Causal Performance Models of Computer Systems: Definition and Learning Algorithms
TECHNICAL REPORT

Jan Lemeire
ETRO Dept.
Vrije Universiteit Brussel, Belgium
jan.lemeire@vub.ac.be

Abstract

Causal models are proposed for the representation of relational information of a performance analysis of computer systems. Performance models fulfill many requirements. Causal models offer a formalization and unification of the properties that we expect. Causal structure learning algorithms attempt to construct such models from experimental data. Existing algorithms have been extended to incorporate the variety of variable types and relations encountered in real performance data. To handle a combination of continuous and discrete variables with possibly non-linear relations, we use the more general conditional independence test based on the mutual information between probabilistic variables. The underlying probability distribution of experimental data is estimated by kernel density estimation. The estimate is constructed by centering a scaled kernel at each observation. Deterministic relations imply that variables contain equivalent information about other variables and cannot be represented by a faithful model. To handle this, the complexity of the relations is used as a criterion to decide upon which of the equivalent variables directly relates to the target and conditional independency is redefined to reestablish the faithfulness of the graphs. Experiments with sequential and parallel programs show that accurate models are inferred. They provide insight in how each variable affects overall performance measures and the analysis can be used to validate independence assumptions and find potential explanations for outliers.

1. Introduction

The design and implementation of high-performance computer applications requires knowledge of many factors that influence performance. This task can be facilitated by tools that support the performance analysis and offer multifunctional performance models. The models should fulfill many requirements. They should provide information on the expected performance, offer insight in the causes of performance degradation, should be constituted of reusable submodels, tell which variables should be known for predicting others, should make it possible to estimate the effects of optimizations and enable to reason under uncertainty. Causal models offer an elegant formalization of these properties. Causality is widely used in social sciences such as economy or sociology, in biology, machine learning, ... A causal model represents the *independency* relations among variables. It reduces a model to independent submodels in which the directly related variables of a variable contain all information about that variable. Besides the encoding of the joint distribution and the explicit representation of dependency information, a causal model aims at representing the underlying physical mechanisms that generated the data. This enables a user to reason about modifications - called *interventions* - such as algorithm or system optimizations.

This paper reports on experiments with existing learning algorithms for finding the causal structure of performance variables. Experimental data were obtained from distributed computer systems. A causal model represents the conditional independence relations among the variables by a Directed Acyclic Graph (DAG). The correspondence of the conditional independencies in the graph and the data is called *faithfulness*. Causal structure learning algorithms try to construct a faithful graph based on the conditional independencies found in the experimental data. Real performance data are more complex than data typically encountered in the academic examples used for research about causal analysis. They contain a mixture of continuous and discrete variables. The relationships between the variables are not always linear, as assumed in most research. All these aspects are handled by the extensions we developed and integrated into the tool Tetrad [24], developed

by the Dept. of Philosophy of Carnegie Mellon University. To measure form-free dependencies we applied the information-theoretic definition of mutual information of two stochastic variables [9]. It is based on the entropy of a stochastic variables. A Gaussian kernel density estimator is used for the reconstruction of the underlying probability distribution from experimental data. The traditional learning algorithms fail on data containing deterministic relations since these models violate the faithfulness assumption. The latter implies that two variables that are deterministically related can contain equivalent information about a third variable, in which case either of both becomes conditionally independent of the third variable by conditioning on the other. This cannot be represented by a faithful graph. Hence it poses a problem for current learning algorithms. We solved this by redefining the faithfulness property and using the complexity of the relations as a criterion to choose among equivalent relations.

The next section gives an overview of related work, causal models are defined in section 3 and the utility for performance analysis is described in section 4. Section 5 explains the structural learning algorithms and the implemented extensions and finally, the experimental results from a performance analysis of the Aztec benchmark parallel application are given.

2. Related Work

Many tools exist for automated performance analysis. They are integrated in frameworks for coordinated monitoring and control of computer applications. It is widely recognized that the complexity of deployed systems surpasses the ability of humans to diagnose and respond to problems rapidly and correctly. Research on automated diagnosis and control - beginning with tools to analyze and interpret instrumentation data - should provide the means to guide the developer and user with understandable information.

Our research focuses on dependency analysis, when the model is not known a priori. The most common approach is to incorporate a priori models, which explicitly or implicitly represent how variables relate to each other. Other approaches let the user himself discover the interrelational structure incrementally. Current tools that support multiple experiment analysis plot performance variables (SCALEA [26]) and inefficiencies (Aksum [12]) as a function of application and system parameters. Others provide regression analysis (AIMS [30]).

We advocate applying statistical learning techniques to induce models automatically. This approach assumes little or no domain knowledge, is therefore generic and has the potential to adapt to changes in the system and its environment. Cohen and Chase use Tree-Augmented Bayesian Networks (TANs) to identify combinations of system-level metrics and threshold values that correlate with high-level performance states in a three-tier Web service under a variety of conditions [8].

Our approach provides support for extending current tools and automating certain tasks by exploiting the automatic causal learning facilities.

Most research does not consider models that contain such a wide variety of variables and relations as encountered in real performance models. They focus on one type of variables, discrete or continuous, where the continuous are most often expected to be quasi-linearly related. In case of deterministic relationships, one should exclude variables from the dataset that are definable in terms of other variables in the set [21].

