Automating Path Analysis for Building Causal Models from Data*

Paul R. Cohen, Adam Carlson,
Lisa Ballesteros, Robert St. Amant

Computer Science Technical Report 93-38

Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

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 statistical methods such as 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.

* This research was supported by DARPA-AFOSR contract F30602-91-C-0076.

1. INTRODUCTION

Because machine learning techniques search for structure in data, they often closely parallel statistical techniques for exploratory data analysis, such as clustering, factor analysis and regression. 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 around the analysis of structural equations—essentially path analysis. Pearl [91; 93] and Spirtes [93] have recently developed a causal induction algorithm with a more general mathematical basis than path analysis; it relies on evidence of nonindependence, a weaker criterion than path analysis, which relies on evidence of correlation.

We developed the 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 Section 3, where we introduce path analysis, or Section 4 where we illustrate a path analysis of Phoenix. Section 5 describes our algorithm. Section 6 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.

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 interpretation of such a rule is that a unit change in a predictor variable x produces b units change in $\hat{y}$. Thus, the regression coefficients depend on the units of measurement of the predictor variables; for example, if x is measured in feet then b will be a factor of 5280 larger than if x is measured in miles. For this and other reasons, regression models are often constructed for standardized variables. A standardized variable is what you get by subtracting from the variable its mean and dividing by its standard deviation. For concreteness and simplicity, we will show how to solve for coefficients in a standardized linear regression model with three predictor variables. Here is the prediction model: $\hat{Y} = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$ (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.

Multiple linear regression finds beta coefficients to construct a least-squares rule. First, multiply the prediction model through by each of the terms on the right hand side, producing three equations:

$$\hat{Y}X_1 = \beta_1 X_1^2 + \beta_2 X_2 X_1 + \beta_3 X_3 X_1$$
$$\hat{Y}X_2 = \beta_1 X_1 X_2 + \beta_2 X_2^2 + \beta_3 X_3 X_2$$
$$\hat{Y}X_3 = \beta_1 X_1 X_3 + \beta_2 X_2 X_3 + \beta_3 X_3^2$$

then take sums:

[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.

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

Figure 2: A path model that does not correspond to a multiple linear regression of Y on X_1, X_2 and X_3.

X_1, X_2 and X_3 exist. The corresponding prediction model is $\hat{Y} = \rho_{1Y}X_1 + \rho_{2Y}X_2 + \rho_{3Y}X_3$. The arcs are labeled with path coefficients, denoted ρ. Path analysis tells us whether these coefficients are correlations or partials, and if the latter, path analysis solves for the values that make the prediction model a least-squares fit to the data.

What follows is an informal description of the steps in path analysis. See [Cohen] for a more formal discussion of why these steps yield path coefficients that ensure that the prediction model is a least-squares fit to the data. The first step is to propose a prediction model and a corresponding path diagram. Imagine Figure 2 is the result. The second step is to lag each arc as a correlation or a path coefficient. If several variables $A, B, \ldots$ all point to another variable K (as X_1, X_2, X_3 point to Y in Fig. 2) then

2. PATH ANALYSIS

The power of path analysis is that we can specify virtually any prediction model we like, and then solve for path coefficients that ensure the model is a least-squares fit to our data. For example, Figure 2 shows a model in which X_1 and X_2 directly cause Y, and X_2 also indirectly influences Y through X_3, and no intercorrelations between X_1, X_2 and X_3.

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

Clearly, with these three equations we can solve for the three unknown path coefficients. We have not shown that these coefficients guarantee that $\hat{Y} = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$ is a least-squares rule, but the interested reader can find this demonstration in [Li, 52] or any good statistics text.

$$\begin{aligned} r_{1Y} &= \beta_1 + \beta_2 r_{12} + \beta_3 r_{13} \\ r_{2Y} &= \beta_1 r_{12} + \beta_2 + \beta_3 r_{23} \\ r_{3Y} &= \beta_1 r_{13} + \beta_2 r_{23} + \beta_3 \end{aligned} \quad (1)$$

The three equations (1) are called the normal equations, and they have an interesting interpretation, illustrated in Figure 1. Consider the first normal equation, $r_{1Y} = \beta_1 + \beta_2 r_{12} + \beta_3 r_{13}$. 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_{1Y} is implied 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 labeled 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. Y 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.)

