



HAL
open science

Towards an Integral Approach for Modeling Causality

Stijn Meganck

► **To cite this version:**

Stijn Meganck. Towards an Integral Approach for Modeling Causality. Machine Learning [cs.LG]. INSA de Rouen; Vrije Universiteit Brussels, 2008. English. NNT: . tel-00915256

HAL Id: tel-00915256

<https://theses.hal.science/tel-00915256>

Submitted on 6 Dec 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Stijn Meganck
Ph.D. dissertation

Towards an Integral Approach for Modeling Causality

September 24, 2008

Promotor: Prof. Dr. Bernard Manderick
Co-Promotor: Prof. Dr. Philippe Leray

Abbreviations

| | |
|-----------|---|
| AG | Ancestral Graph |
| BN | Bayesian Network |
| BNT | Bayes Net Toolbox |
| CBN | Causal Bayesian Network |
| CPAG | Complete Partially Ancestral Graph |
| CPD | Conditional Probability Distribution |
| CPDAG | Complete Partially Directed Acyclic Graph |
| DAG | Directed Acyclic Graph |
| GES | Greedy Equivalence Search |
| GS | Greedy Search |
| I-map | Independence Map |
| JT | Junction Tree |
| JPD | Joint Probability Distribution |
| M-A CaDo | Multi-Agent Causal DiscOvery |
| MAG | Maximal Ancestral Graph |
| MAP | Maximum A Posteriori |
| MCMC | Monte Carlo Markov Chain |
| MDL | Minimum Description Length |
| ML | Maximal Likelihood |
| MyCaDo | My Causal DiscOvery |
| p.d. path | potentially directed path |
| SMCM | Semi-Markovian Causal Model |
| UnCaDo | Unsure Causal DiscOvery |

Symbols

| | |
|-------------------------|--|
| $\perp\!\!\!\perp$ | independent |
| \nsim | dependent |
| \perp | d -separated |
| Π_i | Parents of variable X_i |
| $Ch(X_i)$ | Children of variable X_i |
| $Anc(X_i)$ | Ancestors of variable X_i |
| $Desc(X_i)$ | Descendants of variable X_i |
| $X_i \circ - o X_j$ | Edge with unknown endpoints |
| $X_i \circ - ? - X_j$ | Edge with no arrow into X_j |
| $X_i \circ - ? - o X_j$ | Unsure edge |
| \rightsquigarrow | varies with |
| $exp(X_i)$ | Performing an experiment on variable X_i |

Contents

| | | |
|----------|----------------------------|----|
| 1 | Introduction | 1 |
| 1.1 | Preliminaries | 1 |
| 1.1.1 | Causality | 1 |
| 1.1.2 | Graphical Models | 2 |
| 1.2 | Model Requirements | 3 |
| 1.3 | Causal Bayesian Networks | 4 |
| 1.3.1 | Probabilistic inference | 6 |
| 1.3.2 | Causal Inference | 7 |
| 1.3.3 | Learning | 7 |
| 1.4 | More Expressive Models | 7 |
| 1.4.1 | Latent Variable Modeling | 7 |
| 1.4.2 | Multi-Agent Causal Models | 9 |
| 1.5 | Research Proposition | 10 |
| 1.5.1 | Overview and Contributions | 11 |
| 2 | Bayesian Networks | 13 |
| 2.1 | Probability Theory | 13 |
| 2.1.1 | Probability Distributions | 13 |
| 2.1.2 | Independence | 14 |
| 2.2 | Graph Theory | 15 |
| 2.3 | Bayesian Networks | 16 |
| 2.3.1 | Definition | 16 |
| 2.3.2 | Causal Sufficiency | 17 |
| 2.3.3 | Markov Condition | 17 |
| 2.3.4 | Faithfulness | 18 |
| 2.3.5 | Factorization of the JPD | 19 |
| 2.3.6 | Markov Equivalence | 19 |

| | | |
|----------|---|-----------|
| 2.4 | Probabilistic Inference | 21 |
| 2.4.1 | $\lambda - \pi$ Message Passing | 21 |
| 2.4.2 | Junction Tree Algorithm | 22 |
| 2.5 | Learning Bayesian Networks | 24 |
| 2.5.1 | Learning the Parameters | 24 |
| 2.5.2 | Learning the Structure | 25 |
| 2.6 | Overview of the Chapter | 31 |
| 3 | Causal Bayesian Networks | 33 |
| 3.1 | Definition | 33 |
| 3.2 | Causal Inference | 35 |
| 3.2.1 | Observation vs. Intervention | 35 |
| 3.2.2 | Manipulation Theorem | 36 |
| 3.3 | Structure Learning | 39 |
| 3.3.1 | Learning from Observational Data | 40 |
| 3.3.2 | Learning from Experimental Data | 41 |
| 3.4 | Overview of State-of-the-art | 42 |
| 3.4.1 | Properties of Learning Algorithms | 42 |
| 3.4.2 | CBN Structure Learning with Structural Interventions | 43 |
| 3.4.3 | Alternative Interventions | 45 |
| 3.5 | Overview of the Chapter | 47 |
| 4 | Contributions on Causal Modeling | 49 |
| 4.1 | Our Greedy Approach for Perfect Observational and Experimental Data | 49 |
| 4.1.1 | General Description | 49 |
| 4.1.2 | Choice of Experiment | 50 |
| 4.1.3 | Decision Criteria | 52 |
| 4.1.4 | Performing Structural Experiments | 54 |
| 4.1.5 | Result Analysis | 54 |
| 4.1.6 | Detailed Learning Algorithm | 55 |
| 4.1.7 | Toy Example | 55 |
| 4.1.8 | Experiments and Results | 58 |
| 4.1.9 | Critical Discussion of MyCaDo | 62 |
| 4.2 | Constraint-based Method for Imperfect Observational Data | 62 |
| 4.2.1 | General Description | 63 |
| 4.2.2 | Unsure Independence Test | 64 |
| 4.2.3 | Initial Phase: <i>Unsure</i> Observational Learning | 64 |

| | | |
|----------|---|-----------|
| 4.2.4 | Experimentation Phase: Resolving <i>Unsure</i> Edges | 65 |
| 4.2.5 | Completion Phase | 67 |
| 4.2.6 | Complete Learning Algorithm | 67 |
| 4.2.7 | Toy Example | 67 |
| 4.2.8 | Critical Discussion of UnCaDo | 69 |
| 4.3 | Overview of the Chapter | 69 |
| 5 | Causal Latent Networks | 73 |
| 5.1 | Ancestral Graphs | 73 |
| 5.1.1 | Definitions | 73 |
| 5.1.2 | Semantics | 74 |
| 5.1.3 | Advantages/Disadvantages of AG | 75 |
| 5.2 | Semi-Markovian Causal Models | 75 |
| 5.2.1 | Definitions | 75 |
| 5.2.2 | Semantics | 76 |
| 5.2.3 | Advantages/Disadvantages of SMCM | 76 |
| 5.3 | Causal Inference | 77 |
| 5.3.1 | C-components and C-factors | 77 |
| 5.3.2 | Identification | 79 |
| 5.3.3 | Sketch of Algorithm | 81 |
| 5.3.4 | Technical introduction | 84 |
| 5.3.5 | Algorithms | 91 |
| 5.4 | Learning Causal Latent Networks | 92 |
| 5.4.1 | Learning the Parameters | 92 |
| 5.4.2 | Learning the Structure | 92 |
| 5.5 | Overview of the Chapter | 95 |
| 6 | Contributions on Latent Variable Causal Modeling | 97 |
| 6.1 | Learning Causal Latent Networks | 97 |
| 6.1.1 | Preliminaries | 98 |
| 6.1.2 | General Overview of the Algorithm | 98 |
| 6.1.3 | Performing Experiments | 99 |
| 6.1.4 | Solving $o \rightarrow$ | 100 |
| 6.1.5 | Solving $o - o$ | 101 |
| 6.1.6 | Removing Inducing Path Edges | 102 |
| 6.1.7 | Example | 103 |
| 6.1.8 | Critical Discussion of MyCaDo++ | 105 |

| | | |
|----------|---|------------|
| 6.2 | Parametrisation of SMCMs | 105 |
| 6.2.1 | Factorising with Latent Variables | 106 |
| 6.2.2 | Probabilistic Inference | 108 |
| 6.2.3 | Causal Inference | 108 |
| 6.3 | Overview of the Chapter | 109 |
| 7 | Multi-Agent Causal Models | 111 |
| 7.1 | Definition | 111 |
| 7.1.1 | Additional Notations | 113 |
| 7.1.2 | Properties of MACMs | 113 |
| 7.1.3 | MACM and Cooperative MAS | 118 |
| 7.2 | Causal Inference in MACM | 119 |
| 7.2.1 | Calculating c-factors | 119 |
| 7.2.2 | Recursive Domain Reduction | 120 |
| 7.2.3 | Computation of $P(y do(x))$ | 122 |
| 7.2.4 | Example | 124 |
| 7.3 | Overview of the Chapter | 125 |
| 8 | Contributions on MACM | 127 |
| 8.1 | Separation of MACM | 127 |
| 8.1.1 | Factorization of the JPD | 127 |
| 8.1.2 | From SMCM to MACM | 129 |
| 8.1.3 | Alternative Causal Inference | 130 |
| 8.1.4 | Example | 131 |
| 8.1.5 | Critical Discussion on Separation of MACM | 132 |
| 8.2 | Learning MACMs | 132 |
| 8.2.1 | Related Approaches | 133 |
| 8.2.2 | Setting and Assumptions | 133 |
| 8.2.3 | Problems | 136 |
| 8.2.4 | Multi-Agent Causal Learning Algorithm | 137 |
| 8.2.5 | Critical Discussion of M-A CaDo | 142 |
| 8.3 | Validation of Assumptions | 143 |
| 8.3.1 | Intersection | 144 |
| 8.3.2 | Acyclicity | 144 |
| 8.4 | Overview of the Chapter | 145 |

| | |
|--|-----|
| 9 Conclusion | 147 |
| 9.1 Results | 147 |
| 9.1.1 Causal Bayesian Networks | 147 |
| 9.1.2 Causal Latent Networks | 148 |
| 9.1.3 Multi-Agent Causal Models | 150 |
| 9.2 Possible Applications | 151 |
| 9.3 Directions for Future Research | 152 |
| References | 155 |

List of Figures

| | | |
|------|---|----|
| 1.1 | Graphical representation of tsunami detection system. | 3 |
| 1.2 | Causal Bayesian Network representing a tsunami detection system. | 6 |
| 1.3 | Dynamic causal Bayesian network modeling feedback loop between taking aspirin and fever. | 6 |
| 1.4 | Tsunami detection system with explicit modeling of unmeasured influences. . . | 8 |
| 1.5 | Model of tsunami detection system with unmeasured influences. (a) The underlying DAG. (b) A maximal ancestral graph representation of (a). (c) A semi-Markovian causal model representation of (a). | 9 |
| 1.6 | Two tsunami detection systems forming a bigger network, allowing to issue a global warning. | 10 |
| 2.1 | A simple graph to indicate the different kinship relationships. | 16 |
| 2.2 | A Bayesian network representing a distribution which contains other independences then those entailed by the d-separation criterion. | 19 |
| 2.3 | The DAGs G_1, G_2 and G_3 belong to the same equivalence class, whereas G_4 does not. | 20 |
| 2.4 | Example of three equivalent networks and the respective representative of the Markov equivalence class. | 21 |
| 2.5 | Sketch of the $\lambda - \pi$ propagation algorithm for polytrees. | 22 |
| 2.6 | Image showing (a) original graph, (b) moralization, (c) triangulation, and (d) identifying the cliques. | 23 |
| 2.7 | A junction tree of the graph in Figure 2.6. | 24 |
| 2.8 | Forward equivalence search | 27 |
| 2.9 | Backward equivalence search | 28 |
| 2.10 | Example run of PC. | 31 |

| | | |
|-----|--|----|
| 3.1 | Conceptual sketch of how a CBN generates a JPD, that in its turn can be represented by several probabilistic BNs of which one is a CBN. | 34 |
| 3.2 | Two simple causal Bayesian networks. | 36 |
| 3.3 | CBN representation of tsunami warning system. | 38 |
| 3.4 | The CBN of the tsunami warning system of after disabling the detection buoys via an external intervention: $do(D=false)$ | 38 |
| 3.5 | When there is a correlation between X_i and X_j , one of these three causal structures hold. | 40 |
| 3.6 | A v-structure. | 41 |
| 4.1 | Main description of the MyCaDo approach. | 50 |
| 4.2 | Toy CBN (left) and the corresponding CPDAG (right) used in the example. | 55 |
| 4.3 | All possible instantiations for $X_i - Ne_U(X_i)$, the possible structures compatible with this instantiation and the result of inferring edges. | 57 |
| 4.4 | CPDAG generation, with an "oracle" providing the exact CPDAG from a known CBN. | 59 |
| 4.5 | Experimental data generation from a known CBN. | 60 |
| 4.6 | Plots of the number of experiments given the number of nodes in the graph, for relatively small graphs ($\#nodes \leq 30$) on the left, and bigger graphs ($30 \leq \#nodes \leq 50$) on the right. | 61 |
| 4.7 | Plots of the number of experiments given the maximum number of parents for any node, for relatively small graphs ($\#nodes \leq 30$) on the left, and bigger graphs ($30 \leq \#nodes \leq 50$) on the right. | 61 |
| 4.8 | General overview of the UnCaDo algorithm. | 63 |
| 4.9 | Simple example demonstrating the different steps in the UnCaDo algorithm. ... | 68 |
| 5.1 | (a) A problem domain represented by a causal DAG model with observable and latent variables. (b) A semi-Markovian causal model representation of (a). (c) A maximal ancestral graph representation of (a). | 76 |
| 5.2 | A SMCM with 5 c-components. | 78 |
| 5.3 | The SMCM of Figure 5.2 reduced to the observed variables $\mathbf{W} = \{X, X_2, X_4, X_6, Y\}$ and the associated unobserved variables $\{U_1, U_2\}$. If the original graph was denoted by G , we denote this graph by $G_{\mathbf{W}}$ | 80 |
| 5.4 | In the SMCM of Figure 5.2 the subsets of variables (a) \mathbf{S}_X (b) $Anc(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}}$ (c) $\mathbf{D}_X = Anc(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}} \cap \mathbf{S}_X$ are respectively depicted by rectangles. | 82 |
| 5.5 | A conceptual sketch of the identification algorithm. | 83 |

| | | |
|-----|---|-----|
| 5.6 | The SMCM of Figure 5.2 reduced to S_X , the c-component of X . Lemma 5.2 implies that $Q[X_4, X_6]$ can be calculated from $Q[X, X_4, X_6]$ | 86 |
| 5.7 | The SMCM of Figure 5.2 reduced to $\mathbf{H} = \{X, X_1, X_4, X_6\}$ | 88 |
| 6.1 | (a) A SMCM. (b) Result of FCI, with an i-false edge $X_3 \circ - o X_4$ | 98 |
| 6.2 | General overview of the MyCaDo++ algorithm. | 99 |
| 6.3 | (a) The result of FCI on data of the underlying DAG of Figure 5.1(a). (b) Result of an experiment on X_5 . (c) After experiment on X_4 . (d) After experiment on X_3 . (e) After experiment on X_2 while conditioning on X_3 during the statistical test. (f) After resolving all problems of Type 1 and 2. | 104 |
| 6.4 | (a) The PR-representation applied to the SMCM of Figure 5.1(b). (b) Junction tree representation of the DAG in (a). | 107 |
| 7.1 | Example of a multi-agent causal model of a product decision model. | 112 |
| 7.2 | The agents in this MACM are pairwise acyclic, but the combination of all three agents introduces the directed cycle $X_1, X_3, X_4, X_2, X_{14}, X_{11}, X_7, X_{10}, X_{12}, X_1$ | 115 |
| 7.3 | (a) A directed path between X_1 and X_2 in <i>agent2</i> is seen as a directed edge by <i>agent1</i> . (b) A path between X_1 and X_2 with a common cause in <i>agent2</i> is seen as a bi-directed edge by <i>agent1</i> . (c) A path between X_1 and X_2 that contains converging edges, is not seen by <i>agent1</i> as an edge of any type. | 115 |
| 8.1 | Example SMCM used to demonstrate decomposition into MACM. | 129 |
| 8.2 | The resulting separating MACM from the original SMCM given in Figure 8.1. | 130 |
| 8.3 | (a) The underlying SMCM of the domain, with the division per site. (b) The locally correct model that an <i>agent-i</i> would learn, based on $\mathbf{V}_{M_i} = \{X_1, X_2, X_3\}$ alone. It concludes that there is a directed edge between variables X_3 and X_2 | 134 |
| 8.4 | Example of the edges found when the global network (left) is divided into two agents (right). <i>Agent2</i> detects a <i>g</i> -false edge between X_2 and X_3 using only local information in his independence tests. | 137 |
| 8.5 | Both agents detect a <i>g</i> -false edge between X_4 and X_5 using only local information. | 138 |
| 8.6 | Problem with <i>g</i> -false edges concerning one variable in the intersection X_2 and one private variable of <i>agent2</i> , X_4 | 138 |
| 8.7 | (a) the valid multi-agent causal model, (b) the locally correct model of <i>agent1</i> , (c) the locally correct model of <i>agent2</i> | 140 |
| 8.8 | Global model and its subdivision over the different sites. | 142 |

| | | |
|------|--|-----|
| 8.9 | The correct local models from the global structure in Figure 8.8. | 143 |
| 8.10 | (a) A MACM (b)-(f) Different steps in checking whether the MACM in (a) is acyclic..... | 145 |