3. Causal Models

This chapter will briefly introduce causal models. See [20, 23, 25] for a complete theoretic elaboration. A causal model consists of a Directed Acyclic Graph (DAG) and the Conditional Probability Distributions (CPDs) of each node. The DAG is defined over a set $V = V_1, \ldots, V_n$ of nodes, representing the variables of interest, and a set E of directed edges, or arrows, representing the relations among the variables. The interpretation of such a graph has two components: a probabilistic and a causal one [25]. But causal models are best defined in three steps. The first is reflected by a Bayesian network. Its relevance is based on the qualitative property of conditional independence.

3.1. Conditional Independence

Two stochastic variables X and Y with distributions $P(X)$ and $P(Y)$ are *probabilistically independent* when the joint distribution $P(X,Y)$ can be decomposed as $P(X,Y) = P(X).P(Y)$. On the other hand, when X and Y depend on eachother, $P(X,Y) = P(X).P(Y \mid X)$ or $P(X,Y) = P(Y).P(X \mid Y)$. One can equivalently say that X and Y are independent if $P(Y \mid X) = P(Y)$ or likewise $P(X \mid Y) = P(X)$. Dependence implies that by knowing the state of one variable, something is known about the state of the other variable. In information-theoretic terms [9], it is due to a dependency that the entropy, the uncertainty, of the unknown variable decreases when the other is known. The mutual information $I(Y;X) = H(Y) - H(Y \mid X)$ quantifies the dependency by the decrease of the entropy of Y, it is strictly positive. Both properties are defined in section 6.2.

A conditional distribution defines the probabilities of a set of variables when some other variables are known, what is called conditioning. Variables X and Y are called conditionally independent by conditioning on Z if $P(Y \mid$

$X, Z) = P(Y \mid Z)$. The knowledge of the state X adds no information to the knowledge of Z about Y. Conditional independence is denoted by the ternary operator $.\perp\!\!\!\perp. \mid .$

$$X \perp\!\!\!\perp Y \mid Z \Leftrightarrow P(X \mid Y, Z) = P(X \mid Z) \qquad (1)$$

The above definitions also apply for sets, by simply replacing the variables by sets.

3.2. Bayesian Networks

A Bayesian network offers a dense representation of a *joint distribution*. A joint distribution is defined over a set of stochastic variables $X_1 \dots X_n$ and defines a probability ($P \in [0, 1]$) for each possible state $(x_1 \dots x_n) \in X_{1,dom} \times \cdots \times X_{n,dom}$, where $X_{i,dom}$ stands for the domain of variable X_i.

The distribution can be *factorized* relative to a variable ordering $[X_1, \dots, X_n]$ as follows:

$$P(X_1 \dots X_n) = \prod_i^n P(X_i \mid X_1 \dots X_{i-1}) \qquad (2)$$

Variables can be removed from the conditioning sets if they get conditionally independent of the probability variable by conditioning on the rest of the set. This follows directly from the definition: if $P(X_i \mid X_1 \dots X_{i-1}) = P(X_i \mid X_1 \dots X_{j-1}, X_{j+1} \dots X_{i-1})$, then X_j can be eliminated from the factor of X_i. Such conditional independencies reduce the complexity of the factors in the factorization. The conditioning sets of the factors can be described by a Directed Acyclic Graph (DAG), in which each node has an incoming edge from all variables of the conditioning set of its factor. The joint distribution is then described by the DAG and the conditional probability distributions (CPDs) of all variables conditioned on its parents, $P(X_i \mid parents(X_i))$. A *Bayesian network* is a factorization that is minimal, in the sense that no edge can be deleted without destroying the correctness of the factorization. Note that a graph still depends on the variable ordering. Some orderings lead to the same graphs, but others result in different graph that possibly contain more edges.

A simple example can illustrate the theory. Take 4 stochastic variables A, B, C and D. Their joint distribution is written as $P(A, B, C, D)$ and defines a probability ($P \in [0, 1]$ for each possible state $(a, b, c, d) \in A_{dom} \times B_{dom} \times C_{dom} \times D_{dom}$. Fig. 1 shows how the factorization is simplified by a conditional independency found in the distribution.

We constructed the definition of the Bayesian network by the conditional independencies. On the other hand, the independencies can be read from the graph by the Markov condition. The *Markov condition* states that all non-descendants become independent from a variable by conditioning on the parents of that variable. Combining the independencies that follow from the Markov condition result in other independencies. Pearl developed a graphical criterion, called *d-separation*, to retrieve all independence relations that follow from the Markov condition from a DAG [13]. The DAG is called an *Independence Map* (I-map), since all independencies are present in the distribution. The converse is however not true for all factorizations. This is discussed in the next section.

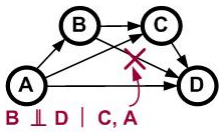

Figure 1. Reduction by conditional independencies of a factorization, based on variable ordering $\{A, B, C, D\}$.

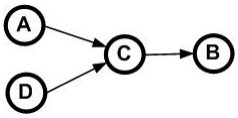

Figure 2. Minimal factorization, based on variable ordering $\{A, D, C, B\}$.

