into a product of conditional probabilities $P(a|\bar{\pi}(a))$, over all nodes $a \in U$, where $\bar{\pi}(a)$ is a set containing the parents of a in D . Finally, a statement $I(A, B|C)$ holds in a probability distribution P iff $P(A|C)P(B|C) = P(AB|C)$.

The following definitions and notation are needed to understand the proposed solution. A *partially directed acyclic graph* (pdag) is a graph which contains both directed and undirected edges, but it does not contain any strictly directed cycles. An *extension* of a pdag G , is any fully directed acyclic graph, D , which has the same skeleton (underlying undirected graph) as G and the same vee structures as G . Three nodes form a *vee structure*, written \overline{abc} , if $a \rightarrow b \leftarrow c$ and a is not adjacent to c . Two nodes are *adjacent*, written \overline{ab} , if either $a \rightarrow b$, $a \leftarrow b$ or $a - b$.

1.3 Overview

Section 2 details the solution to the problem posed in Section 1.1. It presents an algorithm which consists of the following three phases.

- Phase 1 examines the independence statements in M and tries to construct a pdag, G with the following guarantees:

²We assume that M is closed under the graphoid axioms. See remark in Section 5 for processing lists that are not closed.

³Alternatively, such a statement corresponds to a d-separation condition in D [6].

1. If M is dag-isomorphic then every extension of G will be consistent with M .
 2. If Phase 1 fails to generate a pdag, then M is not dag-isomorphic.
- Phase 2 extends a pdag, G , into a dag D , if possible.
 - Phase 3 verifies if D is consistent with M .

If D is found to be consistent with M then M is dag-isomorphic, by definition. If D is found to be inconsistent with M then M is not dag-isomorphic and (by definition) no dag can be consistent with M .

Additional improvements to this algorithm and extensions to the problem are discussed in Section 5.

2 The DAG Construction Algorithm

Phase 1

Generate a pdag G , from M , if possible.

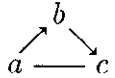
1. For each pair of variables, (a, b) , look through M for a statement of the form $I(a, b|S)$, where S is any set of variables (including \emptyset). Construct an undirected graph G where vertices a and b are connected by an edge iff a statement $I(a, b|S)$ is not found in M . Mark every pair of non-adjacent nodes in G with the set S found in M , call this set $S(a, b)$.
2. For every pair of non-adjacent nodes a and c in G , test if there is a node b not in $S(a, c)$ that is adjacent to both a and c . If there is such a node then direct the arcs $a \rightarrow b$ and $c \rightarrow b$ unless there already exists a directed path from b to a or from b to c , in which case Phase 1 FAILS.
3. If the orientation of Step 2 is completed then Phase 1 SUCCEEDS, and returns a partially directed graph, G .

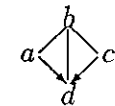
Phase 2

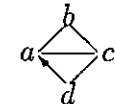
Extend G into a dag, D , if possible.

1. Initially let C be an empty stack and let D equal G .
2. While D contains any undirected arcs repeat 2a, 2b and 2c:
 - (a) Close D under the following four rules, if possible.

Rule 1: If $a \rightarrow b - c$ and a is not adjacent to c then direct $b \rightarrow c$.

Rule 2: If  then $a \rightarrow c$.

Rule 3: If  then direct $b \rightarrow d$.

Rule 4: If  then direct $a \rightarrow b \leftarrow c$.

- (b) If the closure was successful, i.e. there are no directed cycles or new vee structures, then:
 - If D still contains any undirected arcs, select one and choose a direction for it, push the arc and a copy of D onto the stack C and continue the while loop (i.e. go back to 2a).
 - If G contains no more undirected arcs, then the while loop is completed, Phase 2 SUCCEEDS, and returns a directed acyclic graph D .
- (c) If the closure was unsuccessful, then discard the current value of D and pop the most recent copy off of the stack along with the selected arc. Reverse the chosen direction of the arc in D and continue the while loop (i.e. go back to 2a).

Phase 3

Check if D is consistent with M .

1. Test that every statement I in M holds in D (using the d-separation criterion)⁴.

⁴A linear time algorithm for testing d-separation is reported in [4].

2. Pick any total ordering of the nodes which agrees with the directionality of the D and let U_a stand for the set of nodes which precede a in this ordering. For every node a in D , test if the statement $I(a, U_a \setminus \bar{\pi}(a) | \bar{\pi}(a))$ is in M .
3. If both tests are confirmed, EXIT with SUCCESS, and return D ; else, EXIT with FAIL.