So far we have described how a prediction model and a set of data give rise to $\hat{Y} = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$, and how to find coefficients that make the prediction a least-squares fit to our data. We also say a model is an interpretation of the normal equations in terms of causal interpretation of the path diagram in Figure 1. If this were all we could do, both analysis would be identical to linear regression and not worth the effort.

- If $A, B, \ldots$ are independent causes of K, then the path coefficients $p_{KA}, p_{KB}, \ldots$ are just the correlation coefficients $r_{AK}, r_{BK}, \ldots$, respectively.
- If $A, B, \ldots$ are dependent causes of K, then $p_{KA}, p_{KB}, \ldots$ are the standardized partial regression coefficients $\beta_{AK}, \beta_{BK}, \ldots$, respectively, obtained from the multiple regression of K on $A, B, \ldots$

By these rules, $p_{YX_1} = r_{X_1Y}$ because X_1 is independent of all other causes of Y; $p_{YX_2} = \beta_{X_2Y}$ and $p_{YX_3} = \beta_{X_3Y}$ because X_2 and X_3 are dependent causes of Y; and $p_{X_3X_2} = r_{X_2X_3}$ because X_2 is independent of all other causes of X_3.

The next step is to solve for the path coefficients. Because p_{YX_1} and $p_{X_3X_2}$ are correlations, they are calculated directly from our data. To find the other path coefficients we first run a multiple regression of Y on X_2 and X_3, or, equivalently, solve two normal equations, $r_{YX_2} = \beta_{YX_2} + \beta_{YX_3} r_{X_2X_3}$ and $r_{YX_3} = \beta_{YX_2} r_{X_2X_3} + \beta_{YX_3}$, for β_{YX_2} and β_{YX_3}.

Finally we can estimate the correlations between X_1, X_2, X_3 and Y. This involves finding paths between the X variables and Y, and summing the weights of the paths. In 1929, Sewall Wright [Asher, 83; Li, 75;] specified three rules for legal paths. We found that it was possible to reduce Wright's rules to a more compact form by introducing the idea of entering or leaving a node by an arrowhead. In the following rules, entering a node by an arrowhead means to follow a forward arrow or a correlation link to the node. Leaving a node by an arrowhead means to follow a backward arrow or correlation link from a node. These rules for are:

1) A path cannot go through a node twice
2) Once a node has been entered by an arrowhead, no node can be left by an arrowhead

This is significantly easier to implement than Wright's rules.

The weight of a path comprising multiple links is the product of the coefficients along the path. By these rules, the estimated correlation $\hat{r}_{X_1Y}$ is just the path coefficient p_{YX_1}, which is the empirical correlation r_{X_1Y}. However, $\hat{r}_{X_2Y} = \beta_{YX_2} + r_{X_2X_3}\beta_{YX_3}$ because there are two paths between X_2 and Y, the first direct and the second indirect through X_3. Similarly, $\hat{r}_{X_3Y} = \beta_{YX_3} + r_{X_2X_3}\beta_{YX_2}$ because there are two paths between X_3 and Y, one going backward through X_2.

4. PATH ANALYSIS OF PHOENIX DATA

Let us illustrate these steps 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 3, and solved for the path coefficients on each link. You can see that the estimated correlation between, say, *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.

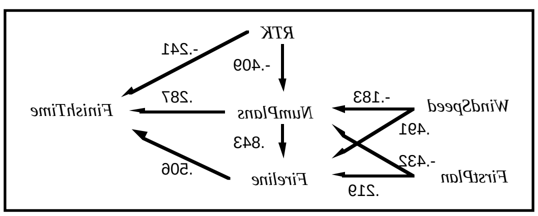

Figure 3: The result of a path analysis of Phoenix data

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 3 is a good model of our data.

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.

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

between estimated and empirical correlations, and we never even calculated R_S^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. We can group heuristics into three general classes. The first class contains what we might call syntactic criteria:

- no cycles are allowed;
- there must be a path (in the sense of Wright's rules) 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_S^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 which don't introduce a new variable):
- the correlation between the variables being connected by a new link;
- the total number of variables and arcs.