3.3. Faithful Bayesian Networks

In the second step of the construction of causal models, we seek for Bayesian Networks that represent *all* conditional independencies of a joint distribution. A model is called *faithful* to a distribution if all and no more independencies found in the graph with the $d - separation$ criterion appear in the distribution. It is proven that if there exists a faithful graph, it is the factorization based on a variable ordering that leads to the minimal number of edges. Fig. 2 shows the factorization with the fewest edges that results from a different variable ordering. The *d-separation* criterion tells us, for example, that variable C separates A from B and also A from D. C blocks the path from A to B. On the other hand, A is initially independent from D, but becomes dependent when conditioned on C. A faithful model is a dense representation of the conditional independencies of a distribution. Note that the graph of Fig. 1 has 2 edges more due to a worse variable ordering. A must for example be connected with D because $A \not\perp\!\!\!\perp D \mid C$, unless $A \perp\!\!\!\perp D$. Both independencies can be read from the graph of Fig. 2: $A \rightarrow C \leftarrow D$ form a $v - structure$.

3.4. Causally interpreted Bayesian Networks

A causal relation is an irreflexive, transitive and asymmetrical (rain creates mud, but mud will not create rain) relation. It has the properties of productivity (the effect is 'produced' by the cause) and locality [7]. It obeys the markov condition (for model $A \rightarrow B \rightarrow C$, if B is blocked, than A does not cause C) and represents a stable and autonomous physical mechanism ("which is conceivable to change one relationship without changing the others") [20].

A *causal model* is a Bayesian network in which the arrows are viewed as representing causal influences between the corresponding variables. This interpretation, attributed to the edges of a faithful model, is based on a reductionist motivation. The model breaks up into the submodels $P(X_i \mid parents(X_i))$ for each variable X_i. Each submodel represents a stochastic process by which the values of X_i are chosen in response to the values of $parents(X_i)$, and the stochastic variation of this assignment is assumed independent of the variations in all other assignments. Moreover, each assignment process remains invariant to possible changes in assignment processes that govern other variables in the system. This modularity assumption enables us to predict the effect of interventions, whenever interventions are described as specific modifications of some factors in the product of the factorization. Pearl defines an intervention as 'surgically' setting a variable to a certain state, resulting in a mutilated model in which all links from the parents are removed [20]. Here, the asymmetry comes into play, since only the effects remain connected to the variable.

The submodels $P(X_i \mid parents(X_i))$ are the fundamental blocks that are able, by the faithfulness property, to explain all relational regularities that can be observed. The rationale is that if a model offering a minimal description of the data has the power to foresee all consequences, it must come close to reality. Nevertheless, the causal interpretation of the relations cannot be guaranteed, unless experiments are performed studying the effects of changes to the system.

4. Causal Performance Models of Computers

The model of parallel performance shown in Fig. 3 is constructed by an expert in an intuitive way, for what he expects as being a true and useful model. It shows the most common variables involved in the performance analysis of a parallel application. The parallel performance metric uses an overhead quantification based on the lost-cycle approach [10]. It aims at attributing each part of the overhead - the

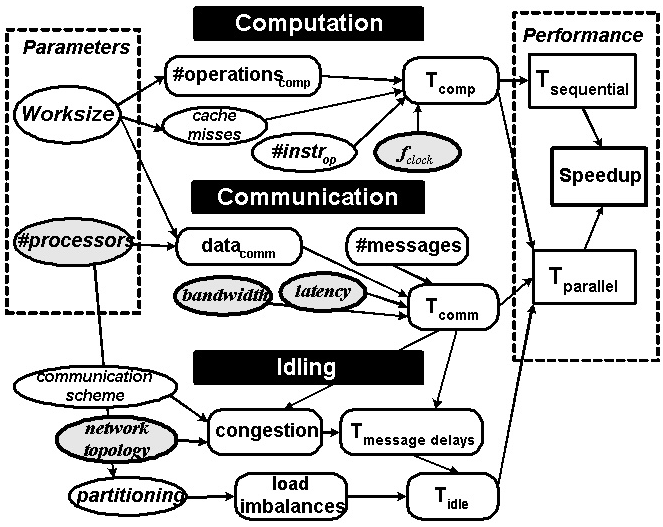

Figure 3. General model of parallel performance

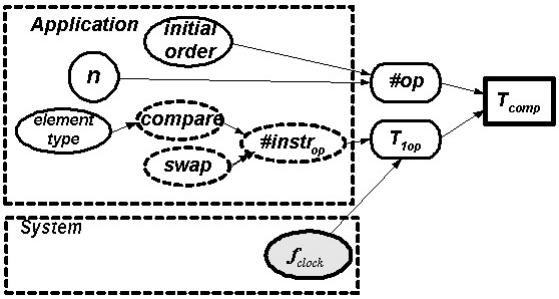

Figure 4. Simplified Causal Model of Quicksort

so-called *lost cycles* - to the purpose it was spent on. Program parameters and computer system characteristics (gray ovals) influence the overheads. The $worksize$ of the application and the number of processors ($\#processors$) are input parameters. Other system parameters are depicted by gray ovals. The performance variables at the right side reflect the output of the model. Intermediate variables like the communication time T_{comm} and idle time T_{idle} measure the overheads that can explain the performance results. Other variables can further explain the values for the overheads. Most of them can be measured physically: in the application (eg. the number of basic operations $\#operations_{comp}$ performed by the algorithm), in the system (like the cache misses) or by MPI profiling capabilities (eg. the size of the communication data $data_{comm}$ or the number of messages). Finally, the number of instructions per operation ($\#instr_{op}$), the *partitioning* and the *communication scheme* denote characteristics of the application.