List of Tables

| | | |
|-----|--|-----|
| 4.1 | Table presenting the exact results of our experiments comparing the different criteria based on the number of nodes in the graph. The values indicate the number of times that particular criteria needed the least amount of experiments to completely orient the graph. The total amount of graphs for each number of nodes is 50. Sometimes different criteria result in the same amount of experiments so the total does not add up to 50. | 61 |
| 4.2 | Table presenting the exact results of our experiments comparing the different criteria based on the maximum number of parents a node can have in the graph. The values indicate the number of times that particular criteria needed the least amount of experiments to completely orient the graph. The total amount of graphs for each maximum number of parents is 100. Sometimes different criteria result in the same amount of experiments so the total does not add up to 100. | 62 |
| 6.1 | An overview of the different actions needed to complete edges of type $o \rightarrow$ | 101 |
| 6.2 | An overview of the different actions needed to complete edges of type $o-o$ | 102 |
| 6.3 | Removing inducing path edges. | 103 |

List of Algorithms

| | | |
|----|--|-----|
| 1 | GES Algorithm (Chickering, 2002b). | 26 |
| 2 | PC algorithm (Spirtes et al., 2000a). | 30 |
| 3 | Adaptive learning of CBN from observational and experimental data. | 56 |
| 4 | Adaptive learning of CBN for imperfect observational data and experiments. . . | 71 |
| 5 | Single agent <i>Identify</i> (C , F , $Q[\mathbf{F}]$) | 91 |
| 6 | Single agent identification of $P(\mathbf{t} do(x))$ | 92 |
| 7 | FCI algorithm (Spirtes et al., 2000a). | 94 |
| 8 | PR-representation algorithm. | 106 |
| 9 | Bi-agent <i>Identify</i> (C , T , $Q[\mathbf{T}]$) | 122 |
| 10 | Bi-agent identification of $P(y do(x))$ | 123 |
| 11 | M-A CaDo | 141 |

Introduction

This dissertation discusses the graphical modeling of causal relationships. As everyone has her/his own intuitive feel of what a causal relationship is, we will start this chapter by explaining the notion of causality as conceived in this dissertation. After that we will discuss our modeling requirements by means of an example and briefly introduce the techniques that will be presented in this dissertation. Next, we present our research proposition and an overview of the contributions. We end the chapter with a concise list of contents of the following chapters.

1.1 Preliminaries

In this section, we introduce our notion of causality and discuss the use of graphical models.

1.1.1 Causality

Everybody has her/his own intuitive idea about causality and when performing most of our actions we use the notion of causality because we are interested in the effects of those actions. Philosophers have spent centuries attempting to explain the notion of causality and even now many of them are still working on the subject, like Spirtes et al. (2000b), Pearl (2000), Salmon (1998), Russo and Williamson (2007) and many others.

One of the most common notions about causality is that a cause should precede its effect in time. Many theories of causation invoke an explicit requirement that a cause precedes its effect in time Pearl (2000). Yet temporal information alone cannot distinguish between genuine causation and spurious associations caused by unknown factors. For instance, the barometer drops before the rain falls, but it does not cause the rain. Furthermore, effects can appear at different time steps after the occurrence of their cause. For instance, it can take several months after being stung by a mosquito that you develop malaria symptoms while switching the light switch causes the light to turn on almost instantly. In this dissertation we

do not take into account an explicit notion of time. We assume that a causal connection is structural in the sense that it is an actual mechanism from nature. I.e. we do not distinguish between the two causal relations discussed before, but look at them as an ever present causal mechanism.

In this dissertation we use probability theory to characterize the relationship between a cause and its effect. There are several reasons for this approach, which we will demonstrate by means of some examples.

If we would consider only deterministic relations to be causal, then claims like "turning on the light switch causes the light to turn on", "being stung by an infected mosquito causes malaria" and "smoking causes cancer" do not hold. For instance, smoking only increases the chance of developing cancer. In order to be able to model these probabilistic relationships, we need to have a mathematical language that can accommodate them.

Another reason for using probabilities is that any causal expression in natural language is subject to exceptions. Consider for instance this example from Pearl (2000): Given these two plausible premises:

1. My neighbor's roof gets wet whenever mine does.
2. If I hose my roof it will get wet.

Taken literally, these two premises lead to the wrong conclusion, which is that my neighbor's roof would get wet whenever I hose mine. In order to resolve this, we would have to add all possible exceptions:

1. My neighbor's roof gets wet whenever mine does, except when I hose my roof, or it is covered with plastic, or ...

Probability theory allows us to focus on main issues of causality by incorporating all the exceptions into the probability distribution of the event.

1.1.2 Graphical Models

Graphical models denote a framework of modeling techniques that combine *probability theory* and *graph theory*, examples include *Bayesian networks* Pearl (1988), *possibilistic networks* Dubois (2006), *hidden Markov models* Rabiner (1989), etc.

Graphical models are specifically suited for handling complex problems that deal with uncertainty. Complexity is solved by organizing knowledge in a modular way while uncertainty is handled by making use of probability theory.

Furthermore, the graphical component of such models constitutes an intuitive model for researchers and a data structure that is suited for the development of efficient algorithms.

Since our goal is to model causality, we want to be able to denote cause and effect relationships. This can only be done in graphical models which contain directed edges. For instance, a Markov Random Field Kindermann and Snell (1980) is an undirected structure and therefore is not of interest to us.

1.2 Model Requirements

The goal of this dissertation is to give an integral view on modeling causal knowledge. We will use an example to illustrate the requirements which a good probabilistic causal model must fulfill.

Assume that we want to model a tsunami detection system. Tsunamis are caused by undersea earthquakes and can cause an enormous amount of damage, so early detection can potentially save many lives.

An undersea earthquake can be detected by seismic measuring equipment and if it causes a tsunami, then tide-sea-level instruments and tsunami detection buoys will measure a high fluctuation in their measurements. This information can then be forwarded to the tsunami warning centers¹. This system is depicted graphically in Figure 1.1.

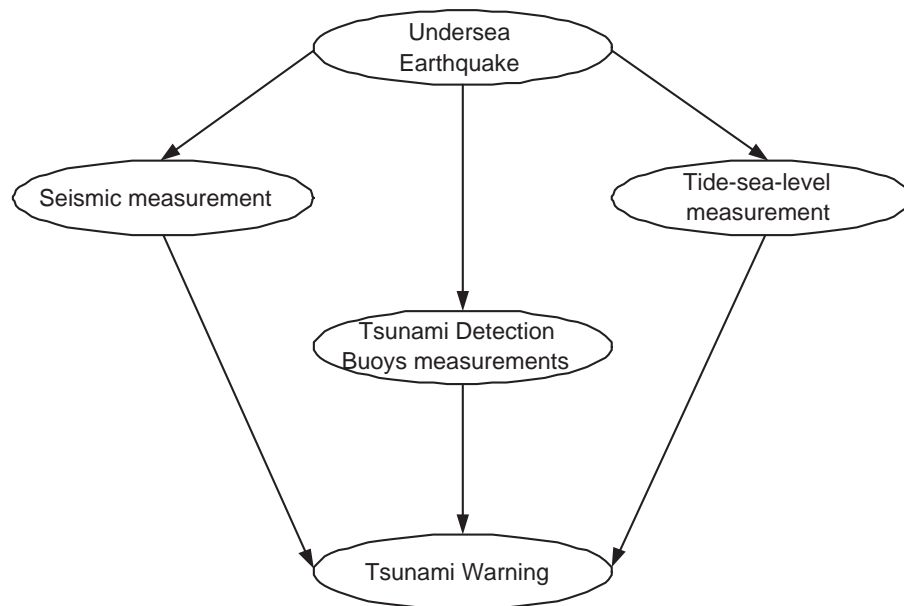


Figure 1.1. Graphical representation of tsunami detection system.

Depending on the strength and the origin of the earthquake, seismic activity detection equipment does not always register it. Furthermore, the tide-sea-level measuring instru-

¹ http://www.tsunami.noaa.gov/warning_system_works.html

ments and the measurements of the tsunami detection buoys can be influenced by weather conditions, passing boats, malfunctions, etc. This means that, as discussed previously, we need probabilities to model our detection system.

The main use of a tsunami detection system is to issue a warning when a tsunami is about to strike. However, when too many warnings are given for relatively harmless tsunamis, people in the surrounding areas lose confidence in the system.

The type of question we want to answer with the model is given a combination of the measurements what is the chance of issuing a tsunami warning. For instance, assume we measure seismic activity but no respective change in measurements for the tide-sea-level measurements and tsunami detection buoys. Solving these types of questions is called performing probabilistic inference, e.g. determining the probability of some event (tsunami warning) given that we have some observational knowledge (seismic activity).

Another interesting type of question explores what would happen if one of the measuring devices broke down. For instance if the measuring instruments inside the tsunami detection buoys stopped working, would warnings still be issued? Solving these questions is called performing causal inference, e.g. what is the probability of some event (tsunami warning) when there is an external intervention on the system (system failure of tsunami detection buoys). I.e. a system failure of the tsunami detection buoys corresponds to the tsunami detection buoys measurements having a certain value, but that value is independent of the presence of an earthquake.

One final question that arises is how such models are constructed. Often experts create a model based on their expertise on the subject. However, for a lot of systems there is no prior knowledge on the causal relationships that hold between the variables. For instance, in bioinformatics, an interesting task is to know which genes interact with each other and cause the over/under expression of other genes Bower and Bolouri (2001). To retrieve a causal model from data (either observational or experimental) is called learning. As discussed before we never take time into account explicitly, so we wish to retrieve causal knowledge from certain statistical patterns in the data, patterns that, in fact, can be given meaningful interpretation only in terms of causal directionality.

A good *probabilistic causal model* needs to be able to (a) model the system, (b) answer both probabilistic and causal inference questions, and (c) be learned from data.

1.3 Causal Bayesian Networks

Causal relationships have been modeled in many different ways like *structural equation modeling* (Haavelmo, 1995), *propensity scores* (Rosenbaum and Rubin, 1983) and *graphical models*

(Pearl, 2000), (Spirtes et al., 2000a). We work on the latter as this is a very powerful and also popular and intuitive representation. For now, we make the assumption that all variables that are of importance to the system are measured and observed and that all data is present at a central site.

Bayesian networks (BNs) are the cornerstones of all causal models used in this dissertation. Though they were originally introduced to concisely represent a joint probability distribution, they have been extended to model causal relationships (Pearl, 2000).

BNs consist of a graphical structure G in the form of a directed acyclic graph (DAG), and a set of parameters θ corresponding to conditional probability distributions of each variable X_i given its parents in the graph G .

Causal Bayesian networks (CBNs) are Bayesian networks with the added property that the edges connecting variables represent a direct *cause-effect* relationship. More specifically, a directed edge $C \rightarrow E$ in a causal Bayesian network indicates that there exists at least one intervention on C that would alter the distribution of the values of E given that all other variables are kept at certain constant values.

An example causal Bayesian network, representing the tsunami detection system, is given in Figure 1.2. There are 5 variables in the network: *Undersea Earthquake* (U), *Seismic Measurement* (S), *Tide – sea – level Measurement* (L), *Tsunami Detection Buoys Measurement* (D) and *Tsunami Warning* (W).

The network structure corresponds to the causal links in the tsunami detection system and the (conditional) probabilities are the parameters of the network.

A CBN can only represent systems in which there are no feedback-loops since its underlying structure is a DAG. It is possible to extend CBN to incorporate feedback loops by explicitly modeling time. For instance we can model the following situation:

- I have fever thus I take aspirin.
- Aspirin diminishes the fever.

by using a so called *dynamic causal Bayesian network* where we differentiate between two types of dependencies and causal relationships. The first is within 1 time step t : from the moment I feel fever I take aspirin: $fever(t) \rightarrow aspirin(t)$. The second is between two time steps t and $t + 1$: Taking the aspirin diminishes the fever: $aspirin(t) \rightarrow fever(t + 1)$. This can be graphically modeled as shown in Figure 1.3. In all models we use in this dissertation we do not model time explicitly, so we only take into account relationships that hold within one time step.

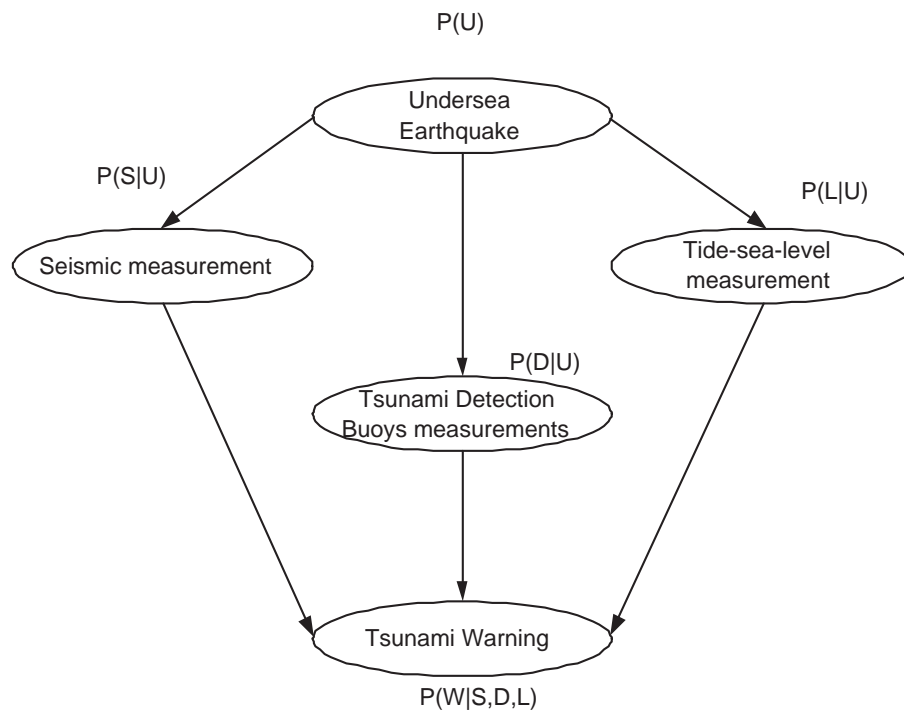


Figure 1.2. Causal Bayesian Network representing a tsunami detection system.

