Automating Path Analysis for Building Causal Models from Data: First Results and Open Problems

Paul R. Cohen Lisa Ballesteros Adam Carlson Robert St.Amant
Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetts, Amherst
cohen@cs.umass.edu carlson@cs.umass.edu balleste@cs.umass.edu stamant@cs.umass.edu

Abstract

Path analysis is a generalization of multiple linear regression that builds models with causal interpretations. It is an *exploratory* or *discovery* procedure for finding causal structure in correlational data. Recently, we have applied path analysis to the problem of building models of AI programs, which are generally complex and poorly understood. For example, we built by hand a path-analytic causal model of the behavior of the Phoenix planner. Path analysis has a huge search space, however. If one measures N parameters of a system, then one can build $O(2^{N^2})$ causal models relating these parameters. For this reason, we have developed an algorithm that heuristically searches the space of causal models. This paper describes path analysis and the algorithm, and presents preliminary empirical results, including what we believe is the first example of a causal model of an AI system induced from performance data by another AI system.

1. INTRODUCTION

This paper describes a statistical *discovery procedure* for finding causal structure in correlational data, called *path analysis* [Asher, 83; Li, 75] and an algorithm that builds path-analytic models automatically, given data. This work has the same goals as research in function finding and other discovery techniques, that is, to find rules, laws, and mechanisms that underlie nonexperimental data [Falkenhainer & Michalski 86; Langley et al., 87; Schaffer, 90; Zytkow et al., 90].[1] Whereas function finding algorithms produce functional abstractions of (presumably) causal mechanisms, our algorithm produces explicitly causal models. Our work is most similar to that of Glymour et al. [87], who built the TETRAD system. Pearl [91; 93] and Spirtes [93] have recently developed causal induction algorithms with a more general mathematical basis than path analysis; they rely on evidence of nonindependence, a weaker criterion than path analysis, which relies on evidence of correlation. Those algorithms will be discussed in a later section.

We developed the path analysis algorithm to help us discover causal explanations of how a complex AI planning system works. The system, called Phoenix [Cohen et al., 89], simulates forest fires and the activities of agents such as bulldozers and helicopters. One agent, called the fireboss, plans how the others, which are semi-autonomous, should fight the fire; but things inevitably go wrong, winds shift, plans fail, bulldozers run out of gas, and the fireboss soon has a crisis on its hands. At first, this chaos was appealing and we congratulated ourselves for building such a realistic environment. However, we soon realized that we could explain very little about the behavior of the fireboss. We turned to regression analysis to answer some questions, such as, "Which has more impact on the time to contain a fire: the wind speed or the number of times the fireboss must replan?" But although regression assumed these factors interacted, it provided no explanation of their causal relationship. For example, we knew that the wind speed could affect the incidence of replanning and not vice versa, but this causal, explanatory knowledge was not to be found in regression models. Nor would automated regression procedures (e.g., stepwise multiple regression) find causal models of our planner. Path analysis, however, is a generalization of regression analysis that produces explicitly causal models. We built one such model of Phoenix by hand, and by automating path analysis as we describe below, we have been able to discover other causal explanations of how the Phoenix fireboss works.

Readers who are familiar with regression analysis might skip to the end of the next section, where we introduce path analysis, or Section 3 where we illustrate a path analysis of Phoenix. Section 4 describes our algorithm. Section 5 discusses two experiments, an informal one in which we applied the algorithm to Phoenix data, and a factorial experiment in which the behavior of the algorithm was probed by applying it to artificial data.

[1] The term "nonexperimental" is perhaps confusing, because data are usually collected in an experiment. Nonexperimental means that the experiment is over and the opportunity to manipulate variables to see effects has passed. Causal hypotheses must therefore be generated and tested with the data, alone.

2. BACKGROUND: REGRESSION

Path analysis is a generalization of multiple linear regression, so we will begin with regression. Simple linear regression finds a *least-squares* line relating a single predictor variable x to a performance variable y. A least-squares line is one that minimizes the sum of squared deviations of predicted values from actual values. That is, simple linear regression finds a line $\hat{y} = bx + a$ that minimizes $\sum_i (\hat{y}_i - y_i)^2$.

Multiple linear regression finds least-squares rules (i.e., planes and hyperplanes) for more than one predictor variable, rules of the form $\hat{y} = b_1x_1 + \ldots + b_kx_k + a$. The regression equation is better represented in standardized form as $\hat{Y} = \beta_1X_1 + \beta_2X_2 + \beta_3X_3$ (standardized variables are denoted with uppercase letters). The interpretation of this model is that a change in X_1 of one standard deviation, s_1, produces β_1 standard deviations change in $\hat{Y}$. Thus, beta coefficients are comparable: if $\beta_1 = .4$, $\beta_2 = .8$, then a standard deviation change in X_2 has twice the influence on Y as a standard deviation change in X_1.