Fig. 4 represents a causal model of performance related data of a quicksort running on a sequential computer. It represents a first-order performance model for the sequential runtime T_{comp}. $\#op$ is the number of basic compare-swap operations, which is determined by the array size n and the $initial\ order$ of the elements. The complexity of the $compare$ and $swap$ statements influence the a number of basic instructions $\#instr_{op}$. Together with the processor's clock frequency f_{clock} they determine the time for 1 operation T_{1op}. In the example of Fig. 4, once we know the number of operations $\#op$, T_{comp} becomes independent of n or the $initialorder$. This shows the reduction of the dependency complexity of the model.

By analyzing of the properties attributed to this model, we find that they correspond to those that define causal models. A variable is expected to be determined by its parents, this corresponds to the Markov condition. The model is assumed not to contain redundant relations, thus to be minimal. Furthermore, the paths between the variables show the dependencies among the variables, which corresponds to the faithfulness condition. In the formalization employed by the research that led to the development of the TETRAD tool [23], the 3 properties - the Markov condition, minimality and faithfulness - are taken as axioms for defining causality.

The questions that are supposed to be answered by a performance analysis are of causal nature. Take the study of network performance: communication delays should be attributed to the different steps of the communication process, like machine latency, transfer time, network contention, flight time, etc [4]. A correct understanding of the origins is indispensable. The task of identifying them becomes even more difficult when implementation-specific low level issues come into play, such as specific protocol behavior, window delays or chatter [19], since these are not always fully understood and often cannot be measured directly. The relations between quantities observed throughout the application and computer system, and the overall performance have a causal interpretation.

5. Utility

The utility of causal models is twofold: they support the modeler as well as the user.

5.1. Support of the performance modeling process

5.1.1. Model construction , in the first place.

5.1.2. Model validation: validation of the (in)dependency assumptions made by the modeler.

5.1.3. Reuse of autonomous submodels: Causality takes the position that the world is reducible. Models for which the Markov Conditions is valid can be separated into independent, local submodels of the form $P(X_i \mid pa_i)$. Independency results in autonomy, a submodel can be altered without affecting the rest of the model. Causality inherently provide autonomous submodels that can be reused in modeling systems in a different context. This approach claims that the statistical analysis of a parallel application running on a certain network, results in a reusable model which correctly predicts the communication delays versus the - application independent - message properties for experiments with different applications.

The ultimate goal of performance modeling is to be able to predict the runtime of applications on unknown systems. This requires the existence of independent application and system characteristics and a simple functional relation to calculate the overall performance. This is the case in the simplified quicksort model of Fig. 4, where the performance characteristics are $\#op$, $\#cycles_{op}$ for the application and f_{clock} for the system. The runtime can be computed by:

$$T_{comp} = \#op.\#cycles_{op}.1/f_{clock} \tag{3}$$

However, the first-order approximation only holds for small problem sizes. When memory overheads come into play, as shown by the extended model of Fig. 5, the separation becomes more interesting [22]. Independency of submodels and application or system characteristics is crucial in the modeling process. Such independencies reflect precisely what is studied by causality.

5.1.4. Detection of abnormal, unexpected dependencies: Consider dynamic, complex or non-homogeneous systems with exceptionally high overheads. A statistical model of the communication performance can be used to warn for exceptional communication delays, when above the average, useful in for example GRID environments [11].

5.1.5. Flexibility: additional information can easily be integrated into the models. Even incomplete information can give relatively accurate estimates. At each stage, it should be possible to refine models [18]. Fig. 5 shows an extended version of the quicksort performance model of Fig. 4. Memory overheads, denoted by T_{memory}, were added to the model. They are caused by the application's $datasize$, $memory$ usage and the processor's $memory$ capacity and bandwidth.

The modeler can integrate additional information into the model, like the cache miss frequencies (measured with, for example, the PAPI tool for accessing hardware counters on microprocessors [6]) or the processor type, as shown in

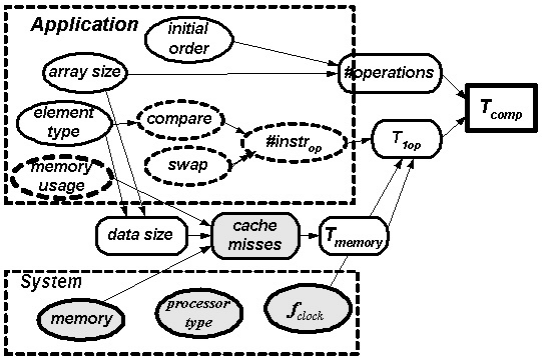

Figure 5. Detailed Causal Model of Quicksort Performance

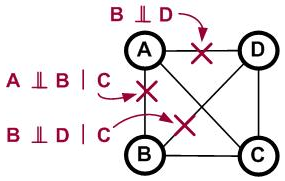

Figure 6. First part of the learning algorithm: edges are removed from the full-connected undirected graph due to conditional independencies.