Others which we have considered, but not yet implemented are

- the "attenuation" of the variables and arcs in the model (the "stretchiness" of the model):
- bandwidth:
- Various adjustments to R_S^2.

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.

2. AUTOMATIC GENERATION OF PATH MODELS

The question that motivated the current research is whether models like the one in Figure 3 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. Path models are graphs which satisfy the constraints described in section 3. We can represent any path model then 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 possible modifications to those models. After 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_S^2, the percentage of variance in the dependent variable, Y, accounted for by the independent variables $X_1, X_2, \ldots$. If R_S^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 model this model accounts for so much variance: every variable directly influences Y.[1] 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 Firaline as causes at the same "level," that is, both pointing directly to FinishTime, but we know that WindSpeed is "causally upstream" of Firaline. In fact, we know that wind speed influences the rate at which fires burn, and, so, dropsply influences the amount of fireline that is cut. So R_S^2, the statistical measure of goodness, is not necessarily the researcher's primary 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_S^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 extent

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.

6. 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.

6.1. 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 4. 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: the wind speed is causally upstream of the other factors, which are measures of behavior of the Phoenix planner. A bad aspect of the models is that both say *WindSpeed* causes *RTK* when, in fact, these are independent variables set by the experimenter. However, probing this curiosity we realized that due to a sampling bias in our experiment, *WindSpeed* and *RTK* covary, so connecting them is not absurd. 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 the algorithm used model scoring and modification scoring functions that valued R^2, while we, when we built Figure 3 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 3.

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.

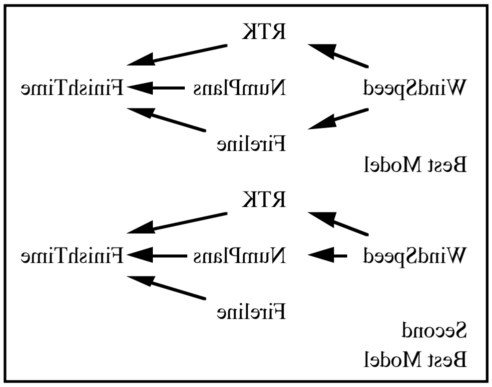

Figure 4: The best and second best Phoenix models found by the algorithm.

6.2. 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 5. 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,

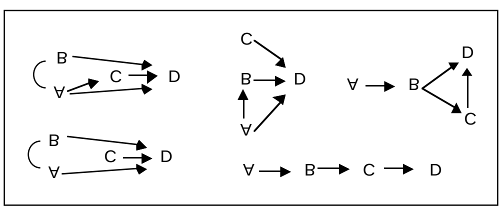

Figure 5: Five path models used to test the algorithm.

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.

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. However, the disparity between the best model score and the true model score was less in the trials that found the true model, which casts doubt on this explanation. The reason for the disparity is actually quite subtle: when we generate data to match the correlation structure of each path model (as described in the Appendix), we do, in fact, generate data that have the expected correlations, but if we fit the path model to those data, the resulting R^2 is lower than expected. Said differently, after assigning link weights to the true model (step 2, above), the model R^2 might be, say, .95, but after we generate data (step 3) and fit the data to the model, we find that the model only accounts for, say, 80% of the variance in the data. This is an artifact of our procedure for generating data. Consequently, in most of our trials, the model from which we generated the data is not the model with the highest score. This problem will have to be solved before we can claim that our algorithm found or failed to find the "best" model.

Table 2. Summary statistics from 240 trials.

	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

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 5 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.

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

We have yet to answer many questions about factors that affect the performance of the algorithm. For example, we ran an experiment to see whether the variance of our assumption data affects performance, but the results were inconclusive. We also need to know how the solution space increases with the number of variables. With four variables, the algorithm searched roughly 100 models on average, but with five variables it searched thousands of models. Clearly, we would prefer this curve to rise less steeply.

Still, we are encouraged by our results, and by the promise the algorithm holds for automatically finding causal models of systems. However, boldness it is, powerful 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.