To find beta coefficients the following equations are derived from the regression equation and solved:

$$\hat{Y}X_1 = \beta_1X_1^2 + \beta_2X_2X_1 + \beta_3X_3X_1$$
$$\hat{Y}X_2 = \beta_1X_1X_2 + \beta_2X_2^2 + \beta_3X_3X_2$$
$$\hat{Y}X_3 = \beta_1X_1X_3 + \beta_2X_2X_3 + \beta_3X_3^2$$

The previous equations can be rewritten in terms of correlations:

$$\begin{aligned} r_{YX_1} &= \beta_1 + \beta_2 r_{X_2X_1} + \beta_3 r_{X_3X_1} \\ r_{YX_2} &= \beta_1 r_{X_2X_1} + \beta_2 + \beta_3 r_{X_3X_2} \\ r_{YX_3} &= \beta_1 r_{X_3X_1} + \beta_2 r_{X_3X_2} + \beta_3 \end{aligned} \qquad (1)$$

Clearly, with these three equations we can solve for the three unknown beta coefficients. We have not shown that these coefficients guarantee that $\hat{Y} = \beta_1X_1 + \beta_2X_2 + \beta_3X_3$ is a least-squares rule, but the interested reader can find this demonstration in [Li, 75] or any good statistics text.

The three equations (1) are called the *normal equations*, and they have an interesting interpretation, illustrated in Figure 1. Consider the first normal equation, $r_{YX_1} = \beta_1 + \beta_2 r_{X_2X_1} + \beta_3 r_{X_3X_1}$. The β_1 term is represented in Figure 1 by the direct path between X_1, and Y; the second term is represented by the indirect path from X_1, through X_2 to Y; and the third term is represented by the indirect path through X_3. Thus, the correlation r_{YX_1} is given by the sum of the weights of three paths in Figure 1, where the weight of a path is the product of the coefficients (either correlations or betas) along the constituent links of the path. The second and third normal equations have similar interpretations in Figure 1. By convention, curved arcs without arrows represent correlations and directed arcs represent causes. The causal interpretation of beta coefficients is plausible because betas are standardized *partial* regression coefficients; they represent the effect of a predictor variable on Y when all the other predictor variables are fixed. You can interpret β_1 as what happens to Y when *only* X_1 is systematically varied. In this sense, beta coefficients provide a statistical version of the control you get in an experiment in which X_2 and X_3 are fixed and X_1 is varied; in such an experiment, the effect on Y is attributable to X_1. (Alternatively, the effects might be due to an unmeasured or *latent* variable that is correlated with X_1; we will not consider this case here.)

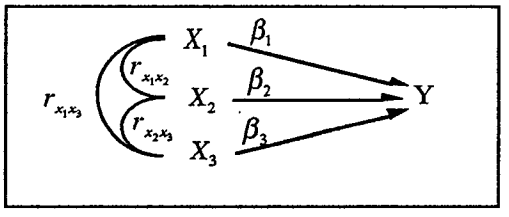

Figure 1: The path model that corresponds to the multiple linear regression of Y on X_1, X_2 and X_3.

So far we have described how a prediction model $\hat{Y} = \beta_1X_1 + \beta_2X_2 + \beta_3X_3$ gives rise to a set of normal equations, and to beta coefficients that make the prediction model a least-squares fit to our data. If this were all we could do, path analysis would be identical to linear regression and not worth the effort. The power of path analysis is that we can specify virtually any prediction model we like, and then solve for beta coefficients that ensure the model is a least-squares fit to our data.

3. PATH ANALYSIS OF PHOENIX DATA

Let us illustrate a path model other than the regression model with an example from the Phoenix system [Cohen & Hart, 93; Cohen et al., 89; Hart & Cohen, 92]. We ran Phoenix on 215 simulated forest fires and collected many measurements after each trial, including:

WindSpeed	The wind speed during the trial
RTK	The ratio of fireboss "thinking speed" to the rate at which fires burn
NumPlans	The number of plans tried before the fire is contained
FirstPlan	The name of the first plan tried
Fireline	The length of fireline dug by bulldozers
FinishTime	The amount of simulated time required to contain the fire

We specified a prediction model and the path model shown in Figure 2, and solved for the path coefficients on

each link. The coefficient on a path from X to Y is the correlation of X and Y if X is uncorrelated with any other node that points to Y; otherwise it is the beta coefficient from the regression of Y on all the correlated nodes that point directly to it (including X). With this rule it is easy to solve for path coefficients by hand, using the correlation matrix and running multiple regressions as necessary.