Fig. 5 . Through statistical analysis, the dependencies with the other variables can be found and the predictive qualities of this extra information can refine the performance model.

Not all variables should be known for performance prediction. By the use of statistics, expectations can be calculated for unknown variables. In this way, it should be possible to create flexible, hierarchical models.

On the other hand, not all variables should be known for performance prediction. The statistical expected value can be used for unknown variables.

5.2. Presentation of a clear performance report

5.2.1. Structuring the variables In order to understand complex situations with many variables and dependencies, a structured representation of the relations is required. This is offered by causal models.

5.2.2. Filtering relevant information Causal models correspond to physical mechanisms and enable the filtering of relevant information, by the statistical analysis, which reveals the impact of every factor, the most influential factors can be filtered.

5.2.3. Reasoning about interventions Besides the explanatory facilities, causal models can be exploited to *reason* about the performance and answer questions like: Which part of the application gives space for adequate optimization? What is the most efficient upgrade of the system?

6. Causal Structure Learning

Besides the formal treatment of causality by Pearl, another important advance was the design of algorithms for learning causal models from experimental data by [28, 23]. Various tools exist that implement these algorithms, for an overview see [14] or [2]. TETRAD (free available at [24]) is open source software, written in Java and contains an extensive set of algorithms. There are two basic types of structure learning algorithms: constraint-based and scoring-based. All learning algorithms of TETRAD are of the constraint-based type [23]. The algorithms consist of two parts: first they construct an undirected graph by finding *direct relations*, which is the same for all algorithms. Secondly, the algorithms try to direct the edges using orientation rules.

The first step of the standard algorithm, *adjacency search*, is based on the property that direct relations cannot become probabilistically independent upon conditioning on some other set of vertices (see the Markov condition). Indirectly related variables become independent by conditioning on some variables on the path between both variables. The algorithm starts with a full-connected undirected graph and removes all edges for which a conditioning set can be found that renders both variables independent. The algorithm will go through all subsets of variables and check for conditional independence. If a test is successful, the edge is removed. The algorithm starts by checking unconditional correlations and then gradually adds nodes to the conditioning set upto a certain maximal number. It selects the nodes in an optimized way to minimize the tests it has to perform. Fig. 6 shows this part of the algorithm for learning the model of Fig. 2.

Secondly, the algorithm tries to direct the edges using orientation rules. These rules are based on the detection of *v-structures*. If three variables are connected by two edges, for example $A - B - C$, there are four possibilities to orient both edges. Only the v-structure can recognized among these. For $A \rightarrow B \leftarrow C$, A and C are initially independent, but become dependent by conditioning on B. For all three other orientation possibilities it is the opposite, A and C are initially dependent, but become independent by conditioning on B. Applied on the undirected graph of Fig. 6, v-structure $A \rightarrow C \leftarrow D$ will be recognized. The $B - C$ relation can be oriented as $B \leftarrow C$, since an opposite orientation leads to v-structure $A \rightarrow C \leftarrow B$ that cannot be recognized in the data. Absence of v-structures leads to

edges that cannot be directed. Thus, in general, the learning algorithm leads to a set of observationally indistinguishable models. The orientation task is however simplified by the knowledge of the input and output variables and any confusion about the direction of some edges can be resolved by an expert since the orientation of the relations in performance models is in most cases trivial.

Under 6 assumptions the algorithm will find the correct equivalence class of indistinguishable equivalent causal models [21]. Models of the same class have the same undirected graph, only the orientation cannot always be determined. But as we already explained, orientation is not our primary concern. One assumption is that the experiment should be typical, random. We will choose the input parameters randomly from a uniform distribution. Another assumption states that the faithfulness condition should hold. This property is violated when there are deterministic relationships among the variables. Performance models contain inevitably deterministic relations. This issue is discussed in section 6.4. The other 4 assumptions are valid.

6.1. Extensions

To capture the complexity of performance data, the existing models and algorithms had to be extended in three ways:

- Mixture of discrete and continuous data, by the use of the information-theoretic concept of mutual information for measuring dependency. It measures the decrease of uncertainty of one variable by knowing another variable.
- Another advantage of mutual information is that it offers a form-free dependency measure, whereas the widely used Pearson correlation coefficient measures the closeness of a relation to linearity.
- Deterministic relations, by which a variable is a function of a set of variables, imply additional conditional independencies that cannot be captured by faithful models. Our work developed extensions of the definition and learning algorithm that is able to reestablish the faithfulness.

Our approach to handle these types of variables and relations is elaborated in the next subsections.

6.2. Information-Theoretic Dependence

TETRAD uses Pearson's correlation coefficient for calculating the dependency of two continuous variables. It gives a measure of how close a relation approximates linearity. Conditional independencies are measured by partial correlations, which can be calculated from the correlation coefficients, but only if linearity holds. Correlations can measure non-linear relations, as long as they are quasi-monotonically increasing or decreasing. The partial correlations will possibly fail. This was confirmed by our experiments. This motivates the use of a form-free definition of dependency.