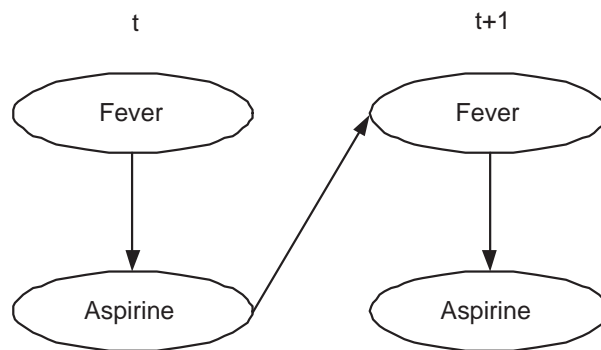


Figure 1.3. Dynamic causal Bayesian network modeling feedback loop between taking aspirin and fever.

1.3.1 Probabilistic inference

We can use a CBN to answer several types of queries. For instance, using the network given in Figure 1.2, what is the probability that there will be a tsunami warning given that there is seismic activity measured but there was no change in measurements in the tsunami detection buoys, $P(W = true|S = true, D = false)$.

1.3.2 Causal Inference

As mentioned before we can also use CBNs to answer causal queries. We denote causal queries by using the $do()$ operator (cf. Section 3.2.1), for instance, if we want to know the probability that there will be a tsunami warning given that the tsunami detection buoys are defect and always issue a high measurement ($D = true$):

$$P(W = true|do(D = true))$$

1.3.3 Learning

Now that we have introduced the models that we will use throughout this dissertation, we want to discuss how these models can actually be learned. Learning causal models means determining the structure of the causal network and the parameters from data. It is in the task of structure learning that most of our contributions lie.

In order to learn the parameters of a CBN, it is assumed that the structure is known. Hence, the task at hand is to recover the probabilistic dependences between a variable and its parents in the graph. Methods for learning the parameters of a CBN are given in Section 2.5.1.

Structure learning amounts to finding the correct cause-effect relationships between the variables of the system that is studied. There are two main approaches on learning the structure of a CBN. One is based on the probabilistic dependences that hold among the variables and the other is based on how good the structure explains the available data on the system. Different learning techniques and algorithms are discussed in Section 2.5.2.

1.4 More Expressive Models

The previous sections allow us to fulfill our tasks in a centralized environment where all variables that are of importance to the system are known. These restrictions put limits on the representational power of our models. In this section, we introduce richer models. First, we remove the requirement that all variables that influence the system are known, this means that we allow *latent variables*. Then, we look at settings in which the data and the system are dispersed across multiple sites.

1.4.1 Latent Variable Modeling

Consider again our tsunami detection system given in Figure 1.1. In this model it is assumed that there are no external influences that can alter the system. This is of course a

naive assumption; in actuality there can be many other actors that influence the behavior of the components of the tsunami detection system. For instance in Figure 1.4 we show how *Volcanic Eruption* (V) or extreme *Weather Conditions* (C) can influence the system.

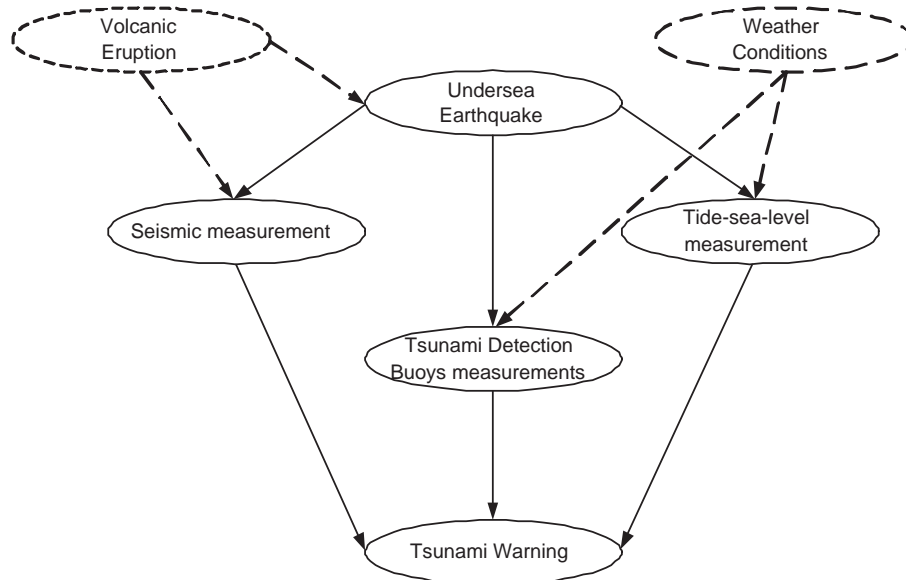


Figure 1.4. Tsunami detection system with explicit modeling of unmeasured influences.

However, these two variables are not measured; variables that influence the system but are never measured are called *latent variables*. When we use causal Bayesian networks (CBNs), we have to explicitly add V and C to the model and have to provide CPDs for variables depending on them. Since they are never measured, this system can not be modeled by CBNs.

The graph with all latent variables explicitly modeled is called the *underlying DAG*.

There are two major paradigms used to model causal relationships in the presence of latent variables, namely *ancestral graphs* (AG) (Richardson and Spirtes, 2002) and *semi-Markovian causal models* (SMCM) (Pearl, 2000), which both implicitly model the latent variables. We call both AG and SMCM *causal latent models*.

In a SMCM, each directed edge represents an immediate autonomous causal relation between the corresponding variables, just as was the case for causal Bayesian networks. A bi-directed edge between two variables represents a latent variable that is a common cause of these two variables. There exist algorithms to perform causal inference in SMCMs, however there was no algorithm to learn the structure. One of the contributions of this dissertation is to propose such an algorithm.

In AGs a directed edge represents an ancestral relation in the underlying DAG. I.e. an edge from variable C to E represents that in the underlying causal DAG with latent variables, there is a directed path between C and E . Bi-directed edges represent a *latent common cause* between the variables. However, if there is a latent common cause between two variables C and E , and there is also a directed path between C and E in the underlying DAG, then in the AG the ancestral relation takes precedence and a directed edge will be found between the variables. Algorithms to learn the structure of an AG from observational data have been proposed by (Spirtes et al., 2000a).

A problem with this representation is that these semantics of edges make some causal inferences in AGs impossible in general. In this dissertation we show how to transform an AG into a SMCM so that inference can be performed after the transformation.

In Figure 1.5 we show an example of an underlying DAG, and the corresponding AG and SMCM.

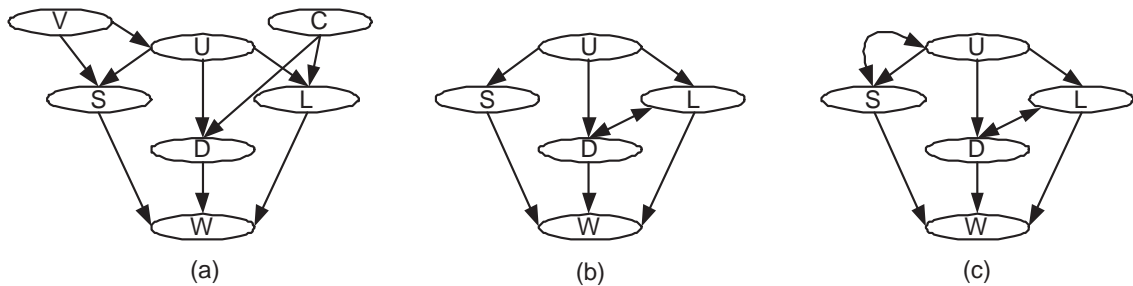


Figure 1.5. Model of tsunami detection system with unmeasured influences. (a) The underlying DAG. (b) A maximal ancestral graph representation of (a). (c) A semi-Markovian causal model representation of (a).

AGs and SMCMs are introduced in Sections 5.1 and 5.2. Inference in these models are discussed in Section 5.3 and learning in Section 5.4.

1.4.2 Multi-Agent Causal Models

Depending on the magnitude of the earthquake, a tsunami can spread over more than 100 miles. This means that several detection systems in different parts of the world may be activated. This also means that some data gathered on the possibility of a tsunami is dispersed over different centra in the world.

Hence, the tsunami detection system is part of a much bigger network consisting of several local detection systems that interact, see Figure 1.6 for an example with two local systems.

A *multi-agent causal model (MACM)* is a technique that is able to model these type of systems. It is assumed that there is no longer one central controller having access to all the

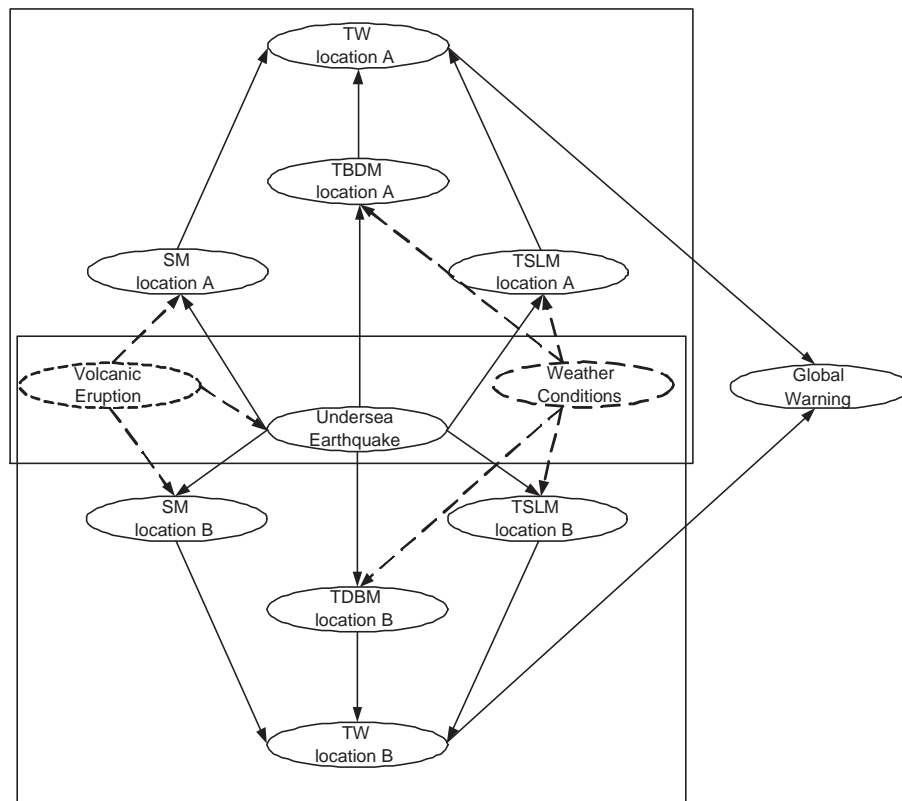


Figure 1.6. Two tsunami detection systems forming a bigger network, allowing to issue a global warning.

observable variables, but instead a collection of agents each having access to non-disjoint subsets of all the variables.

In this dissertation we will discuss how a multi-agent causal model can be learned from data. This is part of joint work with *Sam Maes* and has previously been discussed in his dissertation (Maes, 2005). We will also propose a new causal inference algorithm that allows to calculate the effect of an intervention on all other variables in the system by communicating local distributions.

MACMs are introduced in Section 7.1, inference in MACM in Section 7.2, and learning in Section 8.2.

1.5 Research Proposition

We have shown in the previous discussion that modeling causality amounts to solving a number of tasks:

1. Identify the modeling technique, which defines the complexity of the systems we can model.
2. Identify the queries we wish to answer with our models.

3. Provide algorithms to solve the queries.
4. Provide algorithms to learn the models from data.

The goal of this dissertation is to integrate all these tasks and our research proposition can be described as:

Proposition 1.1. *To provide an integral approach to modeling causal relationships, from learning to inference.*

1.5.1 Overview and Contributions

The build-up of this dissertation is as follows.

In Chapter 2 we start with a concise introduction on probability theory and Bayesian networks.

The rest of the dissertation consists every time of a chapter introducing the modeling technique and discussing its respective state-of-the-art followed by a chapter with our contributions on these models.

We introduce our simplest causal model, namely a causal Bayesian network, in Chapter 3. We demonstrate how it can be learned and used as an inference tool.

Next we propose our contributions for CBN in Chapter 4:

- We introduce a structure learning algorithm, *MyCaDo*, that attempts to minimize the overall cost of learning the structure of CBNs. This is done by using a decision theoretic approach, generic for any decision criteria (Meganck et al., 2006a).
- We describe an algorithm for learning the structure of a CBN which does not make the assumption that the data corresponds exactly to the underlying system, called *UnCaDo*. We introduce an adapted independence test proposed for this goal (Meganck et al., 2008b).

In Chapter 5 we introduce the two main techniques for latent causal modeling, namely semi-Markovian causal models and ancestral graphs and identify the limitations of these techniques.

Chapter 6 contains our contributions on latent variable modeling:

- We introduce an algorithm, *MyCaDo++*, to learn the structure of a semi-Markovian causal model from a mixture of observational and experimental data (Meganck et al., 2006b),(Maes et al., 2007a).
- We propose a transformation that maps a semi-Markovian causal model to a directed acyclic graph in which it is possible to learn the parameters and perform probabilistic inference (Maes et al., 2007a), (Meganck et al., 2007).

We introduce the framework of multi-agent causal models and discuss its properties in Chapter 7. We present joined work with Sam Maes for performing causal inference in this setting (Maes et al., 2004),(Maes et al., 2005a),(Maes et al., 2005b),(Maes et al., 2007b).

Chapter 8 discusses our additions to the framework:

- We extend the MACM approach to allow agents to be coupled in a complex network and propose a new causal inference algorithm for this more complex setting (Meganck et al., 2008a).
- We introduce an algorithm called *M – ACaDo* to learn the structure of a MACM from distributed data while trying to minimize the communication between agents (Meganck et al., 2005a),(Meganck et al., 2005b).

Bayesian Networks

In this chapter we will introduce the concept of Bayesian Networks (BNs). They will be the basis for all our graphical causal models in the remaining chapters. BNs, just as other types of probabilistic graphical models are, as the name suggests, a marriage between two theories: probability theory and graph theory. Henceforth, we will commence with a concise introduction on both theories and then define BNs. The remaining sections of this chapter will focus on the different ways BNs can be used. We end with an overview of the discussed topics in this chapter.