The estimated correlation between two nodes is the sum of the weights of all legal paths between them. A legal path goes through no node twice, and, for each node on the path, once a node has been entered by an arrowhead, it cannot be left by an arrowhead. The weight of a path is the product of the coefficients along it. For example, the estimated correlation between *FirstPlan* and *FinishTime* is the sum of three paths

$$\hat{r}_{FirstPlan,\ FinishTime} = (-.432 \times .287) + (.219 \times .506) + (-.432 \times .843 \times .506) = -.19$$

As it happens, this estimated correlation is very close to the actual empirical correlation, calculated from our data.

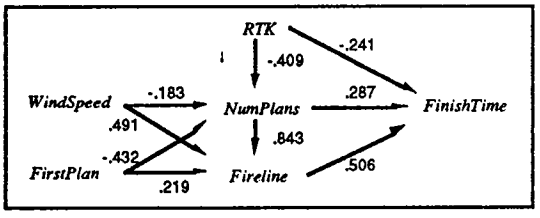

Figure 2: The result of a path analysis of Phoenix data

The disparities or *errors* between estimated and actual correlations of *FinishTime* with all the other factors are shown in Table 1. With the exception of the disparity between the correlations of *WindSpeed* and *FinishTime*, the estimated and actual correlations accord well, suggesting that Figure 2 is a good model of our data.

	WS	*RTK*	*#Pln*	*First*	*Fline*
$\hat{r}_{Factor\ FinishTime}$	.118	-.488	.704	-.197	.765
$r_{Factor\ FinishTime}$	-.053	-.484	.718	-.193	.755

Table 1: Errors in estimated correlations. The first row is the estimated correlation of *FinishTime* and the factor listed in each column; the second row is the actual, empirical correlation.

4. AUTOMATIC GENERATION OF PATH MODELS

The question that motivated the current research is whether models like the one in Figure 2 can be generated automatically by a heuristic search algorithm. This section describes such an algorithm.

The search space for the algorithm is the set of all possible path models. We can represent any path model as an adjacency matrix of size N x N, where N is the number of variables in the model. Thus the search space is of size 2^{N^2}.

The algorithm uses a form of best-first search. A single path model constitutes a state, while the sole operator is the addition of an arc to one variable in the model from another (possibly new) variable. We begin with a graph consisting of just the dependent variables. During the search we maintain a list of all the models and all possible modifications to those models. At each step in the search we select the best modification in the list, apply it to its target model, evaluate the new model, and add it to the list. We continue until either we have reached an acceptable model or we can make no more significant improvements. The most complex issue to be addressed in applying this algorithm is constructing the evaluation function.

We must first distinguish two senses in which a model can be "good." A common *statistical* measure of goodness is R^2, the percentage of variance in the dependent variable, Y, accounted for by the independent variables $X_1, X_2, \ldots$. If R^2 is low, then other variables besides $X_1, X_2, \ldots$ influence Y, and our understanding of Y is therefore incomplete. However, for any set of independent variables, no model accounts for more of the variance in Y than the regression model, in which all independent variables are correlated and all point directly to the dependent variable (e.g., Fig. 1). No wonder this model accounts for so much variance: every variable directly influences Y! It is not a parsimonious model. Nor is it likely to be a plausible causal model of any system we analyze. For example, a regression analysis of the Phoenix data treats *WindSpeed* and *Fireline* as causes at the same "level," that is, both pointing directly to *FinishTime*, but we know that *WindSpeed* is "causally upstream" of *Fireline*. In fact, we know that wind speed influences the rate at which fires burn, and, so, probably influences the amount of fireline that is cut. So R^2, the statistical measure of goodness, is not necessarily the researcher's *pragmatic* measure of goodness. The researcher wants a model that is a plausible causal story, and that includes as few correlational and causal relations among variables as possible, so causal influences are localized, not dissipated through a network of correlations. Such a model will necessarily have a lower R^2 than a highly-connected model, but will often be preferred. In fact, when we constructed the Phoenix model by hand, an important measure of goodness was the errors

between estimated and empirical correlations, and we never even calculated R^2.

Another distinction is between modification evaluation and model evaluation. Assuming that good models are clustered together in model space, a few heuristics can move the search into that general region of the space. These are the modification heuristics. A model evaluation function should dominate the search once the modification evaluation heuristics have brought the search into the right neighborhood. This is achieved by including the model evaluation function as a term in the modification evaluation function.