Entropy is the amount of uncertainty of a stochastic variable. For a discrete random variable X with alphabet A and probability mass function $p(x)$, its entropy is defined as [9]

$$H(X) = -\sum_{x \in A} p(x) log p(x) \tag{4}$$

It represents the number of bits for the minimal code that can describe x and is maximal for the uniform distribution. The conditional entropy $H(Y \mid X)$ is defined as

$$H(X \mid Y) = \sum_{y \in B} H(X \mid Y = y) \tag{5}$$

where B is the alphabet of random variable Y. Probabilistic dependency can be defined as the mutual information $I(X;Y)$ of X and Y, which is the reduction in uncertainty of X when knowing Y:

$$I(X;Y) = H(X) - H(X \mid Y) \tag{6}$$

It is zero when both variables are independent. The mutual information can then be rewritten as

$$I(X;Y) = \sum_{x \in A} \sum_{y \in B} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{7}$$

Conditional independence, $I(X;Y \mid Z)$, is defined in the same way. The G^2 independence test [5], used by TETRAD for discrete variables, is of the same form:

$$G^2 = 2 \sum observed.ln \frac{observed}{expected} \tag{8}$$

For measuring the entropy of continuous variables, we will use the entropy of the discretized distribution. For a variable X with density $f(x)$ and dividing the range of X in bins of length Δ , the entropy of the quantized distribution is [9]

$$H(X^{\Delta}) = -\int_{-\infty}^{+\infty} f(x_i) \Delta log(f(x_i)\Delta) \tag{9}$$

The discretized definition of entropy and mutual information are of the same form as respectively Eq. 4 and Eq. 7. This makes it possible to handle continuous and discrete variables identically and to calculate entropies for a mixture of continuous and discrete variables. The difference with Pearson's correlation coefficient is that mutual information considers each value of X independently: it sums for every x the decrease in uncertainty of Y. The relation between X and Y can be arbitrary, whereas correlation

seeks for a linear relation between the points. The main disadvantage of our test is that larger sample sizes are needed. The definition suggests that for every value of X multiple data points are needed. This approach would be the same as discretizing the continuous variable and would also induce a quantification error. Kernel density estimation overcomes these problems. The reason is that $P(X = x)$ is influenced by the data points in the neighborhood of x. The number of data points needed can be limited, the estimation only assumes a smoothly changing distribution.

6.3. Kernel Density Estimation

For applying the information-theoretic definition of dependency, it is necessary to have an estimation of the probability distributions from the data. For discrete variables the distribution can be estimated by simply counting the number of occurrences of each state and divide them by the number of data points n. For continuous variables, *kernel smoothing* makes it possible to estimate the distribution with limited sample sizes. The kernel estimate is constructed by centering a scaled kernel at each observation. The value of the estimate at point x is the average of the n kernel ordinates at that point. The idea is to spread a 'probability mass' of size $1/n$ associated with each data point about its neighborhood. See [3] for a gentle introduction. The estimated distribution is the result of a convolution of the data points with a well-chosen kernel [29]:

$$p(x) = \frac{1}{nb} \sum_{i=1}^{n} K(\frac{x - x_i)}{b}) \qquad (10)$$

with n the sample size and $K(.)$ the multivariate kernel function, which is symmetric and satisfies $\int k(x)\Delta x = 1$. The factor b is the smoothing bandwidth and determines the width of the kernel. Theoretic analysis and simulation have shown that the choice of the kernel is not crucial, but that bandwidth is the determining factor for having good estimations. We chose the Gaussian function for kernel and for the bandwidth 4 times the range divided by the number of data points.

For estimating a multivariate distribution, a multidimensional Gaussian is used with a specific bandwidth for each dimension.

6.4. Deterministic Relations

The existence and construction of faithful causal models are based on the property that adjacent nodes share unique information. This treatment becomes invalid if some variables contain the same information about a target variable. We call the variables *information-equivalent* for the target, if they have information about the target. Consider X and Y being equivalent for Z, it follows that

$$I(X; Z) > 0 \ \wedge \ I(X; Z|Y) = 0 \ \wedge \ I(Y; Z|X) = 0 \quad (11)$$

either of both variables become conditionally independent from Z by conditioning on the other. Models containing such conditional independencies cannot be represented by a faithful graph. The adjacency search (section 6) will fail in constructing such a model, since it would remove both edges $X - Z$ and $Y - Z$. Situations of information-equivalence are mostly entailed by deterministic relationships. A variable X is determined by a set of variables $(Y_1, Y_2, \ldots, Y_n)$, denoted by $\boldsymbol{Y}$, if $X = f(\boldsymbol{Y})$ is a function. The variables $\boldsymbol{Y}$ contain all information about X, the conditional entropy of X becomes zero due to the knowledge of $\boldsymbol{Y}$. This implies that X gets conditionally independent from any other variable by conditioning on $\boldsymbol{Y}$:

$$I(X; Z \mid \boldsymbol{Y}) = H(X \mid \boldsymbol{Y}) - H(X \mid \boldsymbol{Y}, Z) = 0 \qquad (12)$$

for any Z. The second term is also zero, since conditioning can never increase the entropy. X cannot contain additional information about any other variable. For variables from which $\boldsymbol{Y}$ also gets independent by conditioning on X, X and $\boldsymbol{Y}$ are information-equivalent, as expressed by Eq. 11.