2.1 Probability Theory

In this work uppercase letters X_i, X_j, \dots are used to represent discrete random variables and bold uppercase letters $\mathbf{V}, \mathbf{W}, \dots$ are sets of variables. Corresponding lowercase letters are used to represent their instantiations, e.g. x_i, x_j, \dots and $\mathbf{v}, \mathbf{w}, \dots$. $P(X_i = x_i)$ is used to denote the probability that a variable X_i has value x_i and likewise $P(\mathbf{V} = \mathbf{v})$ the probability that a set of variables \mathbf{V} has the corresponding instantiation \mathbf{v} . Usually, $P(x_i)$ and $P(\mathbf{v})$ are used as abbreviations of $P(X_i = x_i)$ and $P(\mathbf{V} = \mathbf{v})$ respectively.

2.1.1 Probability Distributions

The *joint probability distribution* (JPD) $P(\mathbf{V})=P(X_1, \dots, X_n)$ specifies probabilities to all possible joint instantiations of the variables in \mathbf{V} .

A *conditional probability distribution* (CPD) $P(X_i|X_j)$ is the probability of event X_i given that we know that X_j has occurred.

Using conditional probabilities the JPD $P(X_1, \dots, X_n)$ can be rewritten as follows:

$$P(X_1, \dots, X_n) = P(X_n|X_{n-1}, \dots, X_1)P(X_{n-1}, \dots, X_1)$$

When we iteratively repeat this process by substituting the marginal probability in the right hand side we get:

$$\begin{aligned}
P(X_1, \dots, X_n) &= P(X_n | X_{n-1}, \dots, X_1) P(X_{n-1} | X_{n-2}, \dots, X_1) \dots P(X_2 | X_1) P(X_1) \\
&= \prod_{i=1}^n P(X_i | X_{i-1}, \dots, X_1)
\end{aligned} \tag{2.1}$$

The repeated application of this process is called the *chain rule*.

2.1.2 Independence

Two random variables X_i and X_j are *independent* if:

$$P(X_i, X_j) = P(X_i)P(X_j).$$

Such an independence is denoted as:

$$(X_i \perp\!\!\!\perp X_j)$$

and it means that the state of X_i is irrelevant to the state of X_j and vice versa.

Similarly, two variables X_i and X_j are *conditionally independent* given some other set of variables $\mathbf{S} \subset (\mathbf{V} \setminus \{X_i, X_j\})$, if:

$$P(X_i, X_j | \mathbf{S}) = P(X_i | \mathbf{S})P(X_j | \mathbf{S})$$

given that $P(\mathbf{s}) > 0$ for all $\mathbf{s} \in D_{\mathbf{S}}$, where $D_{\mathbf{S}}$ is the domain of \mathbf{S} . This is equivalent to $P(X_i | X_j, \mathbf{S}) = P(X_i | \mathbf{S})$ or $P(X_j | X_i, \mathbf{S}) = P(X_j | \mathbf{S})$. This means that if the joint state of the variables in set \mathbf{S} is known, X_i has no extra information on X_j and vice versa. Such a conditional independence is denoted as:

$$(X_i \perp\!\!\!\perp X_j | \mathbf{S})$$

If two variables are not independent then they are said to be *dependent*. Again, one can distinguish between a marginal dependence of two variables X_i and X_j , denoted as:

$$(X_i \not\perp\!\!\!\perp X_j)$$

and a dependence conditional on a set \mathbf{S} denoted as:

$$(X_i \not\perp\!\!\!\perp X_j | \mathbf{S})$$

Concise Representation of JPD

Assume that the following conditional independence holds:

$$(X_i \perp\!\!\!\perp \mathbf{S}_i | X_{i-1}) \text{ with } \mathbf{S}_i = \{X_{i-2}, \dots, X_1\}$$

or X_i is independent of \mathbf{S}_i conditional on X_{i-1} . Following the definition of conditional independence this implies:

$$P(X_i|X_{i-1}, X_{i-2}, \dots, X_1) = P(X_i|X_{i-1}).$$

Now we can replace the factor $P(X_i|X_{i-1}, X_{i-2}, \dots, X_1)$ in Equation (2.1) by $P(X_i|X_{i-1})$. This leads to a much smaller representation of the JPD. Even in the case of binary variables this representation already needs to store 2^{i-2} times less numbers than the original JPD.

We can generalize, assuming that there is a set of variables \mathbf{T}_i that makes another set \mathbf{S}_i independent of X_i with $\mathbf{T}_i \cup \mathbf{S}_i = \{X_1, \dots, X_{i-1}\}$ and $\mathbf{T}_i \cap \mathbf{S}_i = \emptyset$, we have:

$$(X_i \perp\!\!\!\perp \mathbf{S}_i | \mathbf{T}_i)$$

This implies

$$P(X_i|X_{i-1}, \dots, X_1) = P(X_i|\mathbf{T}_i).$$

and hence we can rewrite the chain rule in Equation (2.1) as:

$$P(\mathbf{V}) = \prod_{i=1}^n P(X_i|\mathbf{T}_i).$$

2.2 Graph Theory

A graph consists of a set of nodes $\mathbf{V} = \{X_1, \dots, X_n\}$ and a set of edges $\mathbf{E} \in V * V$ between these nodes. Edges can be undirected $-$, directed \leftarrow, \rightarrow or bi-directed \leftrightarrow .

To refer to elements in a graph we use the terms of kinship. For instance in Figure 2.1 the following relationships hold:

- X_1 and X_2 are parents of X_3
- X_3 is the child of X_1 and X_2
- X_1, X_2 and X_3 are ancestors of X_4 and X_5
- X_4 is a descendant of X_1, X_2 and X_3

We denote Pa_i (or $Pa(X_i)$), $Ch(X_i)$, $Anc(X_i)$ and $Desc(X_i)$ as the parents, children, ancestors and descendants of X_i respectively.

A *path* between X_i and X_j is a sequence of edges connecting X_i and X_j irrespective of their orientations. A *directed path* from X_i to X_j is a sequence of edges $X_i \rightarrow X_{p_1} \rightarrow \dots \rightarrow X_{p_k} \rightarrow X_j$ such that in the graph there is an edge $X_{p_i} \rightarrow X_{p_{i+1}}$ for all $i = 1, \dots, k-1$. A directed acyclic graph (DAG) is a graph containing only directed edges in which *directed cycles*, i.e. a directed path which begins and ends at the same variable, are not allowed.

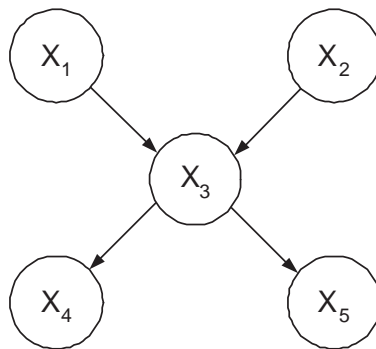


Figure 2.1. A simple graph to indicate the different kinship relationships.

Graphs can be used to represent many types of systems depending on the semantic interpretation that is given to the nodes and the edges. For instance, we can consider each node to be a city and each edge to depict a highway connecting two cities. In the next section we will give an interpretation for a graph that allows it to be used as an independence model of a JPD.

2.3 Bayesian Networks

In this section we introduce Bayesian networks¹, which are the cornerstones of our causal graphical models.

A BN consists of two components, a DAG and a set of parameters θ representing conditional probability distribution (CPDs). Together they model a JPD over a set of variables in a concise way. Furthermore they allow to efficiently calculate probabilistic queries of the form $P(X_i|X_j)$.

2.3.1 Definition

In (Pearl, 1988), (Russell and Norvig, 1995) BNs are defined as follows:

Definition 2.1. A *Bayesian network* is a triple $\langle V, G, P(X_i|\Pi_i) \rangle$, with:

- $V = \{X_1, \dots, X_n\}$, a set of observable discrete random variables
- a directed acyclic graph (DAG) G , where each node represents a variable from V
- parameters: conditional probability distributions (CPD) $P(X_i|\Pi_i)$ of each variable X_i from V conditional on its parents Π_i in the graph G .

¹ Other names found in the literature are (Bayesian) belief networks, probabilistic networks, graphical models, ..., sometimes they are even wrongfully referred to as a *causal graphs*

An example BN is given in Figure 1.2 where the CPDs are indicated next to the nodes².

We will now introduce some properties that we assume to hold when working with BNs.

2.3.2 Causal Sufficiency

The assumption of *Causal Sufficiency* is satisfied if we have measured all the common causes of the measured variables that we include in our model. This means there are no unmeasured variables influencing more than one observed variable, i.e. there are no *latent* variables.

2.3.3 Markov Condition

Intuitively, the *Markov condition* says that the state of a variable depends only on the state taken by variables in its vicinity. This is an assumption that is made in many areas of research that are otherwise intractable. For example, in the area of reinforcement learning or in the analysis of time series, it is often assumed that only the current state of the world has an influence on future steps of the world, independent of previous states of the world.

In (Spirtes et al., 2000a) the Markov Condition is defined as follows:

Definition 2.2. A DAG G over variables \mathbf{V} and a probability distribution $P(\mathbf{V})$ satisfy the **Markov condition** if and only if for every variable $X_i \in \mathbf{V}$, X_i is independent of any of its non-descendants given its immediate parents in the graph, or

$$X_i \perp\!\!\!\perp \mathbf{V} \setminus (\text{Desc}(X_i) \cup \Pi_i) \mid \Pi_i. \quad (2.2)$$

In other words, the parents of a variable X_i , screen X_i off from all other variables, except for the descendants of X_i .

d-separation

The *d-separation* criterion is a simple graphical criterion that allows to retrieve all the independences entailed by the Markov condition in a graph. In (Pearl, 2000) it is defined as follows:

Definition 2.3. A path p is said to be *d-separated* by a set of node \mathbf{Z} if and only if:

- p contains a chain $X_i \rightarrow X_m \rightarrow X_j$ or a fork $X_i \leftarrow X_m \rightarrow X_j$ such that the middle node X_m is in \mathbf{Z} , or
- p contains an inverted fork $X_i \rightarrow X_m \leftarrow X_j$ such that the middle node X_m is not in \mathbf{Z} and no descendant of X_m is in \mathbf{Z} .

² This example is also a causal Bayesian network, see Chapter 3 and later.

A set \mathbf{Z} is said to *d-separate* a set \mathbf{W}_1 from \mathbf{W}_2 if and only if \mathbf{Z} blocks every path from a node in \mathbf{W}_1 to a node in \mathbf{W}_2 .

The connection between *d*-separation and conditional independence is established through the following theorem (Pearl, 2000):

Theorem 2.1. *If sets \mathbf{W}_1 and \mathbf{W}_2 are d-separated by \mathbf{Z} in a DAG G that satisfied the Markov condition, then \mathbf{W}_1 is independent of \mathbf{W}_2 conditional on \mathbf{Z} .*

If two sets of variables \mathbf{W}_1 and \mathbf{W}_2 are *d-separated* given some set \mathbf{Z} we denote this as $\mathbf{W}_1 \perp \mathbf{W}_2 | \mathbf{Z}$.

We define the notion of an *I – map* or *independence map*.

Definition 2.4. *A DAG G is an I – Map of a distribution P if all independences implied by applying the Markov property to G are satisfied by P . I.e. $\mathbf{W}_1 \perp \mathbf{W}_2 | \mathbf{Z} \Rightarrow \mathbf{W}_1 \perp\!\!\!\perp \mathbf{W}_2 | \mathbf{Z}$ for all $\mathbf{W}_1, \mathbf{W}_2, \mathbf{Z}$ in G .*

2.3.4 Faithfulness

The Markov condition entails independence relations that hold among the variables in the graph. In general, it is possible that a distribution on a graph G that satisfies the Markov condition may include other independence relations not entailed by the Markov condition. For instance, consider the DAG of Figure 2.2. The *d*-separation criterion implies that $(X_i \perp\!\!\!\perp X_l | \{X_j, X_k\})$, but in this JPD X_i is also marginally independent from X_l , or, $(X_i \perp\!\!\!\perp X_l)$. We can see that this holds as $P(X_i, X_l) = \{0.23, 0.24, 0.26, 0.27\}$, $P(X_i) = \{0.47, 0.53\}$, and $P(X_l) = \{0.49, 0.51\}$, then we can check that $P(X_i, X_l) = P(X_i)P(X_l)$. So what we have here is that the relations between two variables along two different directed paths exactly cancel each other out.

When something like this occurs, we say that the distribution is *unfaithful* to the DAG. This leads us to the following definition (Spirtes et al., 2000a):

Definition 2.5. *A joint probability distribution P and a DAG G with variables \mathbf{V} are **faithful to one another** if all and only the conditional independence relations true in P are entailed by the *d*-separation criterion applied to G . Formally, for any disjoint subset $\mathbf{W}_1, \mathbf{W}_2$ and \mathbf{Z} of \mathbf{V}*

$$\mathbf{W}_1 \perp\!\!\!\perp \mathbf{W}_2 | \mathbf{Z} \Leftrightarrow \mathbf{W}_1 \perp \mathbf{W}_2 | \mathbf{Z}. \quad (2.3)$$

Moreover, a distribution is **faithful** provided there is some DAG to which it is faithful.

We assume that the distributions we model are faithful, as those distributions that are not faithful to a DAG, cannot be reliably modeled with the current techniques.

Note that in an unfaithful distribution there are dependences between some variables, but that nature “conspires” to make these variables look independent to an observer. This is due to different dependences between these variables exactly canceling each other out.

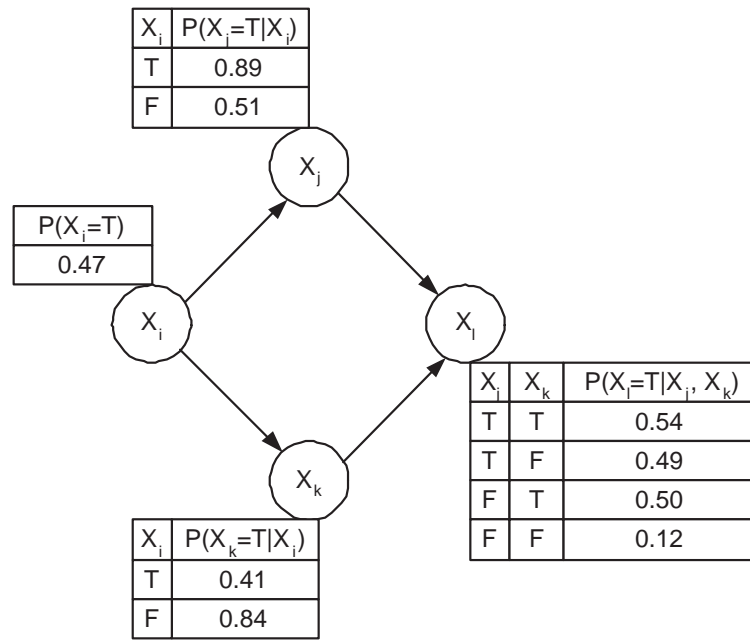


Figure 2.2. A Bayesian network representing a distribution which contains other independences than those entailed by the d -separation criterion.

