An Algorithm for Fast Recovery of Sparse Causal Graphs

Peter Spirtes and Clark Glymour[1]

Abstract

Previous asymptotically correct algorithms for recovering causal structure from sample probabilities have been limited even in sparse causal graphs to a few variables. We describe an asymptotically correct algorithm whose complexity for fixed graph connectivity increases polynomiaily in the number of vertices, and may in practice recover sparse graphs with several hundred variables. From sample data with n = 2,000, an implementation of the algorithm on a Decstation 3100 recovers the edges in a linear version of the ALARM network with 37 vertices and 46 edges. Fewer than 8% of the undirected edges are incorrectly identified in the output. Without prior ordering information the program also determines the direction of edges for the ALARM graph with an error rate of 17%. Processing time is less than 15 seconds.

Recovering Causal Relations

Consider pairs <g, P> for which g is a directed acyclic graph and P is a probability distribution on the vertices of g such that (i) for every vertex v and every set S_v of vertices that are neither parents nor descendents of v, v and S_v are independent conditional on the parents of v; and (ii) every independence relation in P is a consequence of the independence relations in (i). Pairs satisfying these conditions can be viewed as causal structures in which the causal dependencies generate statistical dependencies. When the set of measured variables for which probabilities are provided in the data is such that every common cause of a measured variable is measured, we say the structure is causally sufficient.

[1] We thank Gregory Cooper for a conversation that stimulated this work.

Recovery problems have to do with determining g, or features of g, from the distribution P or from samples obtained from P. In Spirtes, Glymour, and Scheines(forthcoming) we proposed the following algorithm for the recovery problem with causally sufficient structures, using as input independence and conditional independence facts about P[2]:

SGS Algorithm

(A) Start with the empty graph.

(B) For each vertex pair a, b, place an undirected edge between a and b if and only if ~I(a,S,b) for each subset S not containing a or b. Call this undirected graph G.

(C) For each triple a, b, c of vertices such that a and b are adjacent in G, b and c are adjacent in G, and a and c are not adjacent in G, direct the edges a - b and b - c into b if and only if for every set S of vertices containing b but not a or c, ~I(a, S,b).

(D) Output all orientations of the graph consistent with (C).

Verma and Pearl (1990) subsequently proved the correctness of the algorithm and offered a variant that outputs a pattern rather than a collection of graphs. The pattern has an undirected edge between two vertices if the SGS output contains two graphs that orient the edge in different directions; the pattern contains a directed edge if

[2]We denote by "I(a, S, b)" the claim that variables a and b are independent conditonal on the set of variables in S, and by "-I(a,S,b) the denial of that claim.

every graph output by the SGS algorithm has the edge so oriented; and the pattern may have a bidirected edge, e.g., a <-> b provided step C of the algorithm determines that the a - b edge collides with another edge at a and also collides with another edge at b. When all common causes are measured and the data consist of the actual independence and conditional independence relations, the pattern is simply a representation of the class output by the SGS algorithm, but when there are unmeasured common causes or independence facts due to sampling variation rather than to P, the pattern is more general.

Two graphs, g, g' are statistically indistinguishable provided that for every probability distribution P, <g,P> satisfies the conditions (i) and (ii) of the first paragraph if and only if <g',P> does. From the independence facts of a distribution P such that <g,P> satisfies (i), and (ii), the SGS algorithm returns all and only the graphs statistically indistinguishable from g.

In the worst case, the SGS algorithm requires a number of conditional independence facts that increases exponentially with the number of vertices, as must any algorithm based on conditional independence relations. But because for any undirected edge that is in the graph g, the number of conditional independence facts that must be generated and checked in stage (B) of the algorithm is unaffected by the connectivity of the true graph, even for sparse graphs the algorithm rapidly becomes computationally infeasible as the number of vertices increases. Besides problems of computational feasibility, the algorithm has problems of reliability when applied to sample data. The determination of higher order conditional independence relations from sample distributions is generally less reliable than is the determination of lower order independence relations. With, say, 37 binary variables, to determine the conditional independence of two variables on the set of all remaining variables requires considering the relations among the frequencies of 235 distinct states, only a tiny fraction of which will be instantiated even in very large samples.