Our research developed an approach for defining and learning faithful models containing information-equivalent variables [16]. The main idea is that one of the equivalent variables is selected to be directly related to the target variable. The criterion we propose is the *complexity* of the relation, by which the simplest relation should be preferred. In our paper [16] we show that faithfulness can be reestablished by extending the definition of conditional independence with the complexity criterion for information-equivalent relations. It is shown that this leads to consistent models under the assumption that the complexity of relations do not decrease along a causal path. Take the model $A \rightarrow B \rightarrow C$, the relation between A and C will not be simpler than the relations $A - B$ and $B - C$. The adjacency search procedure was modified for finding equivalent variables by comparing the conditional independence tests according to 11. An extra step was added to the algorithm in which the direct edge among the equivalent edges was decided.

7. Correct Model Construction (Aztec)

The learning algorithm will be validated by applying it on performance data retrieved from experiments with the Aztec benchmark library [27, 1]. It provides an iterative algorithm for parallel solving of partial differential equations, defined over a 3-dimensional grid. It supports 2 sparse matrix formats, a point-entry modified sparse row (MSR) format and a block-entry variable block row

(VBR) format. Experiments are run on a dedicated cluster of 8 PentiumII computers connected by a 100MHz non-blocking switch. The number of equations is varied between 100 and 400 and the number of grid points between 5^3 and 20^3. The same experiments are performed for both matrix formats. Our performance analysis tool EPPA [18, 17] uses the MPI profiling facilities to automatically trace the MPI calls and writes them to a database. A tool is used to retrieve data from the database, write it to a TAB-separated files, which are readable by TETRAD. The *PC algorithm*, extended with our modules as described in the previous sections, is applied with default options.

7.1. Overall performance

Fig. 7 shows the model of Aztec's performance constructed by TETRAD. The variables are ordered from input to output, starting with the 4 input parameters at the left and the parallel $runtime$ and $speedup$ at the right. It is immediately clear that the orientation of the edges are not correct, since we expect them to be directed from left to right. As explained before, this is not our main concern. At the time of writing the paper, the knowledge of the input and output variables could not be inserted yet and the effects of deterministic relations on the orientation rules were still unclear. The undirected graph that was learned is more interesting, since it shows how the performance is generated. In order to understand better the overall performance, two additional variables, which the algorithm returns at the end of the execution, were also registered: the total number of floating point operations needed for the computation ($totalFlops$) and the number of iterations ($totalIterations$) that the algorithm performed. The model (Fig. 7) reveals that the variable $totalIterations$ is completely determined by the number of grid points and does not affect the $speedup$ in a direct way. It is $totalFlops$ that incorporates all information of $totalIterations$ about the $runtime$, communication time T_{comm} and idle time T_{idle}. We also added the variable $eqXpoints$, which is simply the product of $nbrEquations$ and $nbrGridPoints$. It is a useful variable since it explains how $totalFlops$ depends on $nbrEquations$ and $nbrGridPoints$. $totalFlops$ gets much bigger for the VBR matrix format as for the MSR format. Finally, the $speedup$ depends on the number of processors, the computation, communication and idle time.

7.2. Communication performance

Fig. 8 shows the learned model of the communication performance. The 4 input parameters are at the left side and the total communication time T_{comm} at the top right. The figure shows the result of the adja-

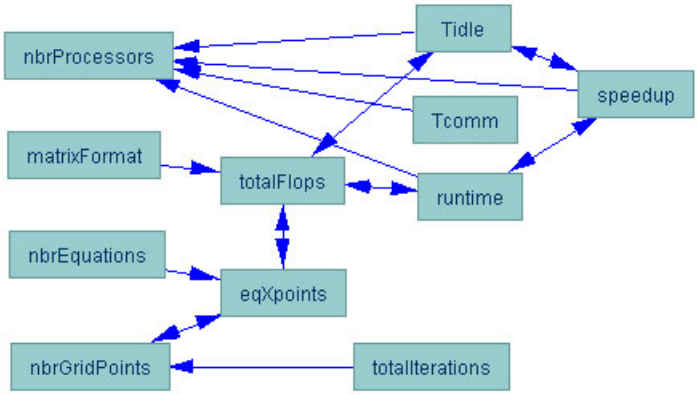

Figure 7. Causal model of Aztec's performance

cency search step only, we omitted the orientation step because the results were unreliable as in the previous model. Nevertheless, the undirected graph shows how T_{comm} is generated, when we imagine the relation going from left to right and down to top. The communication performance is completely defined by the number of processors, the number of messages and the size of the communicated data. We registered 4 variables that are calculated by the algorithm when the partitioning of the matrices is finished. The $internalUnknowns$ are the elements that can be updated using only information on the current processor. $externalUnknowns$ refers to the off-processor elements that are required during the calculations by the $borderUnknowns$ elements. $unknownsSentToNeighbors$ represents the number of elements actually sent. These definitions confirm the model. The communication is primarily affected by $unknownsSentToNeighbors$, which on its turn is influenced by $borderUnknowns$ and this by $externalUnknowns$. The VBR matrix format generates more messages than the MSR format. The variable $eqXpoints$, which is the product of $nbrEquations$ and $nbrGridPoints$, determines $internalUnknowns$ in combination with the number of processors. On the contrary, the relation of $externalUnknowns$ with $nbrEquations$ and $nbrGridPoints$ cannot be replaced by $eqXpoints$ alone.