We rely on a variety of heuristics to guide search toward good models and to evaluate a path model. These comprise three general classes. The first contains what we might call syntactic criteria:

- no cycles are allowed;
- there must be a legal path (as defined in Section 3) from every variable to a dependent variable;
- a dependent variable may have no outgoing arcs.

The second class contains domain independent heuristics. These apply to any path analysis problem; however, the weighting of the heuristic may depend on the particular problem. Some heuristics we have tried are

- R^2, the statistical measure of goodness;
- the predicted correlation error (minimize the total squared error between the actual correlation matrix and the predicted correlation matrix for the model);
- parsimony (i.e. the ratio of variables to arcs, or the number of arcs that don't introduce a new variable);
- the correlation between the variables being connected by a new link;
- the total number of variables and arcs.

The third class represents domain/problem dependent forms of knowledge. These include knowledge that

- particular variables are independent;
- particular variables are likely/unlikely causes of others;
- a particular range of values for a domain independent heuristic is appropriate for the problem.

Our evaluation of a modification is a function of these heuristics. In the current implementation we use a weighted sum of the actual correlation between the variables to be linked, the predicted correlation error and the evaluation of the resulting model.

The modification evaluation step hides a good deal of computation required for the heuristic evaluation functions, including path coefficients in the model and the correlation estimates between variables. In the basic algorithm we calculate these parameters from scratch for each newly generated model. Because these calculations dominate search cost, we are currently working on improving the algorithm by incrementally calculating changes to each model's evaluation. This should greatly increase efficiency.

5. EXPERIMENTS

We have run several experiments to demonstrate the performance of our algorithm and probe factors that affect performance. In the first experiment we tested whether the algorithm could generate a model from the Phoenix data. The second experiment was more exhaustive; for this we used artificial data.

5.1. EXPERIMENT 1

We provided the algorithm with data from 215 trials of the Phoenix planner, specifically, *Windspeed RTK NumPlans Fireline* and *FinishTime* (we dropped *FirstPlan* from the data set). We designated *FinishTime* the dependent variable. The search space of models was large, $2^{N^2-N} = 1,048,576$, but the model-scoring and modification-scoring functions were sufficiently powerful to limit the solution space to 6856 models. When the algorithm terminated, after four hours work on a Texas Instruments Explorer II+ Lisp Machine, its best two models were the ones shown in Figure 3. These models have much to commend them, but they are also flawed in some respects. A good aspect of the models is that they get the causal order of *WindSpeed*, *NumPlans*, and *Fireline* right: wind speed is causally upstream of the other factors, which are measures of behavior of the Phoenix planner. However, both models say *WindSpeed* causes *RTK* when, in fact, these are independent variables set by the experimenter. (Later, probing this curiosity we realized that due to a sampling bias in our experiment, *WindSpeed* and *RTK* covary, so connecting them is not absurd.)

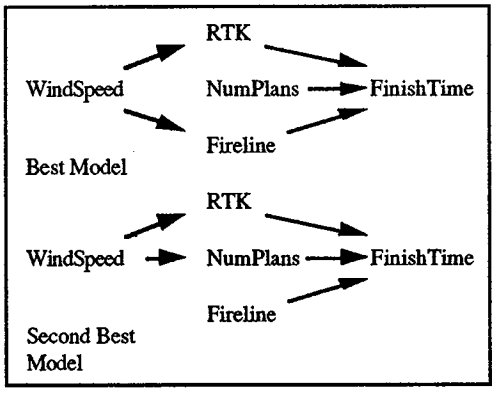

Figure 3: The best and second best Phoenix models found by the algorithm.

A disappointing aspect of the models is that neither recognizes the important influence of *RTK* on *NumPlans* and the influence of *NumPlans* on *Fireline*. In defense

of the algorithm, however, we note that it used model scoring and modification scoring functions that valued R^2, while we, when we built Figure 2 by hand, were concerned primarily about the errors between estimated and empirical correlations, and not concerned at all about R^2. Thus we should not be too surprised that the algorithm did not reproduce our model in Figure 2.

Although the algorithm ran slowly, and is in some ways disappointing, it did produce what we believe is the first causal model of a complex software system ever generated automatically.

5.2. EXPERIMENT 2

The first experiment raised more questions than it answered: Does the performance of the algorithm depend on the sample variance for each variable? Does performance depend on the number of data for each variable? How do we know whether the algorithm is finding the "right" model? To address these and other questions we ran an experiment in which we constructed path models that represented "truth" and tested how frequently our algorithm could discover the true models. Specifically, we followed these steps:

1. Randomly fill some cells in an adjacency matrix to produce a path model.
2. Randomly assign weights to the links in the path model subject to the constraint that the R^2 of the resulting model should exceed .9.
3. Generate data consistent with the weights in step 2.
4. Submit the data to our algorithm and record the models it proposed.
5. Determine how well our algorithm discovered the true model.