To illustrate the difficulty of recovering the graph g (or a set of equivalent graphs) from the probability distribution P, consider an example due to Herskovits and Cooper (1990). Their Kutato[1] Algorithm is a heuristic entropy minimization procedure for recovering a directed graph given sample data and a total ordering of the vertices such that vi > V2 implies that there is no directed edge from V2 to vi. The asymptotic reliability of the procedure is unknown. Nonetheless from large sample data the algorithm recovers most of the connections on a sparse graph-the ALARM network (Beinlich, Suermondt, Chavez, and Cooper1989)--with 37 variables and 46 edges. In their example, the direction of the edges is not recovered from the data but is determined by the prior ordering given to the computer.[3] See Figure 1.

[3]Herskovits and Cooper say that a variant of the Kutato' algorithm can determine the orientation of edges without a prior ordering of the variables, but they do not describe the properties of the application or give an example. They are also investigating Bayesian alternatives that are much faster than the Kutato' procedure.

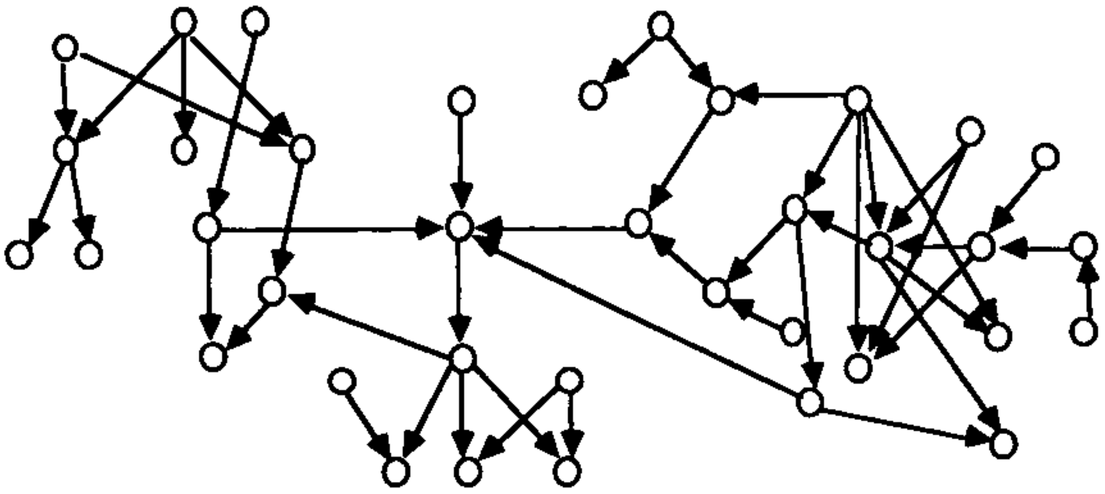

Fig. 1: Alarm Network

Using 10,000 cases an implementation on a Macintosh II required about 22 and one half hours, about a quarter of which was required to read the data-base. The output omitted two correct edges and included two false edges. By comparison, the SGS algorithm has been implemented in the TETRAD II program using partial correlation tests for conditional independence. Run on a DEC workstation with 20 megabyte RAM the procedure stops at about 17 variables because of space requirements to store the conditional independence facts. Space could be traded for time, but the ALARM case is out of sight.

Verma and Pearl have suggested an improvement on the SGS algorithm. For each pair of variables a, b introduce an undirected edge between them if they are dependent conditional on the set of all other variables. Call the resulting network N. (In the true graph, G, the parents of any variable form a maximal complete subgraph-a clique-in the network N.) Again for each pair of variables a, b adjacent in N, determine if a, b are dependent conditional on all subsets of variables in the cliques in N containing a or b. If so a is adjacent to b in G. The complexity is thus bounded by the size of the largest clique in N.

The practical value of the improvement is limited by the fact that conditional independence relations of the order of the number of vertices of the graph (minus two) must still be estimated, with consequent costs in computational efficiency and reliability. With discrete data the great majority of the corresponding states will not be instantiated in the data, and with data from linear structures the formula for higher order correlations is recursive: to compute the partial correlations of nth order, three partial correlations of order n-1 must be determined, and so on.