8. Conclusions

Causal modeling provides a theoretical framework for capturing relational regularities in performance data of computer systems and for dividing a complex model into independent submodels. Algorithms can detect these regularities in experimental data and construct causal models.

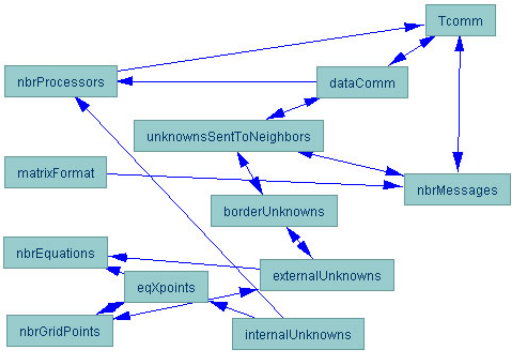

Figure 8. Causal model of Aztec's communication time

We enlarged the scope of existing structure learning algorithms by using the form-free information-theoretic concept of mutual information and by introducing the complexity criterion for selecting direct relations among equivalent relations.

Experiments with the Aztec benchmark application show that accurate models are learned. Causal modeling make explicit what is done by the scientist when analyzing performance and allow further automation.

References

[1] Aztec homepage. http://www.cs.sandia.gov/CRF/aztec1.html.

[2] Bn software overview. http://www.cs.ubc.ca/ murphyk/Software/BNT/bnsoft.html.

[3] Gentle introduction to kde. http://www.maths.uwa.edu.au/ duongt/seminars/intro2kde/.

[4] R. M. e. a. Badia. Dimemas: Predicting mpi applications behavior in grid environments. In *Workshop on Grid Applications and Programming Tools (GGF8)*, 2003.

[5] Y. M. M. Bishop, S. E. Fienberg, and Holland. *Discrete multivariate analysis. Theory and practice*. Cambridge: MIT Press, 1975.

[6] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *International Journal of High Performance Computing Applications*, 14(3):189–204, 2000.

[7] M. Bunge. *Causality and Modern Science*. Dover Publications, New York, 1979.

[8] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *OSDI*, pages 231–244, 2004.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[10] M. Crovella and T. J. LeBlanc. Parallel performance using lost cycles analysis. In *SC*, pages 600–609, 1994.

[11] B. B. et all. Performance evaluation and monitoring of interactive grid applications. In Kranzlmüller et al. [15].

[12] T. Fahringer and C. S. Jr. Automatic search for performance problems in parallel and distributed programs by using multi-experiment analysis. In S. Sahni, V. K. Prasanna, and U. Shukla, editors, *HiPC*, volume 2552 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2002.

[13] D. Geiger, T. Verma, and J. Pearl. d-separation: From theorems to algorithms. In M. Henrion, R. D. Shachter, L. N. Kanal, and J. F. Lemmer, editors, *UAI*, pages 139–148. North-Holland, 1989.

[14] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. CRC Press, 2003.

[15] D. Kranzlmüller, P. Kacsuk, and J. Dongarra, editors. *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, September 19-22, 2004, Proceedings*, volume 3241 of *Lecture Notes in Computer Science*. Springer, 2004.

[16] J. Lemeire. Causal models as minimal descriptions of multivariate systems. http://parallel.vub.ac.be/ jan.

[17] J. Lemeire. Documentation of eppa tool (experimental parallel performance analysis). http://parallel.vub.ac.be/eppa.

[18] J. Lemeire, A. Crijns, J. Crijns, and E. F. Dirkx. A refinement strategy for a user-oriented performance analysis. In Kranzlmüller et al. [15], pages 388–396.

[19] I. NetPredict. Common mistakes in performance analysis, white paper. NetPredict Inc, 2003.

[20] J. Pearl. *Causality. Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[21] R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. *TETRAD 3: Tools for Causal Modeling - User's Manual*. http://www.phil.cmu.edu/projects/tetrad/tet3/master.htm, 1996.

[22] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. M. Badia, and A. Purkayastha. A framework for performance modeling and prediction. In *SC*, pages 1–17, 2002.

[23] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer Verlag, 2nd edition, 1993.

[24] P. Spirtes, C. Glymour, R. Scheines, and J. Ramsey. The tetrad project. http://www.phil.cmu.edu/projects/tetrad/.

[25] J. Tian and J. Pearl. A general identification condition for causal effects. In *AAAI/IAAI*, pages 567–573, 2002.

[26] H. L. Truong and T. Fahringer. Scalea: A performance analysis tool for distributed and parallel programs. In B. Monien and R. Feldmann, editors, *Euro-Par*, volume 2400 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2002.

[27] R. S. Tuminaro, M. Heroux, S. A. Hutchinson, and J. N. Shadid. Official aztec user's guide: Version 2.1. Technical Report SAND95-1559, Sandia National Laboratories, dec 1999.

[28] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *In Proc. of the 6th workshop on uncertainty in Artifical Intelligence*. Cambridge, 1991.

[29] M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall, London, UK., 1995.

[30] J. Yan, S. Sarukkai, and P. Mehra. Performance measurement, visualization and modeling of parallel and distributed programs using the aims toolkit. *Software - Practice and Experience*, 25(4):429–461, 1995.