Steps 3 and 5 require some explanation. Imagine someone asks you to generate two samples of numbers drawn from a normal (Gaussian) distribution with mean zero such that the correlation of the samples is a particular, specified value, say, .8. Now make the problem more difficult: generate N columns of numbers so that all their pairwise correlations have particular, specified values. Solving this problem (step 3, above) ensures that we generate data with the same correlational structure as the path model specified in step 2. The details of the process are sketched in the Appendix.

We evaluated the performance of our algorithm (step 5, above) by two criteria:

Shared path score: For each true model, make a list of all the paths from each independent variable to the dependent variable; what fraction of these paths exist in the model discovered by our algorithm?

Shared link score: For each true model, make a list of all the links between variables; what fraction of these links exist in the model discovered by our algorithm?

Our experiment was designed to test the effects of sample size on the performance of the algorithm. The Phoenix data included 215 values of each variable, but for this experiment we looked at four levels of the factor: 10, 30, 50, and 100 data per sample. We thought that the performance of the algorithm might deteriorate when sample sizes became small, because the effects of outlier values in the samples would be exacerbated. We generated five true path models with four variables, shown in Figure 4. For each model we generated 12 variants, that is, 12 sets of data for variables A,B,C and D that conformed to the correlation structure established in step 2, above. This produced 60 models. We ran our algorithm on these variants in each of the 4 conditions just described. Thus, the algorithm ran 240 times.

The algorithm found the true model in 109 of the 240 trials. Summary statistics for these trials, and trials in which the algorithm did not find the true model, are presented in Table 2. When the algorithm did find the true model, it explored 174.6 models, on average. This is about four percent of the 4096 models that were possible. Trials that did not find the true model explored only 86.96 models, which suggests that one reason they failed to find the true model is they gave up too soon. This in turn suggests that our criteria for terminating a search can be improved.

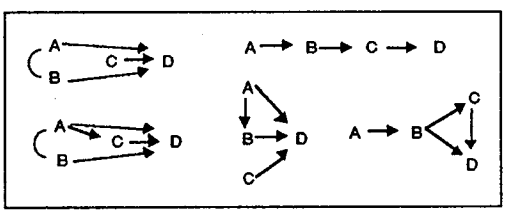

Figure 4: Five path models used to test the algorithm.

In all trials, the average score of the best model found by the algorithm was better than the average score of the true model. Our immediate inclination was to say the scoring criteria led the algorithm astray, away from the true model. In fact, the problem arises from using R^2 as our model evaluation function. The value of R^2 monotonically increases as the number of links in the model increases. Consequently, in most of our trials, the model from which we generated the data is not the model with the highest score since the addition of new links to it produced a model with a higher R^2. We must investigate other model evaluation metrics before we can evaluate whether or not our algorithm will find the best model.

We would like to place more stock in two measures of structural similarity between the true model and the best found model. The shared path score is the proportion of paths in the best found model that also exist in the true model. For example, the top-left model in Figure 4 has 5 paths from A, B and C to D, so if the best found model

was, say, the top-right model, which has only three paths from A, B and C to D, then the shared path score would be 0.6. Another criterion is the shared link score, which is the fraction of the links connecting variables in the true model that are also in the best found model. Unfortunately, neither score strongly differentiates trials in which the true model was found from those in which it wasn't. On average, 56-61% of the paths, and 50-54% of the links, in the true model are also in the best found model. These numbers are not high. On the other hand, if the disparity between the true model and the best found model was just two links, then the average shared link score would be 52%. So although the shared path and shared link scores are low, they suggest that the best found model differed from the true one by little less than two links on average. Clearly, the algorithm can be improved, but it performs better than the first impression given by the shared path score and shared link score.

	Trials that did find the true model		Trials that did not find the true model	
	Mean	Std.	Mean	Std.
Number of models	174.600	93.300	86.962	72.770
Best model score	.974	.027	.954	.018
True model score	.847	.258	.708	.417
Shared path score	.612	.202	.565	.133
Shared link score	.542	.193	.501	.144

Table 2. Summary statistics from 240 trials.

We ran a one-way analysis of variance to find the effects of the number of data points in our samples. The dependent variables were the five measures in Table 2. Happily, sample size appears to have no impact on the number of models expanded by the algorithm, the score of the best found model, or the shared path or shared link scores.