2.3.5 Factorization of the JPD

The Markov condition implies that the CPDs of a BN represent a factorization of the joint probability distribution as a product of conditional probability distributions of each variable given its parents in the graph:

$$P(\mathbf{V}) = \prod_{X_i \in \mathbf{V}} P(X_i | \Pi_i) \quad (2.4)$$

If we compare this factorization to that of equation (2.1), we see that the JPD can be represented by a much smaller amount of parameters than the full JPD or via the chain rule (Equation 2.1). It is by replacing $P(X_i | X_{i-1}, \dots, X_1)$ by $P(X_i | \Pi_i)$ in the factorization via the chain rule that we obtain a significant advantage, when the amount of parents of each variable is small, or $\#\Pi_i \ll i - 1$.

Furthermore, as in such a case each CPD typically involves only a small number of variables, the parameters of a BN can be estimated from data with increased reliability.

2.3.6 Markov Equivalence

We saw above that a DAG specifies a unique set of probability distributions that must exhibit those conditional independences found via the d -separation criterion. The opposite is not

true, the conditional independences and dependences implied by a probability distribution do not necessarily determine a unique DAG.

For example, the DAGs G_1 , G_2 and G_3 in Figure 2.3 all exhibit the same d -separations, i.e. $X_i \perp X_j | X_m$ and $X_i \nsim X_j$. This means they represent the same set of probability distributions. We can also see this when we write down the factorization of the JPD as in equation (2.4) for each of these DAGs.

$$P(X_i, X_j, X_m) = P(X_i | X_m) P(X_m | X_j) P(X_j) = P(X_i | X_m) P(X_m) P(X_j | X_m) = P(X_m | X_i) P(X_j | X_m) P(X_i)$$

G_1 G_2 G_3

The DAG G_4 contains the same edges as the other DAGs, but it represents other independences and dependences. More specifically, there is a v -structure at X_m ($X_i \rightarrow X_m \leftarrow X_j$), and thus the d -separations are $X_i \perp X_j$ and $X_i \nsim X_j | X_m$.

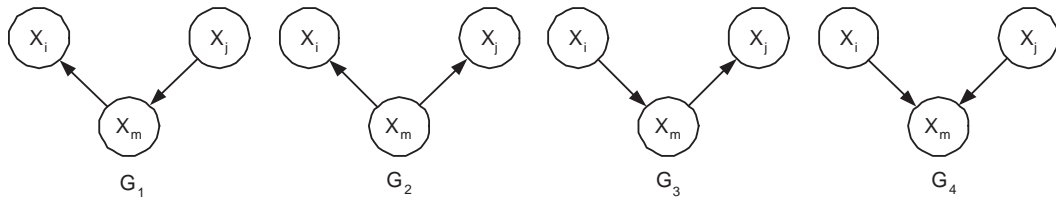


Figure 2.3. The DAGs G_1 , G_2 and G_3 belong to the same equivalence class, whereas G_4 does not.

Property 2.1. The d -separation criterion defines a reflexive, symmetric and transitive relation on the set of DAGs called the **Markov equivalence** relation. It partitions the space of DAGs into equivalence classes, whose members exhibit the same d -separation properties.

Practically, two DAGs are equivalent if they have the same *skeleton* (i.e. the undirected graph obtained by dropping the orientation of all edges in a DAG) and the same v -structures (Verma and Pearl, 1990).

There is a graphical model that represents the class of Markov equivalent graphs:

Definition 2.6. A *completed partially directed acyclic graph (CPDAG)* is a representative of all the graphs that belong to the same Markov equivalence class. A CPDAG has the same skeleton as all the graphs in the equivalence class. It consists of directed and undirected edges. An edge is directed in a CPDAG if it is directed the same way in all graphs in the Markov equivalence class and undirected if some graphs in the equivalence class differ on the direction of the edge.

For example, in Figure 2.4(a,b,c) we show three equivalent graphs and their representative in Figure 2.4(d) which is the respective CPDAG.

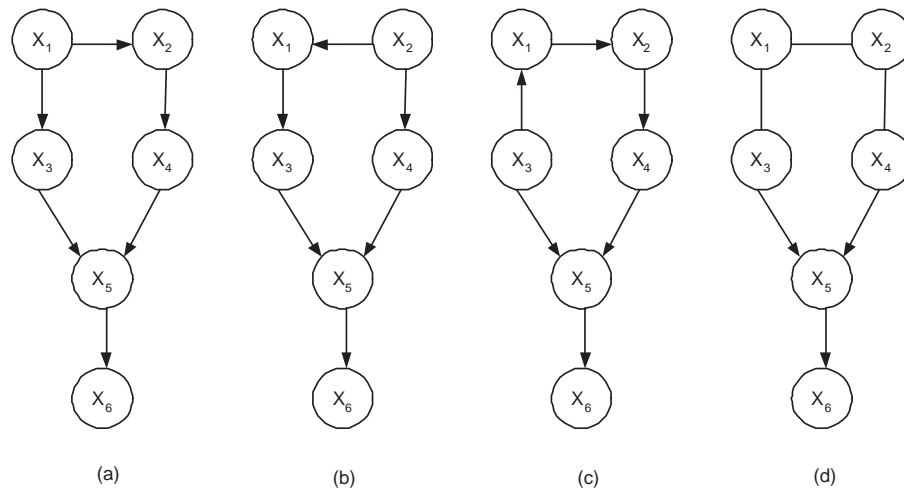


Figure 2.4. Example of three equivalent networks and the respective representative of the Markov equivalence class.

An important consequence of the existence of equivalence classes is that in general when learning the graphical component of a BN from a probability distribution alone, it is not a DAG that is obtained, but rather an equivalence class. In general, when a DAG is needed, a member of the equivalence class is chosen arbitrarily. It is only if additional knowledge is provided, such as the orientation of some edges, or if experiments are conducted, that we can differ between members in the same equivalence class and that a complete DAG can be learned.

2.4 Probabilistic Inference

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed event. For instance, using the BN in Figure 1.2, what is the probability that there is an undersea earthquake U given that the detection buoys made an alarming measurement $P(U = true|D = true)$.

In this section we will discuss some algorithms for performing this task using BNs.

2.4.1 $\lambda - \pi$ Message Passing

In (Pearl, 1988), Pearl introduced the $\lambda - \pi$ propagation algorithm for BNs with a polytree as a graph. A polytree is a DAG where undirected cycles are not allowed, or, in other words that there is only one path between any two nodes in the graph.

When evidence for some variable is received in the form of knowledge concerning its value, messages are calculated at the node representing that variable in the graph. For every child of the node there is a π message and for every parent a λ message. The only information

that is used to obtain the messages is local to that variable X_i , i.e. the CPD of X_i given its parents in the graph, or $P(x_i|\pi_i)$, combined with the evidence that was received.

Then the messages are sent to the respective neighbors (parents and children) of variable X_i . When these nodes have received the messages from all their parents (or all their children), they in their turn calculate messages for their children (or their parents) and so on. Note that root nodes and leaf nodes respectively cannot receive π and λ messages, and thus they start to propagate immediately.

Finally, when a variable X_i has received the messages from all its neighbors, it can combine all these messages to calculate its updated belief. See Figure 2.5 for a sketch of what happens in variable V .

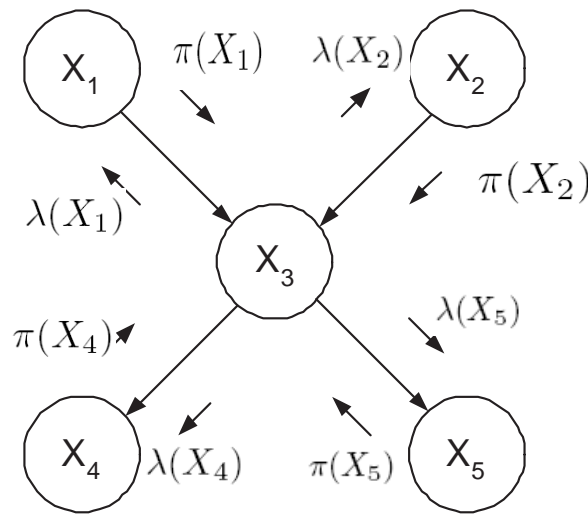


Figure 2.5. Sketch of the $\lambda - \pi$ propagation algorithm for polytrees.

The propagation converges in time proportional to the diameter of the graph and the work done in a node is proportional to the size of the CPD present at that node. Hence the belief propagation algorithm is linear in the number of network parameters.

2.4.2 Junction Tree Algorithm

As opposed to the belief propagation from above, this technique can handle inference in BNs whose graph contains undirected cycles (Jensen, 1996). It essentially consists of merging highly-interconnected subsets of nodes into supernodes or cliques.

The transformation of a BN into a junction tree consists of three phases: moralization, triangulation and clique building. In the first phase the DAG is transformed into an undirected graph adding edges between the parents nodes in each v -structure and then dropping

the directionality of the edges. Triangulation aims at removing cycles of length > 3 with no chord (i.e. an edge between two non-adjacent nodes in such a cycle), by adding a chord.

Then, build the clique graph by first identifying cliques (i.e. maximal complete sub-graphs) in the moralized and triangulated graph. The clique graph consists of nodes that each represents a clique. These nodes are connected with an edge if the set of variables contained in two cliques intersect. See Figure 2.6 for an example of these steps.

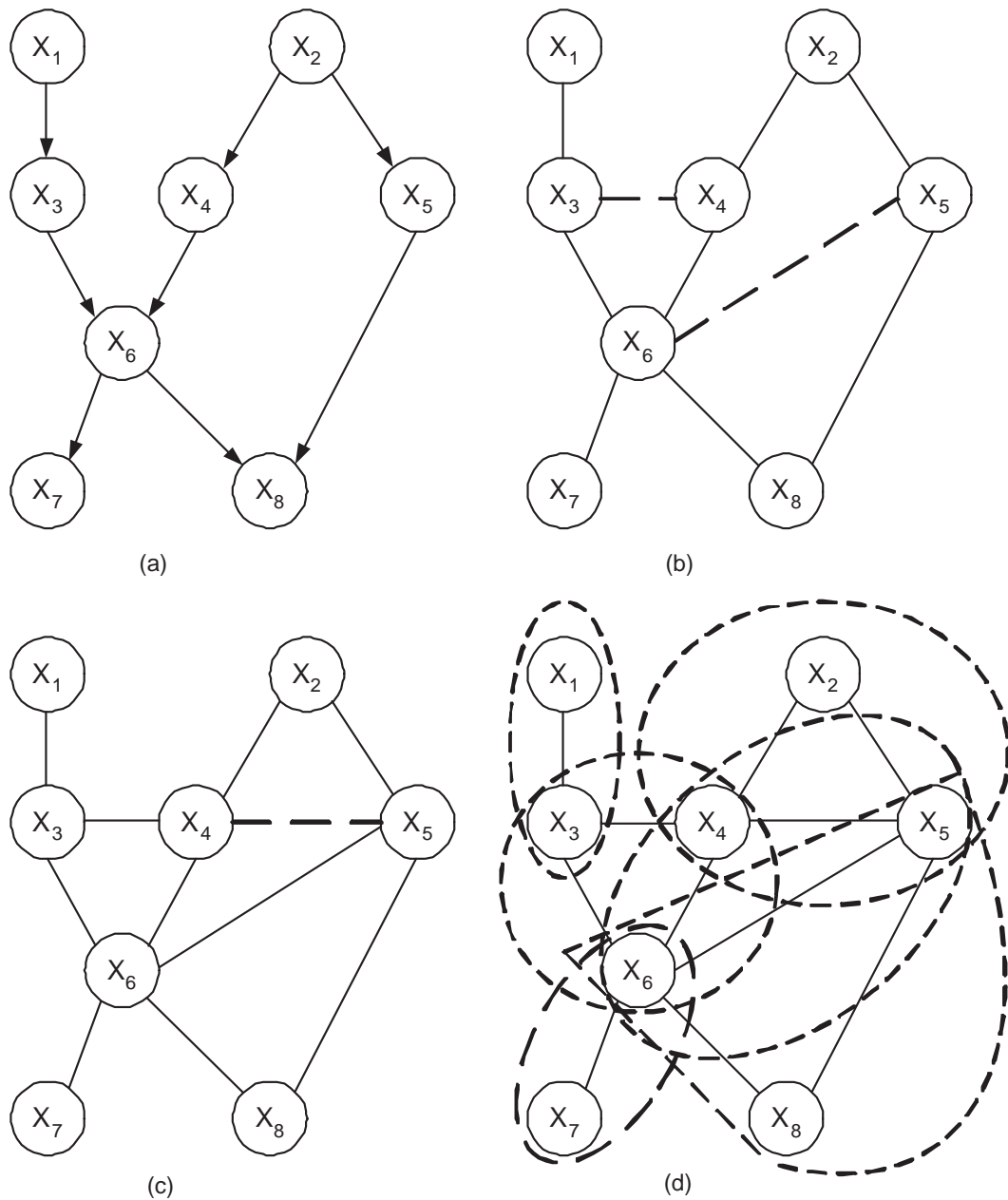


Figure 2.6. Image showing (a) original graph, (b) moralization, (c) triangulation, and (d) identifying the cliques.

Finally, the junction tree is a subgraph of the clique that is: (a) a tree, (b) contains all the variables of the clique graph and (c) has the junction tree property. This property states that for any two cliques in the junction tree that are connected by a path, the intersection between these two cliques must be contained in each clique on the path. In Figure 2.7 we see a junction tree of the graph in Figure 2.6.

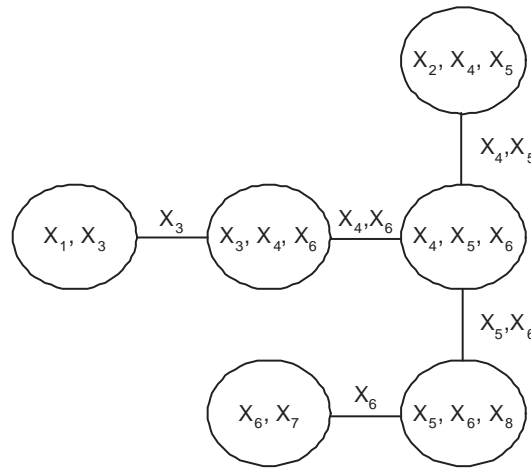


Figure 2.7. A junction tree of the graph in Figure 2.6.

In the junction tree, inference can be done with an algorithm that closely resembles the one introduced above for polytrees. The time complexity of junction tree construction is exponential in the size of the maximal clique. See (Jensen, 1996) for a complete specification of the algorithm.

2.5 Learning Bayesian Networks

A BN consists of a DAG structure and parameters and both can be learned from data. In this section we will show some of the most common techniques.

2.5.1 Learning the Parameters

When learning the parameters of a BN it is generally assumed that the structure of the BN is known. Hence, the task of parameter learning comes down to finding the values:

$$P(X_i | \Pi_i) \quad (2.5)$$

for each variable X_i in the network. The set of all parameters $P(X_i | \Pi_i)$ is often noted as θ . We will describe two of the most used approaches in literature. More information can be

found in, for instance, (Neapolitan, 2003).

Maximum Likelihood

The simplest method is called *maximum likelihood* (ML) estimation. This is a statistical method that estimates the probability of an event $P(x_i|\pi_i)$ by the frequency of this event in the data. This gives us:

$$\hat{P}(X_i = x_k | \Pi_i = x_j) = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}}$$

where $N_{i,j,k}$ is the number of times that the data contains event that $X_i = x_k$ and $\Pi_i = x_j$.

We denote the set of parameters found by using this method as θ^{ML} .