3 Correctness

Phase 1

This phase examines M and generates a graph, G subject to the above guarantees, if possible. That is, if M is dag-isomorphic then every extension of G is consistent with M . The correctness of Step 1 of this phase follows from the following lemma [11] (a detailed proof of which can be found in [10]. appendix):

Lemma 3.1 *Let M be any dag isomorphic dependency model, a dag D is consistent with M iff the following two conditions hold:*

1. \overline{ab} in D iff $\forall_S, I(a, b|S) \notin M$.
2. \overline{abc} in D iff \overline{abc} and $\neg \overline{ac}$ in D and \forall_S , if $I(a, c|S) \in M$ then $b \notin S$.

The *only-if* portion of this lemma guarantees that:

1. If there exists some dag D^* which is consistent with M , then any dag D consistent with M must have the same skeleton as D^* .
2. Furthermore, every dag D , consistent with M must have the same vee structures as D^* .

The *if part* guarantees that every dag D which has the same skeleton and vee structures as D^* , is consistent with M .

The first step of Phase 1 attempts to construct this invariant skeleton, if M is dag-isomorphic. The arrowheads added in the second step identify the invariant vee structures, again, if M is dag-isomorphic.

Note however, that Step 2 of Phase 1, directs arcs immediately upon finding one set S satisfying condition 2 of the lemma. This decision is correct due to the following lemma:

Lemma 3.2 *For any dag-isomorphic dependency model M and any three variables a , b and c forming a chain \overline{abc} ,*

if \exists_S s.t. $I(a, c|S) \in M$ and $b \notin S$ then $\forall_{S'} I(a, c|S') \in M$ implies $b \notin S'$.

Proof: Suppose \overline{abc} and \exists_S s.t. $I(a, c|S) \in M$ and $b \notin S$. In order for S to d-separate a and c , it must be the case that $a \rightarrow b \leftarrow c$ — if b were not head-to-head then this two link path would be active given any set not containing b . Now since b is head-to-head it must be the case that any set S' which contains b will activate this two link path, hence for any S' if $I(a, b|S') \in M$ then $b \notin S'$. \square

This lemma permits the use of the first S found to orient the vee structures.

If M is not dag-isomorphic it would be possible for Phase 1 to build a graph that is not a pdag if it weren't for the failure condition in Step 2. The next example illustrates a failure resulting from an application of Phase 1 on a non-dag-isomorphic dependency model.

Example 3.3 *Let $U = \{a, b, c, d\}$ and M be the closure of the set $\{I(a, c|\emptyset), I(a, d|\emptyset), I(b, d|\emptyset)\}$ under symmetry⁵.*

Step 1 of Phase 1 will construct the skeleton $a - b - c - d$, and $S(a, c) = S(a, d) = S(b, d) = \emptyset$. Since there is a chain \overline{abc} and $\neg\overline{ac}$ and $b \notin S(a, c)$ Step 2 could direct $a \rightarrow b \leftarrow c$. Similarly since \overline{bcd} and $\neg\overline{bd}$ and $c \notin S(b, d)$, Step 2 could direct $b \rightarrow c \leftarrow d$.

One of the two directions would be assigned first, then upon attempting the second the algorithm would FAIL.

Phase 2

The task of Phase 2 is to find a whether a pdag, G , has any extensions and to find one if such exists. This is a purely graph theoretic task; it does not involve M .

⁵Symmetry states that $I(A, B|C)$ iff $I(B, A|C)$. Unless otherwise noted, dependency models are assumed to be closed under symmetry since this is a trivial operation.

To prove that this phase of the construction is correct, it is sufficient to prove that each of the four rules is sound, namely, that the orientation choices dictated by these rules never need to be revoked.

- Rule 1: If $a \rightarrow b - c$ and a is not adjacent to c then direct $b \rightarrow c$.

Directing $b - c$ as $b \leftarrow c$ would create a new vee structure, \overline{abc} , thus if there is a consistent extension it must contain $b \rightarrow c$.

- Rule 2: If $\begin{array}{ccc} & b & \\ a & \nearrow & c \\ & \longleftarrow & \end{array}$ then $a \rightarrow c$.

Directing $a - c$ as $a \leftarrow c$ would create a directed cycle, $[abca]$, thus if there is a consistent extension it must contain $a \rightarrow c$.

- Rule 3: If $\begin{array}{ccc} & b & \\ a & \nearrow & c \\ & \searrow & \\ & d & \end{array}$ then direct $b \rightarrow d$.