6. RELATED ALGORITHMS

6.1 INDUCTIVE CAUSATION ALGORITHM

The underlying probability structure of a set V of random variables in a system can be described by a joint probability distribution P. Pearl and Verma [1991] have developed a method for inducing causal structure from $\hat{P}$, a sampled distribution of V, which they refer to as the Inductive Causation Algorithm (IC algorithm). Causal models are represented by directed acyclic graphs (DAG) whose nodes represent random variables and whose edges represent causal relationships between those variables. DAG construction is based on the notion of conditional independence under the assumptions that $\hat{P}$ is stable and model minimality holds true. $\hat{P}$ is considered to be stable if it has a unique causal model. Minimality means it is reasonable to rule out any model for which there exists a simpler model that is equally consistent with the data.. Two variables x and y are considered independent if a change in either of them does not cause a change in the other. Conditional independence can be stated in mathematical terms as $P(x|y,z) = P(x|z)$. Intuitively this means that given some knowledge about the value of x given z, discovering knowledge about y does not effect what we know about x. In this case, x and y are conditionally independent on z. This condition is also known as the Markov condition applied to graphs: for all graph vertices x, x is independent of its remote ancestors given its parents.

To find D, a causal model, the algorithm begins with $\hat{P}$, by looking for a set $S_{xy} \subseteq V$ for all pairs of variables x and y such that x and y are conditionally independent on z, $\forall z \in S_{xy}$. If no such set can be found and x and y are dependent in every context, then an undirected link is placed between x and y to represent that there is a *potential causal influence* between x and y.

In the second step, for all pairs of unconnected variables having undirected links to a common variable z, if $z \in S_{xy}$ then do nothing. Otherwise, add arrowheads pointing at z. The reasoning behind this step is that if a and b are not independent conditional on z, but are correlated to z, then this correlation must be due to their common effect on z or through a common cause.

Finish building D by recursively adding arrowheads by applying the following rules:

1. If x and y are joined by an undirected link and there exists some directed path from x to y, then add an arrowhead at y. Since x is known to effect y through x's effect on some other variable, then the direction of potential causal influence between x and y must be toward y (this prevents cycles).
2. If x and y are not connected but there exists a variable z such that there is a direct effect of x on z and there is an undirected link from z to y (i.e. $x \rightarrow z - y$), then add an arrowhead at y. Nodes x and y are not connected because they were found to be conditionally independent on some S_{xy}. If $z \notin S_{xy}$, then arrowheads would have been placed towards z from both x and y in step two for the reason stated above. Therefore, x and y must be conditionally independent on z and since we know x has a causal influence on z, the direction of potential causal influence must be from z to y in order for the conditional independency to hold.

Finally, if there is an unidirected or bidirected link from x to y, then mark all unidirected links from y to z in which z is not connected to x.

The causal model resulting from analysis of a data set by the IC algorithm will contain four types of links representing different relationships between the variables they connect; unmarked unidirectional (potential causal influence), marked causal influence (direct causal influence), bidirectional (evidence of a common cause), and undirected (undetermined relationship). The model is used as a template for developing a causal theory which is consistent with the data. That theory is built upon the following theorems:

If every link of a directed path from x to z is marked then x has a causal influence on z.

A marked unidirectional link from x to y suggests a direct causal influence of x on y.

The existence of a bidirectional link between x and y, suggests a common cause z affecting both x and y, and there is no direct causal influence between them.

Pearl and Verma have shown that particular patterns of dependencies dictate a causal relationship between variables and that the IC algorithm can be used to recover the structure of those causal relationships. However there are situations for which the method returns poor or inconclusive results. With respect to the latter, if there are no independencies between the variables in $\hat{P}$, all of the variables will be interconnected with undirected links since it may be impossible to determine the expected nature of the potential causal influence. It may also be impossible to determine the nature of a potential causal influence between two variables when there are no *control variables* acting on either of the variables in question. If the direction of potential causal influence between variables x and y is uncertain but there exists a variable z that is known to affect x and not y, then we can eliminate the possiblity of x causing y. z is known as a control variable.

6.2 TETRAD II

Spirtes et al. [1993] take a different approach to discovering causal structure, based on constraints on correlations. Their approach is implemented in the Tetrad II program. During search, constraints implied by a model are compared to constraints found in the data, and the program returns a model that best reflects constraints in the data. To describe this method it will be necessary to give a few definitions:

An *open path* is a directed link from variable x (the source) to variable y (the sink).

A *trek* is a pair of open paths between two variables x and y, having the same source and intersecting only at that source. One of the paths can be the empty path which consists of one vertex (the source of the trek).