We should like an algorithm that has the same input/output relations as the SGS procedure but for sparse graphs does not require the determination of higher order independence relations, and in any case requires as few conditional independence relations as possible. The following procedure starts by forming the complete undirected graph, then "thins" that graph by removing edges with zero order conditional independence relations, thins again with first order conditional independence relations, and so on. The set of variables conditioned on need only be a subset of the set of variables adjacent to one or the other of the variables conditioned, and can even be confined to adjacent variables on certain undirected paths.

PC Algorithm:

Let Acab denote the set of vertices adjacent to a or to b in graph C, except for a and b themselves. Let Ucab denote the set of vertices in graph C on (acyclic) undirected paths between a and b, except for a and b themselves. (Since the algorithm is continually updating C, Acab and Ucab are constantly changing as the alogorithm progresses.)

A.) Form the complete undirected graph C on the vertex set V.

B.)

```
    n = 0.
    repeat
        For each pair of variables a, b adjacent in C, if Acab
        n Ucab has cardinality greater than or equal to n and
        a, b are independent conditional on any subsets of
        Acab n Ucab of cardinality n, delete a-b from C.
        n = n + 1.
    until for each pair of adjacent vertices a, b, Acab n
    Ucab is of cardinality less than n.
```

C.) Let F be the graph resulting from step B. For each triple of vertices a, b, c such that the pair a, b and the pair b,c are each adjacent in F but the pair a, c are not adjacent in F, orient a - b - c as a -> b <- c if and only if a and c are dependent on every subset of Apac n Upac containing b. Output all graphs consistent with these orientations.

Note that Acab n UQab is not in general the set of *parents* of a or b (in the oriented graph) on undirected paths between a, b, since descendents of a, b may also occur.

An obvious modification of the algorithm will generate patterns rather than collections of graphs.

The complexity of the algorithm for a graph G is bounded by max(|AGab|) over all pairs of vertices a,b, which is never more than the sum of the two largest degrees in G. Generally stage B of the algorithm continues testing for some steps after the correct undirected graph has been identified. The number of steps required before the true graph is found (but not necessarily until the algorithm halts) depends on the maximal number of treks[4] between a pair of variables, say a, b, that share no vertices adjacent to a or b. If these maximal numbers are held constant as the number of vertices increases, so that k, the maximal order of the conditional independence relations that need be tested, does not change, then the worst case computational demands of the algorithm increase as

$$\frac{n!}{2!(n-2)!} 2^k$$

[4]A trek is a pair of directed paths from some vertex z to a, b respectively, intersecting only at z, or a directed path from a to b or a directed path from b to a.

which is bounded by n^2. It should be possible to recover sparse graphs with as many as several hundred variables. Of course the computational requirements increase exponentially with k.

In many cases it may be more efficient to perform conditional independence tests on all subsets of Acab rather than to compute I)Gab. We have not yet theoretically determined the trade-off.

The structure of the algorithm and the fact that it continues to test even after having found the correct graph suggest a natural heuristic for very large variable sets whose causal connections are expected to be sparse, namely to set a fixed bound on the order of conditional independence relations that will be considered.

Proposition: The PC and SGS algorithms give the same output.

Proof:

Let PGab denote the set of vertices in directed graph G that are parents of a or of b, except for a and b themselves. We note a lemma:

Lemma: In any pair <G, P> meeting conditions i and ii,. if vertices a, b are not adjacent then they are independent conditional on PGab n **UGab.**

The proof is a trivial modification of the argument Verma and Pearl give for their Lemma 1.[5]

Now we show that steps A and B of the PC algorithm produce the correct undirected graph. Let G be a directed graph produced by the SGS algorithm. (Every graph produced by the SGS algorithm shares the same underlying undirected graph.) First we will show that every edge in the undirected graph of G is also in the undirected graph F. The algorithm starts with a complete graph C, and only removes an edge between a and b if at some stage of construction of C, a and b are independent on some subset of $A_C ab \cap U_C ab$. However, if the edge between a and b is in the undirected graph of G, then a and b are not independent on *any* subset of variables not containing a or b. Every edge in the undirected graph of G is also in C at every stage of construction (and hence in F, which is C at its final stage of construction.)

We will now show that if a and b are not adjacent in the undirected graph of G, then a and b are not adjacent in F. If a and b are not adjacent in G, then a and b are independent on some subset of variables not containing a or b. By the lemma, then, a and b are independent conditional on the set $P_G ab \cap U_G ba$. Since every edge in the undirected graph of G is in C at every stage of construction, $P_G ab \cap U_G ab$ is a subset of $A_C ab \cap$

[5]Verma (1990), pp. 221-222.