Bayesian estimation

Bayesian estimation consists of finding the most likely parameters $P(X_i|\Pi_i)$ given that the data was observed using priors over the parameters, e.g. *maximum a posteriori* (MAP) estimation. If we assume that each $P(X_i|\Pi_i)$ follows a multinomial distribution, the joint distribution of all $P(X_i|\Pi_i)$ follows a Dirichlet distribution. This allows us to approximate the parameters as:

$$\hat{P}(X_i = x_k | \Pi_i = x_j) = \frac{N_{i,j,k} + \alpha_{i,j,k} - 1}{\sum_k (N_{i,j,k} + \alpha_{i,j,k} - 1)}$$

where $\alpha_{i,j,k}$ are the parameters of the Dirichlet distribution associated with the prior $P(X_i = x_k | \Pi_i = x_j)$.

We denote the set of parameters found by using this method as θ^{MAP} .

2.5.2 Learning the Structure

There are two big groups of structure learning algorithms, namely score-based (Heckerman, 1995) and constraint-based (Spirtes et al., 2000a), (Pearl, 2000). Both have the limitation that without extra assumptions about the underlying distribution, they can only learn the BN up until its Markov equivalence class.

Score-based Learning

Score-based algorithms assign a score to each possible network based on how well the network represents the data and how complex the network is. Then, it is necessary to search through the space of graphs for the network that maximizes this score. In practice, it is impossible to perform an exhaustive search over the space of graphs because the number of possible graphs grows super exponentially with the number of nodes (Robinson, 1977).

Algorithm 1 GES Algorithm (Chickering, 2002b).

Require: A set of samples from a probability distribution faithful to a CBN $P(V)$.**Ensure:** BN.

1. FES: Forward equivalence search
 - a) Start with an empty graph.
 - b) Consider all graphs obtained by single arc additions.
 - c) Group Markov equivalent graphs together and represent them as a CPDAG.
 - d) Choose among these the CPDAG with the highest score.
 - e) Repeat steps (b) and (c) until the score can not be improved.
 2. BES: Backward equivalence search
 - a) Start with the CPDAG resulting from FES.
 - b) Consider all graphs obtained by single arc deletions.
 - c) Group Markov equivalent graphs together and represent them as a CPDAG.
 - d) Choose among these the CPDAG with the highest score.
 - e) Repeat steps (b) and (c) until the score can not be improved.
-

Different heuristics have been proposed for this problem resulting in several algorithms like K2 (Cooper and Herskovits, 1992) and Greedy-Search (GS) (Chickering, 2002b).

Typically, a scoring metric consists of a measure of how well the samples D from a distribution can be modeled by the network G and a penalty factor diminishing the score of complex networks (cf. Occam's razor). Some commonly used scores are the *Bayesian information criterion* (Schwartz, 1978):

$$Score_{BIC}(G, D) = \log L(D|\theta^{ML}, G) - \frac{1}{2} Dim(G) \log N$$

and *minimum description length* (Lam and Bacchus, 1994):

$$Score_{MDL}(G, D) = \log L(D|\theta^{ML}, G) - |A_G| \log N - c \cdot Dim(G)$$

where N is the number of samples, c the number of bits necessary to stock every unique parameter, $|A_G|$ the number of arcs and $Dim(G)$ is the number of free parameters in the graph in the graph G and θ^{ML} the set of parameters $P(X_i|I_i)$ found by ML estimation (cf. Section 2.5.1).

An attribute of these scoring metrics is that they assign the same score to Markov equivalent graphs.

Taking into account the fact that several networks encode the same (in)dependence relationships (Markov equivalence), we can shrink the search space. We can search in the space of Markov equivalent representatives for the one that corresponds to the correct BN, this is done for instance in Greedy Equivalence Search (GES) (Chickering, 2002b). The

size of the space of DAGs and the space of CPDAGs is asymptotically close (ratio of 2.7), but the DAG space has some drawbacks. For instance, iterative algorithms that work by adding/removing or inverting arcs can get stuck in local minima because of the large number of equivalent DAGs having the same score.

We will discuss the GES algorithm in more detail as it is one of the most used and discussed in literature. After a first initialization, GES, as described in Algorithm 1, works in two phases. The different steps of the algorithm are shown on an example in Figures 2.8 and 2.9.

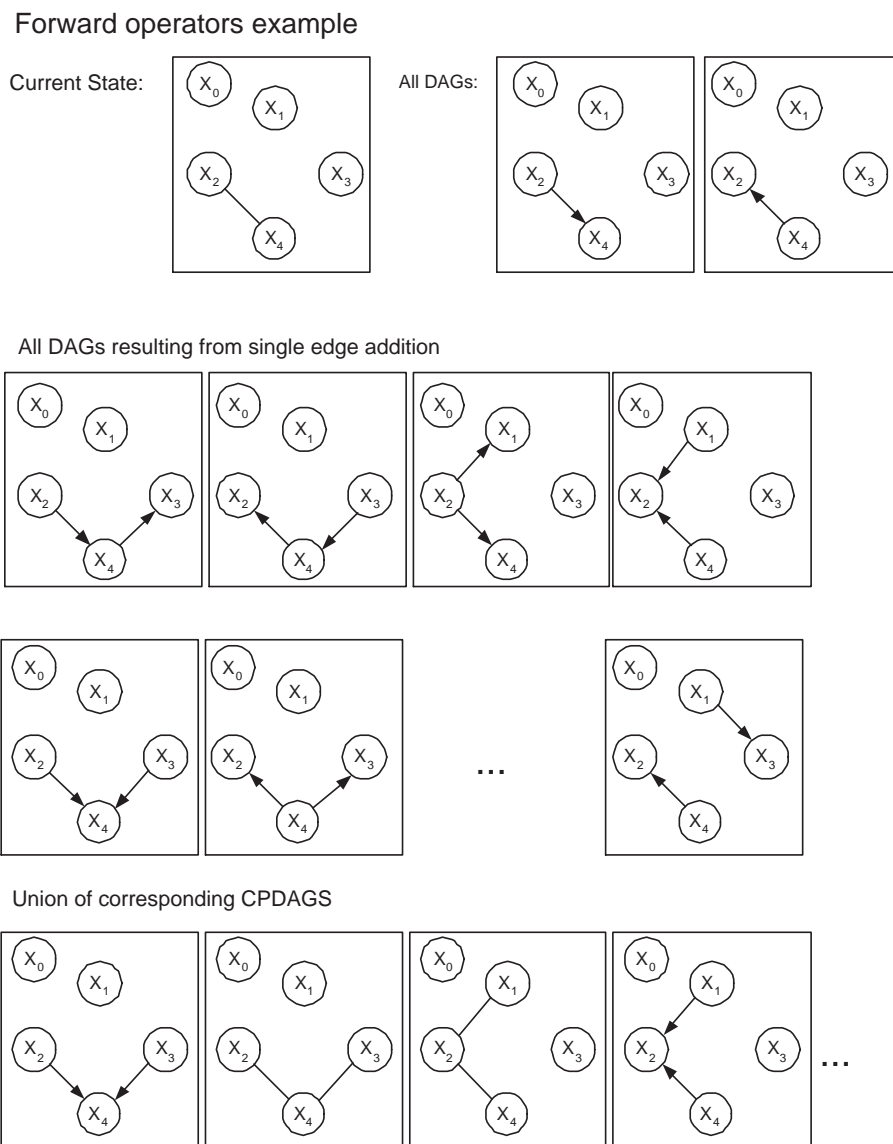


Figure 2.8. Forward equivalence search

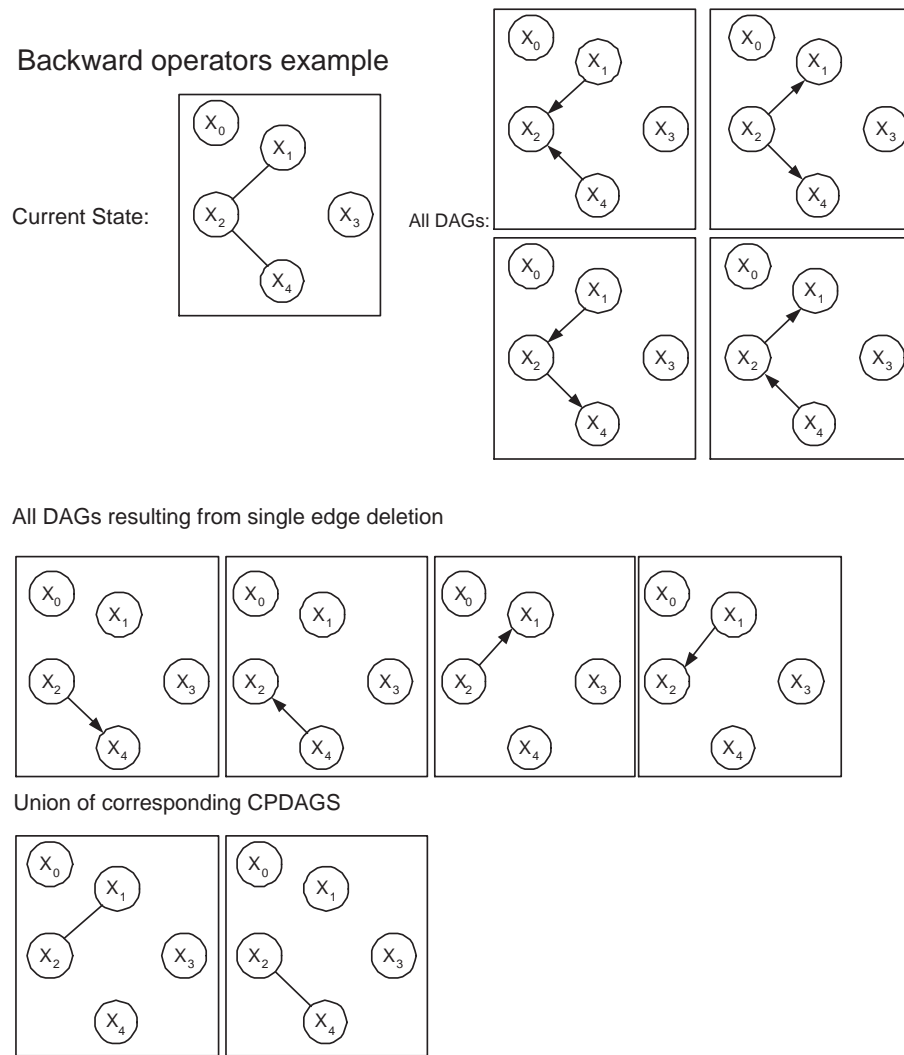


Figure 2.9. Backward equivalence search

GES starts by initializing the current state of the search to be the equivalence class corresponding to the unique DAG with no edges. That is the first state of the search space corresponds to all possible marginal and conditional independence constraints.

In the first phase of the algorithm, we repeatedly replace the current CPDAG ϵ by a member of its neighborhood $Ne^+(\epsilon)$ containing all CPDAGs modeling all DAGs formed by adding an edge to any of the DAGs in the equivalence class modeled by the current CPDAG that has the highest score ϵ^+ until no such replacement increases the score.

The second phase consists of repeatedly replacing the current CPDAG ϵ with the member of its neighborhood $Ne^-(\epsilon)$, which in this case contains all CPDAGs modeling all DAGs formed by removing an edge of any of the DAGs in the equivalence class modeled by the

current CPDAG that has the highest score ϵ^- . Once the algorithm reaches a local maximum in the second phase, it terminates and returns the current CPDAG.

GES has been proved to be asymptotically correct by Chickering (2002b). I.e. if enough data is provided so that the empirical distribution provided by the data converges to the correct theoretical distribution, then GES using the BIC score will converge to the correct CPDAG representing the theoretical distribution.

Chickering (2002b) provided a set of operators allowing to generate the neighborhoods $Ne^+(\cdot)$ and $Ne^-(\cdot)$ of a CPDAG without having to go to the space of DAGs. However, to calculate the score of a CPDAG it is still necessary to choose an instantiation of the CPDAG, i.e. a DAG in the equivalence class modeled by the CPDAG and calculate its score.

Constraint-based Learning

Constraint-based learning algorithms look for (in)dependencies in the data and try to model that information directly into the graphical structure. The structure of these algorithms is as follows:

- Initialization: Start with a complete undirected network
- Skeleton discovery: Remove edges when independences are found to hold between the variables.
- Edge orientation, v -structures: Search for triples for which certain properties hold and orient these triples accordingly.
- Edge orientation, edge propagation: Based on the already oriented edges, apply some orientation rules until no more edges can be oriented.

Examples include PC (Spirtes et al., 2000a), IC (Pearl, 2000) and BN-PC (Cheng et al., 1997).

We will discuss the PC-algorithm in detail, since we adjust PC for learning CBN with latent variables later in this work. The complete PC-algorithm is given in Algorithm 2. PC is proved to be correct asymptotically, which means that if the amount of available data becomes large enough so that all independence tests give the correct result, then the PC algorithm will return the CPDAG modeling the Markov equivalent class of the correct graph modeling the distribution.

Several heuristics exist to make PC faster since the number of independence tests that need to be performed becomes very large.

Example

As an example assume that the correct graph is given in Figure 2.10(a). We start by initializing our complete DAG in Figure 2.10(b).

Algorithm 2 PC algorithm (Spirites et al., 2000a).**Require:** A set of samples from a probability distribution faithful to a CBN $P(V)$.**Ensure:** CPDAG.

1. Initialization:
 - G is complete undirected graph on the vertex set V .
2. Skeleton discovery:
 - $n=0$;
 - repeat
 - repeat
 - select an ordered pair of variables X_i and X_j that are adjacent in G such that $Ne(X_i)\setminus X_j$ has cardinality $\geq n$, and a subset S of $Ne(X_i)\setminus X_j$ of cardinality n , and if X_i and X_j are independent given S delete edge $X_i - X_j$ from G and record S in $Sepset(X_i, X_j)$ and $Sepset(X_j, X_i)$;
 - until all ordered pairs of adjacent variables X_i and X_j such that $Ne(X_i)\setminus X_j$ has cardinality $\geq n$ and all subsets S of $Ne(X_i)\setminus X_j$ of cardinality n have been tested for independence;
 - $n=n+1$;
 - until for each ordered pair of adjacent vertices X_i, X_j , $Ne(X_i)\setminus X_j$ is of cardinality $< n$.
3. Edge orientation: v-structure discovery
 - For each triple of vertices X_i, X_j, X_k such that the pair X_i, X_j and the pair X_j, X_k are each adjacent in G but the pair X_i, X_k are not adjacent in G , orient $X_i - X_j - X_k$ as $X_i \rightarrow X_j \leftarrow X_k$ if and only if X_j is not in $Sepset(X_i, X_k)$.
4. Edge orientation: inferred edges
 - Repeat
 - PC-Rules to come here
 - until no more edges can be oriented.

Now we start the skeleton discovery phase by performing conditional independence tests. When a conditional independence between two variables is found that means the corresponding edge can be removed (Because we assume faithful distributions, hence $\mathbf{W}_1 \perp\!\!\!\perp \mathbf{W}_2 | \mathbf{Z} \Leftrightarrow \mathbf{W}_1 \perp \mathbf{W}_2 | \mathbf{Z}$). Assuming that we test independences concerning X_1 first, we find that $X_1 \perp\!\!\!\perp X_2$ and so the edge $X_1 - X_2$ is removed (Figure 2.10(c)). After performing (conditional) independence tests for each pair of variables, we end up with the structure given in Figure 2.10(d), as you can see this matches the skeleton of the correct graph.