Directing $b - d$ as $b \leftarrow d$ would imply that $a - b$ must be directed as $a \rightarrow b$ or else there would be a directed cycle, $[adba]$. Now if $b - c$ is directed as $b \rightarrow c$ then there is a directed cycle, $[bcd b]$, and if it is directed as $b \leftarrow c$ then there is a new vee structure, \overline{abc} . Thus if there is a consistent extension it must contain $b \rightarrow d$.

- Rule 4: If $\begin{array}{ccc} & b & \\ a & \nearrow & c \\ & \searrow & \\ & d & \end{array}$ then direct $a \rightarrow b \leftarrow c$.

First, $a - b$ must be directed as $a \rightarrow b$ or there would be a new vee structure, \overline{dab} . If $b - c$ is directed as $b \rightarrow c$ then $c - d$ cannot be directed as $c \rightarrow d$ or there would be a directed cycle, $[cdabc]$. Moreover, $c - d$ cannot be directed as $c \leftarrow d$ or there would be a new vee structure, \overline{bcd} . Thus if there is a consistent extension, then it must contain $a \rightarrow b \leftarrow c$.

Following are two simple examples of pdags which cannot be extended into dags.

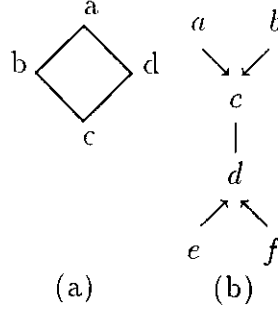


Figure 1: Two pdags which cannot be extended.

Example 3.4 Consider the graph of Figure 1.a. Initially, no rules apply, so the algorithm would select an arc and direct it, e.g. $a \rightarrow b$. Now Rule 1 will apply twice, directing $b \rightarrow c \rightarrow d$. However a third application to infer $d \rightarrow a$ would produce a directed cycle. It is easy to see that a cycle would even if the chosen direction of the selected arc were reversed. Thus this graph has no dag extension.

Example 3.5 Consider the graph of Figure 1.b. Any application of Rule 1 to direct the arc $c - d$ would create a new vee structure. Hence this graph as well, has no dag extension.

Phase 3

The soundness of Step 1 follows from the definition of consistency; it simply checks if each and every independence statement of M is represented in D . The soundness of Step 2, namely that testing only statements of the form $I(a, U_a \setminus \bar{\pi}(a) | \bar{\pi}(a))$ is sufficient follows from the proof of the soundness of d-separation[12].

Example 3.6 Let $U = a, b, c$ and $M = \{I(a, b | \emptyset), I(a, c | \emptyset), I(b, c | \emptyset)\}$. Phase 1 will produce an empty graph which can trivially be extended into an empty dag. But every independence statement is true in an empty dag, including, e.g. $I(a, b | c)$ which is not in M . Thus M is not dag isomorphic.

Example 3.7 Let $U = a, b, c, d$ and $M = \{I(a, b|\emptyset), I(a, b|d), I(ab, d|c)\}$. Phase 1 will produce G and Phase 2 will extend it into D . But D does not reflect $I(a, b|d) \in M$. Thus M is not dag isomorphic.

4 Complexity Analysis

Phase 1 can be completed in $O(|M| + |U|^2)$ steps, as follows:

- Start with a complete graph G . For each statement, $I(A, B|S)$ in M , and for each pair of variables $a \in A$, and $b \in B$ remove the links $a - b$ from G and define $S(a, b) = S$.
- For each node a let $N(a) = \{b|a - b\}$ be the set of neighbors of a .
- For each separating set $S(a, b)$ defined above, note that $C(a, b) = N(a) \cup N(b) \setminus S(a, b)$ must be children of a and b so direct $a \rightarrow c \leftarrow b \forall c \in C(a, b)$.

Phase 2 may appear to require an exponential amount of time in the worst case due to possible backtracking in Step 2(c). However, we conjecture that if G is extendible, then Rules 1-4 are sufficient to guarantee that no choice will ever need to be revoked. Empirical studies have, so far, confirmed our conjecture. Thus it would be possible to replace the backtrack step with a definite failure, in which case the time complexity of this phase would be polynomial, no more than $O(|U|^4 * |E|)$.

Phase 3 can be completed in $O(|M| * |E| + |M| * |U|)$ steps.

5 Extensions and Improvements

In general, the set of all independence statements which hold for a given domain will grow exponentially as the number of variables grows. Thus it might be impractical to specify M by explicit enumeration of its I-statements. In such cases it may be desirable, instead, to specify a basis, L , such that M is the logical closure of L , (i.e. $M = CL(L)$), relative to some semantics, (e.g. the graphoid axioms, correlational graphoids axioms, or even probability theory).