Ucab, and hence a and b are independent conditional on some subset of Acab n Ucab.

It remains only to show that step C of the algorithm orients the graph correctly. Assume that in G, a, c are not adjacent but a is adjacent to b and b is adjacent to c. In G, the a - b and b - c edges collide at b if and only if there is no set S containing b and not a or c such that a, c are independent conditional on S. Since a, c are not adjacent in G, they are independent conditional on the set PQac n Uoac. If the edges in G do not collide at b, then b is a parent of a or of c, so b is in Poac n Ucac, which is a subset of AFac n UFac containing b. If the edges do collide at b in G, then a, c are dependent on every set containing b and not a or c, and hence dependent on every subset of AFac n UFac that contains b.

An Application of the PC Algorithm

We have applied the PC algorithm to a linear version of the ALARM network. Using the same directed graph, linear coefficients with values between .5 and 1.0 were randomly assigned to each directed edge in the graph. Using a joint normal distribution on the variables of zero indegree, three sets of simulated data were generated, each with a sample size of 2,000. The covariance matrix and sample size were given to a version of the TETRAD II program with an implementation of the PC algorithm. This implementation takes as input a covariance matrix, and it outputs a pattern. It does not check to determine whether variables adjacent to vertices vi, V2 lie on an undirected path between vi and V2- No information about the orientation of the variables was given to the program. Run on a Decstation 3100, for each data set the program required less than fifteen seconds to return a pattern. In each trial the ouput patterm omitted three edges in the ALARM network; in one of the cases it also added two edges that were not present in the ALARM network. Of the 43 edges in the ALARM network that the program did find, the orientation of three of them is not determinable in principle from the probabilities. In the first trial the program correctly found all three edges that could not be directed in principle, in the second trial it incorrectly directed one edge that could not be directed and incorrectly failed to direct one edge that could be directed, and in the third trial it oriented one of the three. Of the remaining forty edges, the trials misoriented 5, 6, and 7 edges respectively, always by judging that an edge was directed into *both* of its vertices (as the pattern output allows) when in the ALARM graph it is directed into only one. The results are summarized below:

	# of omitted undirected edges	#false undirected edges	#orientation errors
Trial 1	3	0	5
Trial 2	3	2	6
Trial 3	3	2	7

The implementation used did not determine the adjacency sets lying on undirected paths between two variables because in this case with correlation data it was computationally cheaper to determine the partial correlations for all subsets of Acab than to keep track of Acab n Ucab. With discrete count data for which the determination of conditional independence relations is more computationally demanding, the alternative procedure described in our statement of the algorithm might be faster. For example, for one pair of vertices in the network, Aab consists of 8 vertices while Aab n Uab consists of only two vertices.

The comparison of 15 seconds for the PC algorithm with 22 and one half hours for the Kutato' algorithm should not be taken as a direct comparison of the efficiencies of the algorithms, since the DecStation 3100 is much faster than a Macintosh, and without the assumption of linearity considerably more time would be required in numerical operations to determine conditional independence.[6] Nonetheless, the PC algorithm appears to be very fast and reliable for sparse graphs. For similar data from a similarly connected graph with 100 variables, the present implementation should require less than two minutes.

[6]It may in fact be the case that for large samples and variable sets the errors introduced by assessing conditional independence through partial correlations or other aggregate measures are adequately repaid in time savings.

References

Beinlich, I., Suermondt, H., Chavez, R., & Cooper, G. (1989). The ALARM Monitoring System. (Technical Report KSL 88-84). Stanford University: Knowledge Systems Laboratory, Medical Computer Science.

Herskovits, E., & Cooper, G.. (1990). Kutato[1]: An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases. In Proceedings of the Sixth Conference on Uncertainty in Ai(pp. 54-62). Cambridge, MA.

Spirtes, P., Glymour, C, & Scheines, R. (forthcoming). Causality from Probability. In G. McKee (Ed.), Evolving Knowledge in Natural and Artificial Intelligence. Pitman.

Verma, T., & Pearl, J. (1990). Equivalence and Synthesis of Causal Models. In Proceedings of the Sixth Conference on Uncertainty in AI, (pp. 220-227). Cambridge, MA.