The next step is edge orientation, in which we first try to find the v -structures. We can orient the triple $X_2 - X_4 - X_3$ as $X_2 \rightarrow X_4 \leftarrow X_3$, giving us Figure 2.10(e). Now the final orientation step, the application of the rules to infer the direction of edges, can be applied given us Figure 2.10(f). The edge $X_0 - X_2$ remains undirected, as we have no information on how to direct this edge. However, we can instantiate this edge as long as we do not

create new v -structures or cycles to become the BN that correctly models the underlying (in)dependences (cf. Markov equivalence, see Section 2.3.3).

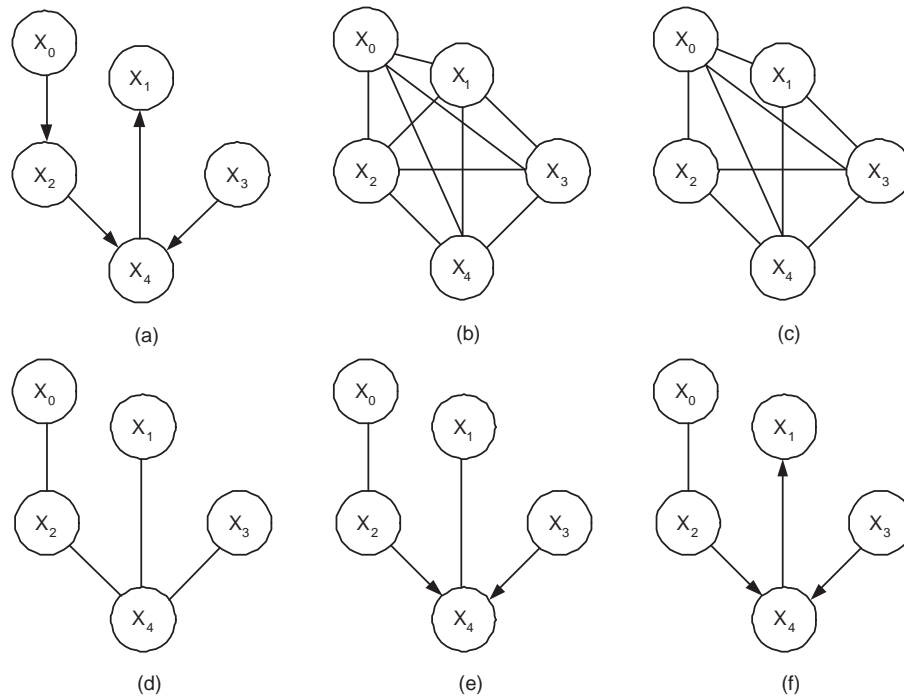


Figure 2.10. Example run of PC.

Markov Blanket Discovery

Recently, a lot of research has been done on learning the structure of BNs by learning the individual *Markov blankets* of each variable and then joining this information to construct the complete graph. A Markov blanket of a variable X_i is the minimal set of variables \mathbf{M}_i that when conditioned upon renders X_i independent of all other variables $\mathbf{V} \setminus (\mathbf{M}_i \cup \{X_i\})$. This set consists of Π_i , $Ch(X_i)$ and $Pa(Ch(X_i)) \setminus \{X_i\}$, meaning the parents, children and other parents of the children of X_i .

Some relevant publications for these algorithms are (J-P. Pellet, 2008) and (Pena et al., 2007).

2.6 Overview of the Chapter

In this chapter we introduced the basic building blocks of our causal models, namely Bayesian networks. These graphical models can be used to concisely represent a JPD by modeling the independences in a graphical structure and keeping a set of CPDs.

We showed how both the CPDs and the graph can be learned from observational data and briefly introduced some of the most common learning techniques. Furthermore, we introduced several ways to perform probabilistic inference, which is the calculation of the effect of observing the state of a certain variable on the probability of other variables in the system.

In the next chapter we will show how BNs can be extended to model causal information.

Causal Bayesian Networks

In this chapter we will introduce *Causal Bayesian Networks* (CBNs). A CBN is an extension of a *probabilistic* BN, as introduced in the previous chapter, that is capable of modeling causal information. The main difference with probabilistic BNs is the semantics of the edges, where they are viewed as carries of dependence relationships in BNs they are representatives of causal relations in CBNs. If we reflect on the requirements of a good causal model (Section 1.2), we have to describe what exactly can be represented by a CBN (Section 3.1), which queries we can resolve with a CBN (Section 3.2), and how it can be learned from observational or experimental data (Section 3.3).

3.1 Definition

A causal Bayesian network is a Bayesian network with an added assumption. A CBN is defined as follows (Pearl, 2000):

Definition 3.1. A *causal Bayesian network* (CBN) is defined as a triple $\langle \mathbf{V}, G, P(X_i | \Pi_i) \rangle$ with:

- a set of observable discrete random variables $\mathbf{V} = \{X_1, \dots, X_n\}$
- a directed acyclic graph (DAG) G , where each node is associated with a variable from \mathbf{V}
- conditional probability distributions (CPD) $P(X_i | \Pi_i)$ of each variable from \mathbf{V} conditional on its immediate parents in the graph G

Furthermore, the directed edges represent an autonomous causal relation between the corresponding variables.

The difference between a CBN and a classical BN is that the edges of a CBN are viewed as representing autonomous causal relations among the corresponding variables, while in a BN the arrows only represent a probabilistic dependency, and not necessarily a causal one. A directed edge $C \rightarrow E$ in a CBN indicates that there exists at least one intervention on C

that would alter the distribution of the values of E given that all other variables are kept at certain constant values.

This means that in a CBN, each CPD $P(X_i|II_i)$ represents a stochastic assignment process by which the values of X_i are chosen in response to the values of II_i in the underlying domain. This is an approximation of how events are physically related with their effects in the domain that is being modeled. For such an assignment process to be autonomous means that it must stay invariant under variations in the processes governing other variables (Pearl, 2000). For example, a change in $P(X_i|II_i)$ should not alter the distribution of $P(X_j|II_j), \forall j \neq i$.

So we assume that the causal Bayesian network is the one that *generates* the JPD. In Figure 3.1 we see a conceptual sketch: the box represents the real world where a causal Bayesian network generates the data in the form of a joint probability distribution. Below we see the BNs that represent all the independence relations present in the JPD (Markov equivalent structures). Only one of them is the causal Bayesian network, in this case the rightmost.

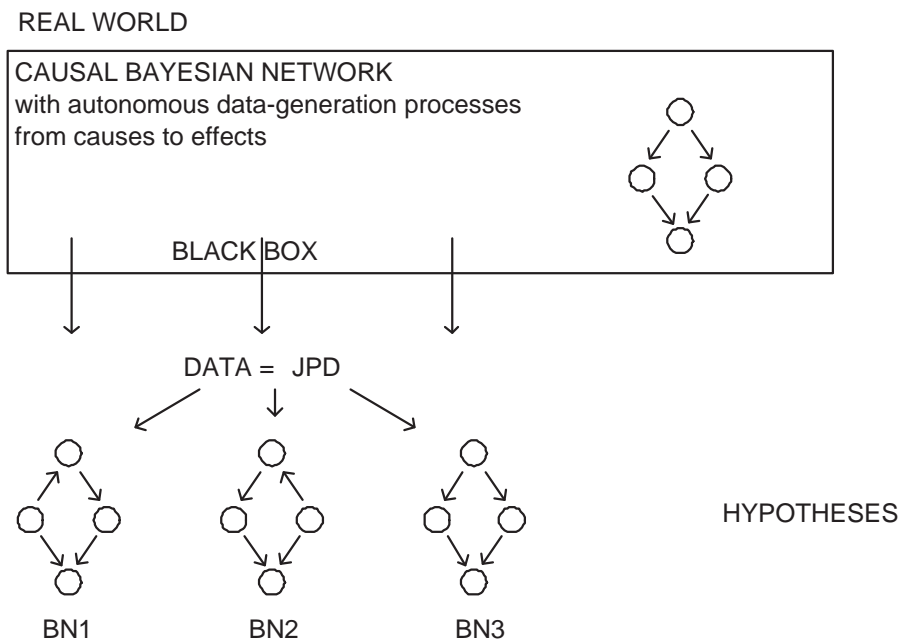


Figure 3.1. Conceptual sketch of how a CBN generates a JPD, that in its turn can be represented by several probabilistic BNs of which one is a CBN.

In the next section we show the main use of CBNs, namely causal inference.

3.2 Causal Inference

We start by explaining the difference between observing and intervening, then treat an important theorem that specifies how to incorporate the effect of an intervention in a CBN. After that, the problem of calculating causal effects or identification is explained.

3.2.1 Observation vs. Intervention

An important issue in graphical models is to distinguish between different types of belief revision, each of which modify a given probability distribution in response to information obtained. There exist many types of belief revision such as different types of imaging (Lewis, 1981), (Crestani and Rijsbergen, 1994) and traditional probabilistic conditioning. However, an elaborate discussion on them falls outside the scope of this thesis.

We consider only two types of conditioning: probabilistic conditioning (conditioning by observation) and conditioning by intervention, which will be discussed below.

Definition 3.2. Conditioning by observation refers to the way in which a probability distribution of X_j should be modified when a modeler passively observes the information $X_i = x_i$.

They are presented by conditional probabilities that are defined as follows:

$$P(X_j = x_j | X_i = x_i) = P(x_j | x_i) = \frac{P(X_j = x_j, X_i = x_i)}{P(X_i = x_i)}. \quad (3.1)$$

It is important to realize that this is typically not the way the distribution of X_j should be modified if we intervene externally and force the value of X_i to be equal to x_i .

Definition 3.3. Conditioning by intervention¹ refers to the way the distribution X_j should be modified if we intervene externally and force the value of X_i to be equal to x_i .

To make the distinction clear, Pearl has introduced the **do-operator** (Pearl, 2000)²:

$$P(X_j = x_j | do(X_i = x_i)) \quad (3.2)$$

The interventions we are treating here are surgical in the sense that they only directly change the variable of interest (X_i in the case of $do(X_i = x_i)$). We sometimes write $do(x_i)$ as a shorthand for $do(X_i = x_i)$.

Generally, the two quantities will be different:

¹ Throughout this text the terms *intervention* and *manipulation* are used interchangeably.

² In the literature other notations such as $P(X_j = x_j | X_i = x_i)$, $P_{X_i=x_i}(X_j = x_j)$, or $P(X_j = x_j | X_i = \hat{x}_i)$ are abundant.

$$P(X_j = x_j | do(X_i = x_i)) \neq P(X_j = x_j | X_i = x_i) \quad (3.3)$$

and the quantity on the left-hand side cannot be calculated from the joint probability distribution $P(\mathbf{V})$ alone, without additional assumptions imposed on the graph, i.e. that the directed edges represent an autonomous causal relation as in CBNs.

Consider the simple CBNs of Figure 3.2: in the left graph

$$P(x_j | do(x_i)) = P(x_j | x_i)$$

as X_i is the only immediate cause of X_j , but

$$P(x_i | do(x_j)) = P(x_i) \neq P(x_i | x_j)$$

as there is no direct or indirect causal relation going from X_j to X_i . The equalities above are reversed in the graph to the right, i.e. there it holds that $P(x_j | do(x_i)) = P(x_j) \neq P(x_j | x_i)$ and $P(x_i | do(x_j)) = P(x_i | x_j)$.



Figure 3.2. Two simple causal Bayesian networks.

Before showing a more challenging example of the difference between conditioning by observation and intervention, we introduce a theorem that specifies how an intervention modifies the JPD associated with a CBN.

3.2.2 Manipulation Theorem

Performing an external intervention in a domain that is modeled by a CBN, modifies that domain and the JPD that is used to model it.

We only study the simplest form of interventions in this work, i.e. fixing a set of variables \mathbf{M} to some constants $do(\mathbf{M} = \mathbf{m})$. In principle more general types of intervention are possible, such as fixing the value of variables \mathbf{M} in a way that depends on previously observed variables. Furthermore, we only take into account interventions on observable variables.

For reasons of clarity, we repeat the factorization of the JPD:

$$P(\mathbf{V}) = \prod_{X_i \in \mathbf{V}} P(X_i | \Pi_i)$$

To obtain the post-intervention distribution after fixing a set of variables $\mathbf{M} \subset \mathbf{V}$ to fixed values $\mathbf{M} = \mathbf{m}$, the factors with the variables in \mathbf{M} conditional on their parents in the graph (i.e. their causes in the pre-intervention distribution), have to be removed from the JPD. Formally these are : $P(M_i|Pa(M_i))$ for all variables $M_i \in \mathbf{M}$. This is because after the external intervention, it is this intervention rather than the parent variables in the graph that cause the values of the variables in \mathbf{M} . Furthermore the remaining occurrences of \mathbf{M} in the JPD have to be instantiated to $\mathbf{M} = \mathbf{m}$.

An intervention of this type only has a local influence in the sense that only the incoming links of a manipulated variable have to be removed from the model, no factors representing other links have to be modified, except for instantiating the occurrences of the manipulated variables \mathbf{M} to \mathbf{m} . This is a consequence of the assumption of CBNs that the factors of the JPD represent assignment processes that must stay invariant under variations in the processes governing other variables. Formally, we get (Spirtes et al., 2000a):

Theorem 3.1. *Given a CBN with variables $\mathbf{V} = X_1, \dots, X_n$ and we perform the intervention $do(\mathbf{M} = \mathbf{m})$ for a subset of variables $\mathbf{M} \subset \mathbf{V}$, the post-intervention distribution becomes:*

$$P(\mathbf{V}|do(\mathbf{m})) = \prod_{X_i \in \mathbf{V} \setminus \mathbf{M}} P(X_i|I_i) \Big|_{\mathbf{M}=\mathbf{m}} \quad (3.4)$$

Where $\Big|_{\mathbf{M}=\mathbf{m}}$ stands for instantiating all the occurrences of the variables \mathbf{M} to values \mathbf{m} in the equation that precedes it.

Assume we have the CBN representing the Tsunami warning system example given in the introduction and repeated here in Figure 3.3. Imagine we want to disable the tsunami detection buoys D (e.g. to save electricity) by performing the appropriate intervention $do(D=false)$. Such an intervention changes the way in which the value of D is being produced in the real world. Originally, the value of D was being decided by its immediate cause: *Undersea Earthquake* U . After performing the intervention, namely disabling the buoys, the presence or absence of an undersea earthquake no longer influences the value of D . Instead the new cause of the value of D is the intervention itself.

In Figure 3.4 the graph of the post-intervention CBN is shown. There we can see that the link between the original cause of D has been severed and that the value of D has been instantiated to *false*.