APPENDIX: DATA GENERATION

Every path model has associated with it a predicted correlation matrix, calculated from its path weights, which describes the relationships among its variables. For the purpose of data generation, the path weights of a given model are randomly generated under the following constraints:

- all predicted correlations for the model must be less than one
- $0.1 \geq \beta_S > \rho$.
- $.2 \geq$ correlational path weights > 1
- $.5 \geq$ causal path weights > 1

After initialization of the path weights, the predicted correlation matrix is calculated. It is essential that data generated for a model have the same correlational structure as the model. The technique for the generation of variates from a multivariate normal distribution was chosen for its efficiency and because it returns a variate having the correlational structure multiplying the variance-covariance matrix used to calculate that variate.

A p-dimensional random vector $\bar{x}$ is defined to have a multivariate normal distribution if and only if every non-trivial linear combination of the p components of $\bar{x}$ has a univariate normal distribution. The p-variate normal distribution having mean $\bar{\mu}$ and variance-covariance matrix V (nonsingular) is defined by the pdf

$$f(x) = (2\pi)^{\frac{-p}{2}}|V|^{\frac{-1}{2}}\exp\left\{-.5(\bar{x}-\bar{\mu})'V^{-1}(\bar{x}-\bar{\mu})\right\} \quad (1)$$

The distribution of $\bar{x}$ can be represented as a linear transformation of p independent normal variates in $\bar{z}$, such that

$$\bar{x} = A\bar{z} + \bar{\mu} \quad (2)$$

The method for generating $\bar{x}$ from this distribution involves generating $z_1, z_2, \ldots, z_p$ independent normal variates and forming $\bar{z}$ from them. Since each z_p is sampled

scores. But the sample size has a large significant effect on the score of the true model. This effect turned out to be a statistical artifact, due entirely to the sensitivity to sample size of the method for generating data. Thus, it appears that the algorithm can work even with relatively few data.

5. CONCLUSION

We have described an algorithm for both analysis and first results from experiments with the algorithm. In a crude sense the algorithm "works," that is, it generates causal models from data. It works slowly, however, and we have yet to establish criteria for demonstrating clearly that it works well. The algorithm's speed can certainly be improved: for example, we keep all models that have been generated, which takes a lot of space, so begins considerably to the run time. Still, the complexity of the search space is exponential in the square of the number of variables. It remains to be seen whether we can improve the algorithm enough to build large causal models in a reasonable time.

More important in the near term is whether the algorithm produces good models. As we noted earlier, goodness can be evaluated by both statistical and pragmatic criteria. The algorithm is currently parameterized, which enables us to adjust the relative importance of the criteria. As you might expect, if β_S is weighted heavily and parsimony is devalued, then the algorithm will always generate regression models, in which all variables point directly to the dependent variable and are fully intercorrelated. If parsimony is also weighted heavily, the algorithm tends to keep the flat causal structure of regression models, but drops the intercorrelations among the predictor variables. If errors between expected and empirical correlations and parsimony are weighted heavily, then the algorithm will produce causal models with nodes between the predictor and dependent variables, such as the models in Figure 4. All of these kinds of models are good by some criteria.

If by good we mean that the algorithm finds the "correct" model—the model that corresponds to the system that generated the data—then evaluation of goodness is technically challenging. Our procedure was to first decide upon a path model, then generate data consistent with the correlation structure of the model, then see whether our algorithm could find the original model. While the logic of the experiment is sound, two technical problems arose. First, it often happened that we generated data that were better explained by a model other than the original model. In this case, the "correct" model, that is, the one from which we generated the data, might be given a low score by the statistical component (i.e., β_S) of the model scoring function. Second, in the event that the algorithm does not find exactly the "correct" model, but one very like it, how should we measure closeness? Our criteria, shared both and shared links, were not ideal. More needs to be done to solve both of these problems.

from the normal distribution with standard deviation 1. the variance-covariance matrix is equal to the correlation matrix. A is formed from a Cholesky decomposition of V and $\overline{\mu}$ is chosen to be a zero vector. These values are supplimted into equation (5) to calculate $\overline{x}$. The process is repeated n times, each vector $\overline{x}$ forming the columns of a $q \times n$ data matrix having n data points for each variable.

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.

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 thereon.

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. 182–191.

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): 35-48.

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., 1993. Predicting and explaining success and task duration in the Phoenix planner. Proceedings of the First International Conference on AI Planning Systems. Morgan Kaufmann. 106-112.

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 Kaufmann. 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.