Partial equations represent correlational constraints on three variables. Let x, y, and z be three measured variables in a causal model, then $r_{xz} = r_{xy} \cdot r_{yz}$ is their partial equation. This is equivalent to saying the partial correlation between x and z (with y held constant) is equal to zero (known as a vanishing partial). For a linear model, vanishing partials are equivalent to conditional independencies.

Tetrad equations represent correlational constraints on four variables. If x,y,z, and v are four measured variables in a causal model, one of the following is their tetrad equation:

$$r_{xy} \cdot r_{zv} = r_{xz} \cdot r_{yv}, \text{ or, } r_{xy} \cdot r_{zv} - r_{xz} \cdot r_{yv} = 0$$
$$r_{xy} \cdot r_{zv} = r_{xv} \cdot r_{yz}, \text{ or, } r_{xy} \cdot r_{zv} - r_{xv} \cdot r_{yz} = 0$$
$$r_{xz} \cdot r_{yv} = r_{xv} \cdot r_{yz}, \text{ or, } r_{xz} \cdot r_{yv} - r_{xv} \cdot r_{yz} = 0$$

where r_{ij} is the sample correlation between variables i and j. These constraints are known as *vanishing tetrad differences*. A DAG, G, linearly implies $r_{xy} \cdot r_{zv} = r_{xz} \cdot r_{yv} = 0$ iff either r_{xy} or $r_{zv} = 0$, and r_{xz} or $r_{yv} = 0$, or there is a set O of random variables in G such that $r_{xy.o} = r_{zv.o} = r_{xz.o} = r_{yv.o} = 0$.

Associated probability, $P(t)$, of a vanishing tetrad difference is the probability of getting a tetrad difference equal to or larger than that found for the sample, under the assumption that the difference vanishes in the population. $P(t)$ is determined by lookup in a chart for the standard normal distribution.

T-maxscore is the sum of all $P(t)$ implied by the model and judged to hold in the population.

Tetrad score is the difference between the sum of all $P(t)$ implied by the model and judged to hold in the population, and the weighted sum of $(1-P(t))$ for all $P(t)$ implied by the model that are not judged to hold in the population.

The input to TetradII is a sample size, a correlation or covariance matrix, and a DAG, representing an initial causal structure, having no edges between measured variables and all having all latent variables connected. The search generates all possible models derived by addition of one edge to the initial model and each model is ranked by its tetrad score, which basically rewards a model for implying constraints which are judged to hold in the population and penalizes the model for implying constraints which are not judged to hold in the population. For example, consider the DAG's of Figure 5. 5a implies the following vanishing tetrad differences:

$$r_{xy} \cdot r_{zw} - r_{xz} \cdot r_{yw} = 0$$
$$r_{xy} \cdot r_{zw} - r_{xw} \cdot r_{yz} = 0$$
$$r_{xz} \cdot r_{yw} - r_{xw} \cdot r_{yz} = 0$$

If the population distribution is represented by 5b, then the only vanishing tetrad difference implied by 5a that exists in the population is the second one.

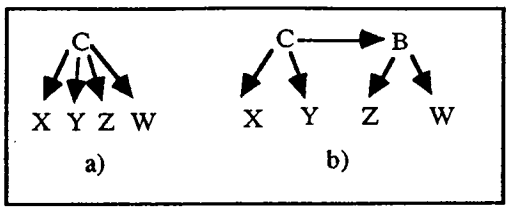

Figure 5: DAGs implying different sets of vanishing tetrad differences.

This process is repeated recursively on each model generated until no improvements can be made to any model. Search is guided by the t-maxscore since models are expanded according to their t-maxscore. If M' is the model generated by adding an edge to model M and M''s t-maxscore is less than the tetrad score of M, it is eliminated from the search. A proof of the following theorem is provided by Spirtes (1993):

> If M' is a subgraph of directed acyclic graph M, then the set of tetrad equations among variables of M' that are linearly implied by M is a subset of those linearly implied by G.

This theorem intuitively says that the addition of links to a graph will never cause more tetrad equations to be implied by the resulting graph. It then follows that if $t - \max score_{M'} < tetrad - score_M$, neither M' or any modification of M' will have a tetrad score as high as M. The algorithm essentially uses a form of best first search and in cases where the search is too slow to be practical, the depth of search is limited to reduce the amount of time spent on search. The program returns a list of suggested additions of edges to the initial model which will yield the model found by the search to have the highest tetrad score.