The major difficulty in permitting the dependency model to be specified as the closure of some basis lies in solving the so called *membership problem*. Simply stated, the problem is to decide if a particular statement, I_0 , is contained in the closure, M , of a given list of statements, L . In general, membership problems are often undecidable, and of those that are decidable, many are NP-complete. In particular, the membership problems for both graphoids and probabilistic independence are unsolved [3].

However, in spite of this difficulty, it may still be possible to have an efficient dag construction algorithm, because the queries required are of a special form. The algorithm makes four types of queries to M :

1. (Phase 1, Step 1) “Is there any S such that $I(a, b|S) \in CL(L)$?”
2. (Phase 1, Step 2) “Is b in any set S such that $I(a, c|S) \in CL(L)$?”
3. (Phase 3, Step 1) “Is every statement in $CL(L)$ represented in D ?”
4. (Phase 3, Step 2) “Is every statement represented in D in $CL(L)$?”

In the case that M is assumed to be the graphoid closure of L , queries of type 1, 2 and 3 are all manageable. The queries for Phase 1 can both be quickly answered due to the following lemma⁶:

Lemma 5.1 *If \exists_S s.t. $I(a, b|S) \in CL(L)$ then $\exists_{A,B,C}$ s.t. $I(aA, bB|C) \in L$*

Remark: Note that this simplification is possible due to the special form of these queries, namely that a and b are both singletons and any separating set will suffice.

Type 3 queries pose no particular problem since the axioms of graphoids hold for d-separation. Thus it is enough to check that each statement in L is represented in D to ensure that the every statement in closure of L is represented in D .

However, to check that each statement represented in D is contained in $CL(L)$ it is necessary to make the $|U|$ membership queries explicated in Step 2 of Phase 3. Although these statements have a special form, it is yet unclear whether a lemma similar to 5.1 exists to simplify these queries.

Another possible source for simplification is to note that the dag D being tested in Step 2 of Phase 3 is not just any random dag, but the output of the

⁶This lemma follows immediately from the form of the graphoid axioms.

construction algorithm. While Example 3.6 demonstrates that it is possible for D to contain I-statements which are not in $CL(L)$, it may still be the case that any such I-statements must have either a certain form or some other property that would simplify the membership query.

References

- [1] D. R. Cox. Causality; some statistical aspects. To appear in *J. Roy. Statist. Soc. Ser. A*.
- [2] M. Frydenberg. The chain graph markov property. *Scand. J. Statist.*, 17:333 – 353, 1990.
- [3] D. Geiger. *Graphoids – A Qualitative Framework for Probabilistic Inference*. PhD thesis, UCLA, 1990.
- [4] D. Geiger, T. S. Verma, and Judea Pearl. Identifying independence in bayesian networks. *Networks*, 20:507 – 534, 1990.
- [5] S. L. Lauritzen, A. P. Dawid, B. Larsen, and H. G. Leimer. Independence properties of directed markov fields. *Networks*, 20:491–505, 1990.
- [6] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufman, San Mateo, CA, 1988.
- [7] J. Pearl, D. Geiger, and T. S. Verma. The logic of influence diagrams. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Networks and Decision Analysis*, pages 67 – 87. John Wiley and Sons, Ltd., Sussex, England, 1989.
- [8] J. Pearl and T. S. Verma. A theory of inferred causation. In J. A. Allen, R. Fikes, and E. Sandwall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441 – 452. Morgan Kaufmann, San Mateo, 1991.
- [9] Judea Pearl and Azaria Paz. Graphoids: A graph-based logic for reasoning about relevance relations. In B. Du Boulay et al., editor, *Advances in Artificial Intelligence-II*, pages 357–363. North Holland, Amsterdam, 1986.

- [10] T. S. Verma. Invariant properties of causal models. Technical Report R-134, UCLA Cognitive Systems Laboratory, 1991.
- [11] T. S. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings 6th Conference on Uncertainty in AI*, pages 220 – 227, 1990.
- [12] T.S. Verma. Causal networks: Semantics and expressiveness. Technical Report R-65, UCLA Cognitive Systems Laboratory, 1986. Also in: R. Shachter, T.S. Levitt and L.N. Kanal, editors, *Uncertainty in AI 4*, pages 325-359, Elsevier Science Publishers 1989.
- [13] N. Wermuth. On block-recursive linear regression equations. Technical Report ISSN 0177-0098, Psychological Institute, University of Mainz, Mainz, FRG, September, 1991. Forthcoming in the Brazilian Journal of Probability and Statistics.
- [14] S. Wright. Correlation and causation. *J. Agricult. Res.*, 20:557 – 585, 1921.