7. CONCLUSION

We have described an algorithm for path analysis and first results from experiments with the algorithm. In a crude sense the algorithm "works," that is, it generates causal models from data. However, preliminary results suggest that other methods of modification and model evaluation as well as algorithm evaluation need to be explored. We believe our use of the difference between estimated and empirical correlations, and of R^2, as terms in our modification evaluation function serve to bring the search into the neighborhood of good model space, but it is obvious that R^2 applied as a model evaluation function is less than ideal because it drives the algorithm to generate regression models (i.e., "flat" models in which all the predictor variables point directly at the dependent variable and are intercorrelated). Followup experiments to those reported earlier confirm this: As R^2 is given more weight in the modification evaluation and model evaluation, we see more regression models. Clearly our measures of structural similarity, shared link score and shared path score, also can be improved. These problem must be addressed before we can establish that the algorithm works well.

At this early stage, we have not made any empirical comparisons between our algorithm and those of Pearl and Verma, and Spirtes and his colleagues. We believe that unlike Pearl's algorithm, a lack of independencies will not negatively effect the results obtained by our algorithm; and unlike TetradII, ours does not require an initial model. However, the complexity of our algorithm is significantly greater than that of the others. Moreover, it performs more computation for each model because it estimates edge weights and predicted correlations. We are not convinced that this is a negative aspect of our algorithm since we believe estimated correlations, and specifically the deviations between estimated and actual correlations, are important criteria in controlling search in our algorithm.

Our algorithm would probably be helped a lot if the modification and model evaluation functions were "monotonic" in a sense similar to TetradII, which eliminated models under particular conditions that guaranteed that no descendants of the eliminated models could be preferred to previous models. We need modification and model evaluation functions that will allow us to prune subtrees of the space of models. For example, it is easy to compute whether a modification will increase or decrease the difference between estimated and actual correlations; we might prune modifications that don't reduce disparities.

Causality plays an important role in human understanding. As researchers, we feel it is not enough to be able to predict the behavior of computer systems; we must strive to discover rules that dictate their behavior. We believe behavior results from the interaction of a system's architecture, environment, and task; all these components must be present in a causal model. These models are essential to establishing general theories concerning the types of behaviors that might occur in certain types of tasks, architectures, and environments. We have yet to answer many questions about factors that affect the performance of our algorithm. Still, we are encouraged by our results, and by the promise the algorithm holds for automatically finding causal models of systems. However ponderous it is, however difficult it is to evaluate, the fact remains that the algorithm generated causal models of a complex AI planning system, and promises to be a valuable tool for those who seek to understand AI systems with statistical methods.

Acknowledgments

We would like to thank Glenn Shafer for his help with the development of these ideas.

This research was supported by DARPA-AFOSR contract F30602-91-C-0076. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

Asher, H.B. 1983. *Causal Modeling.* Sage Publications, Newbury Park, CA.

Cohen, P.R. *Empirical Methods for Artificial Intelligence.* Forthcoming.

Cohen, P.R & Hart, D.M., 1993. Path analysis models of an autonomous agent in a complex environment. To appear in *Proceedings of The Fourth International Workshop on AI and Statistics.* Ft. Lauderdale, FL. 185–189.

Cohen, P.R., Greenberg, M.L., Hart, D.M. & Howe, A.E., 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine,* 10(3): 32-48.

Falkenhainer, B.C. & Michalski, R.S., 1986. Integrating quantitative and qualitative discovery: the ABACUS system. *Machine Learning* 1(1): 367-401.

Glymour, C., Scheines, R., Spirtes, P. & Kelly, K., 1987. *Discovering Causal Structure.* Academic Press.

Hart, D.M. & Cohen, P.R., 1992. Predicting and explaining success and task duration in the Phoenix planner. *Proceedings of the First International Conference on AI Planning Systems.* Morgan Kaufmann. 106-115.

Langley, P., Simon, H.A., Bradshaw, G.L. & Zytkow, J.M., 1987. *Scientific Discovery: Computational Explorations of the Creative Processes.* The MIT Press.

Li, C.C. 1975. *Path Analysis–A Primer.* Boxwood Press.

Pearl, J. & Verma, T.S., 1991. A theory of inferred causation. *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference,* J. Allen, R. Fikes, & E. Sandewall (Eds.). Morgan Kaufman. 441–452.

Pearl, J. & Wermuth, N., 1993. When can association graphs admit a causal interpretation? *Preliminary Papers of the Fourth International Workshop on AI and Statistics.* Ft. Lauderdale, FL. 141–150.

Schaffer, C., 1990. A proven domain-independent scientific function-finding algorithm. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* The MIT Press. 828–833.

Spirtes, P., Glymour, C. and Scheines, R. (1993) *Causation, Prediction and Search.* Springer-Verlag.

Zytkow, J.M., Zhu, J. & Hussam, A., 1990. Automated discovery in a chemistry laboratory. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* The MIT Press. 889–894.