The full factorization of the JPD representing our tsunami warning system before the intervention, using notation wise: seismic measurement S , Tide-sea-level measurement L , tsunami warning W , is:

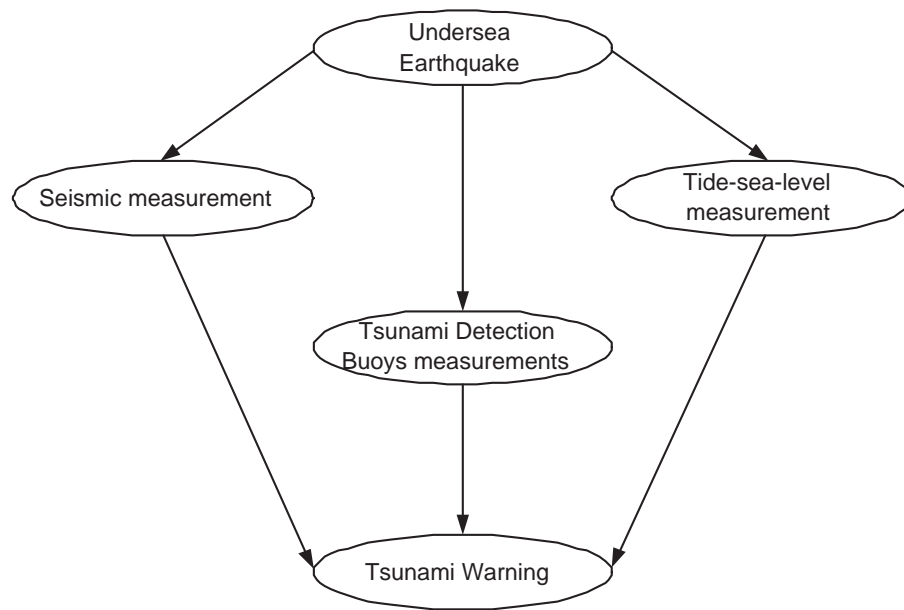


Figure 3.3. CBN representation of tsunami warning system.

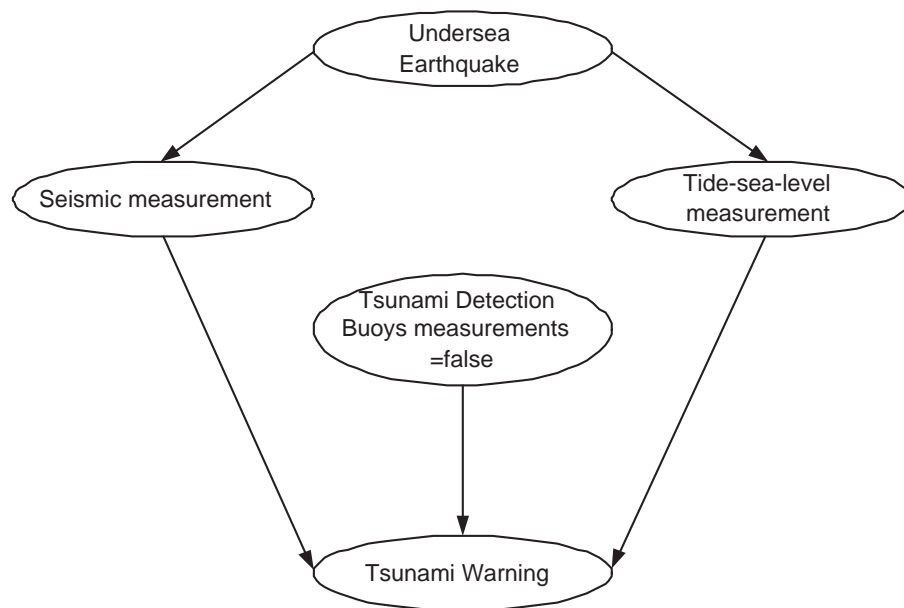


Figure 3.4. The CBN of the tsunami warning system of after disabling the detection buoys via an external intervention: $do(D=false)$.

$$P(\mathbf{v}) = P(u).$$

$$P(s|u).$$

$$P(l|u).$$

$$P(d|u).$$

$$P(w|s, d, l)$$

When we apply the manipulation theorem of Equation (3.4) to incorporate the effect of disabling the detection buoys we obtain the post-intervention distribution:

$$\begin{aligned} P(\mathbf{v}|do(D = false)) &= P(u). \\ &P(s|u). \\ &P(l|u). \\ &P(w|s, D = false, l) \end{aligned}$$

As a comparison, the post-observation distribution after merely observing that the detection buoys do not detect a tsunami is as follows:

$$\begin{aligned} P(\mathbf{v}|D = false) &= P(u). \\ &P(s|u). \\ &P(l|u). \\ &P(D = false|u). \\ &P(w|s, D = false, l) \end{aligned}$$

The only difference with the post-intervention distribution is the presence of the factor $P(D = false|u)$.

Remark that the post-observation distribution can be obtained using the definition of a conditional probability (using the first letters of the variable names):

$$P(u, s, l, w|d) = \frac{P(u, s, l, w, d)}{\sum_{u, s, l, w} P(u, s, l, w, d)}$$

This equation implies that to obtain the post-observation distribution, the JPD contains sufficient information and the structure of the BN is not needed. In contrast, to obtain the post-intervention distribution we explicitly need information contained in the structure, i.e. the parents of the variables of the manipulated variables.

3.3 Structure Learning

In this section we discuss learning CBNs from data and/or experiments. We focus on learning the structure since we are now interested in finding causal relationships among variables and not just in finding probabilistic dependencies. We start by showing the intrinsic difficulties of this task and how in the light of our assumptions we can perform it. Then we give an overview of the properties of different learning algorithms and give an overview of the state of the art. In the next chapter we will introduce our contributions on learning the structure of CBNs. Parameter learning can be done as for probabilistic BNs, see Section 2.5.1.

3.3.1 Learning from Observational Data

The biggest problem when learning causal relationships from observational data is that we simply do not observe causal relationships. What we actually observe is the cause, the effect and the fact that they occur in a fixed pattern. This correlation of variables implies some form of causal relationship. The goal of learning causal structure from observational data is to find exactly which causal relationship is depicted.

In the causal interpretation, a correlation implies an unresolved causal structure, which means that if two variables X_i and X_j are correlated they have one of the following underlying causal structures (Shipley, 2000):

- X_i is a cause of X_j
- X_j is a cause of X_i
- There is a common cause X_c of X_i and X_j

These three possibilities are depicted graphically in Figure 3.5.

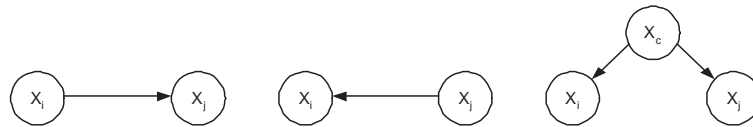


Figure 3.5. When there is a correlation between X_i and X_j , one of these three causal structures hold.

The third possibility occurs if not all variables that influence the system are known. We call X_c a *latent* variable, we will discuss models with latent variables in the Chapter 5.

If we only have data on two variables it is impossible to distinguish between $X_i \rightarrow X_j$ and $X_i \leftarrow X_j$. However when there are three variables that have a specific set of (in)dependence relationships it is possible to identify (under the assumption of causal sufficiency (Section 2.3.2)) which variables are causes of another one. This dependence relationships can be graphically represented by what is known as a *v*-structure shown in Figure 3.6. The (in)dependence relationships that hold in this case are:

- X_i is independent of X_j
- X_i is dependent of X_m
- X_j is dependent of X_m
- X_i is dependent of X_j given X_m

Intuitively you can imagine that X_i and X_j are two diseases with a similar symptom X_m (feaver). We assume that the occurrence of both diseases is independent of each other. When we know that someone has feaver, then adding the knowledge that she/he does not have disease X_i increases our belief in disease X_j .

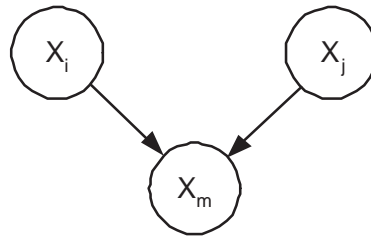


Figure 3.6. A v-structure.

It is clear that it is not always possible to find the causal relationship between any two variables based on observational data. The PC algorithm introduced in Section 2.5.2 is an algorithm, that searches explicitly for *v*-structures and its result is a CPDAG in which all directed arrows are causal (assuming all independence tests are correct).

Learning using Algebraic Constraints

Recent work by (Shpitser and Pearl, 2008) focuses not only on finding dependence relationships in the observational data but also algebraic constraints. They study one type of algebraic constraints called *dormant independences*, which are conditional independences that hold in interventional distributions. They give a complete algorithm for determining if a dormant independence between two sets of variables resides in an interventional distribution that can be predicted without resorting to the actual interventions. They further show the usefulness of dormant independences in model testing and induction. The details can be found in Shpitser and Pearl (2008). In this dissertation we do not take into account dormant independence.

3.3.2 Learning from Experimental Data

Many researchers use either randomised or controlled experiments to discover causal relationships (Shiple, 2000). Assuming that there are no latent variables and that we know that there is a correlation between two variables X_i and X_j we can perform an experiment on either variable to discover the underlying causal relationship. For instance when we manipulate X_i we can see whether this exerts a change in the distribution of X_j . If this is the case then we know that X_i is the cause of X_j and otherwise we know that X_j is a cause of X_i .

A *structural experiment* on a variable X_i , removes the influence of other variables in the system on X_i and cuts the connection between X_i and its parents in the graph. The experiment forces a distribution on X_i , and thereby changes the joint distribution of all variables in the system that depend directly or indirectly on X_i but does not change the conditional

distribution of other variables given values of X_i . After the intervention, the associations of the remaining variables with X_i provide information about which variables X_i influences.

However not all experiments are structural, it is possible that an experiment only changes the conditional distribution of the variable given its causes. These type of interventions are called *parametric interventions*. It is even possible that the targets of an intervention are unknown and might even be more than a single variable (fat hand). Such interventions are known as *uncertain interventions*.

There are two main approaches to perform experiments, the controlled and the randomized experiment.

Controlled Experiment

The controlled experiment consists of proposing a hypothetical structure of cause-effect relationships, deducing what would happen if particular variables are controlled or "fixed" in a particular state, and then comparing the observed result with the predicted outcome.

Randomized Experiment

The randomized experiment consists of creating identically experimental units on which we use a certain treatment (intervention). Remember that if we find a correlation between two variables X_i and X_j , there are three possible causal explanations: $X_i \rightarrow X_j$, $X_i \leftarrow X_j$ and $X_i \leftarrow X_c \rightarrow X_j$ with X_c a common cause. Randomization serves two purposes in retrieving causal relationships. First, it ensures that there is no causal effect coming from the experimental units to the treatment variable or from a common cause to both. Second, it helps to reduce the likelihood in the sample of a chance correlation between the treatment variable and some other cause of treatment. By randomising the treatment variable we generate a sampling distribution that allows us to calculate the probability of observing a given result by chance if, in reality, there is no effect of the treatment.

3.4 Overview of State-of-the-art

In this section we give an overview of the state-of-the art algorithms for learning the complete structure of a CBN. We start by identifying the most common properties and assumptions of the algorithms and then discuss them in more detail. We divide the algorithms into two sections depending on the type of experiments that are used.

3.4.1 Properties of Learning Algorithms

We give a list of a set of properties and assumptions of most structure learning algorithms.

Constraint vs. Score-based:

As we have seen in Section 2.5.2, there are two main groups of structure learning algorithms: constraint-based and score-based. Both techniques have been used to learn CBNs as well.

Experimental vs. Observational and Experimental Data:

When learning CBNs it is often necessary to perform experiments since we can only learn a graph from observational data up to its Markov equivalence class, see Section 3.3. Some algorithms are based solely on experiments while others use the observational data as a starting point.

Perfect vs. Imperfect Data:

Sometimes it is assumed that an infinite amount of observational or experimental data is present (we call this *perfect* data), which means that the independence relations in the data exactly model the d -separation relations as shown by Pearl (2000). However for real world problems this is often not the case, which is a cause for possible errors during learning.

(no) causal sufficiency:

Most studies assume causal sufficiency (cf. Section 2.3.2), that is there are no latent variables, cause this complicates the learning process. In Chapters 5 and 6 we discuss algorithms that take into account latent variables.

Structural vs. Alternative Interventions:

Since experiments are needed to learn the completely oriented CBN, we have to perform interventions on the system. Several types of interventions can be used, for instance structural and parametric, see Section 3.3.2.

3.4.2 CBN Structure Learning with Structural Interventions

We will discuss several studies that have been performed to learn a CBN from observational and experimental data and will indicate which of the previously mentioned properties they have.

Theoretical Study

Eberhardt et al. (2005a) have done a theoretical study on the lower bound of the number of structural experiments needed in the worst case scenario. They study a constraint-based method in which observational data is used to learn the dependencies in the data. Causal sufficiency and perfect data are assumed.

It is shown that when combining structural interventions with search procedures for graphical causal models $N - 1$ experiments suffice to determine the causal relations among $N > 2$ variables when each experiment randomizes at most one variable. They also show that no adaptive procedure can do better than the $N - 1$ lower bound on the number of experiments required to identify the structure in the worst case.

Eberhardt et al. (2005b) extended this result by allowing multiple structural experiments to be performed simultaneously, in which case the lower bound becomes $\log_2(N) + 1$. For all K , $0 < K < \frac{1}{2}N$ they provide an upper bound on the number of experiments needed when each experiment simultaneously randomizes K variables. For $K < \frac{N}{2}$ they show that $(\frac{N}{K} - 1) \lceil \frac{N}{2K} \log(K) \rceil$ experiments are sufficient and in the worst case necessary.

Active Learning

Tong and Koller (2001) and Murphy (2001) proposed score-based active learning techniques to learn the structure of causal networks without latent variables from perfect experimental data gathered by structural experiments.

Both techniques proposed to perform experiments based on (a) the current belief about the structure, (b) the causal information that will be gained by an experiment. The belief is modeled by $P(G|D^i)$, a probability distribution over the set of DAGs given the data seen so far. They update this belief after each experiment and then reiterate the process. Since the space of DAGs is super exponential in the number of nodes, an approximation is needed for $P(G|D^i)$.

Tong and Koller (2001) assumed that there are a number of query variables that can be experimented on and then the influence on all other variables is measured. In order to choose the optimal experiment they introduce a utility function, the loss-function, based on the uncertainty of the direction of an edge, to help indicate which experiment gives the most information. Using the results of their experiments they update the distribution over the possible networks and network parameters. Since it is impossible to do this for the entire set of DAGs they use an approximation based on the ordering of the variables proposed by Friedman and Koller (2000).

Murphy (2001) proposed a technique which resembles the one of Tong and Koller (2001) but different approximations are used to overcome working in the space of DAGs. Murphy

(2001) used MCMC to approximate the belief state $P(G|D^i)$ and importance sampling to calculate the expected utility.

Learning from Mixture of Observational and Experimental Data

Cooper and Yoo (1999) described another score-based method which can learn the structure from an arbitrary mixture of imperfect observational and experimental data. It is assumed that the performed experiments are structural. A closed-form Bayesian scoring metric was derived that can be used in this context: the metric takes into account whether the data is from observations or from experiments and adapts the score likewise. The new scoring metric is an adaptation of the one proposed by Cooper and Herskovits (1992) and Heckerman et al. (1995) for observational data alone.

3.4.3 Alternative Interventions

In most studies it is assumed that an intervention involves randomly assigning values to a single variable and measuring some other possible response variables. Recently researchers have studied different types of interventions. We will discuss two of these approaches that were introduced by Eberhardt and Scheines (2006) and Eaton and Murphy (2007) respectively.

Parametric Interventions

Eberhardt and Scheines (2006) argued that Fisher's theory on randomized trial is far from complete when there is no a priori designation of causes and effects. First of all, there is no guaranteed way to find the optimal sequence of experiments. Secondly there are statistical problems when combining results of experiments on different sets of variables and finally there are many more types of interventions besides randomized assignments.

Performing a structural intervention, i.e. randomizing a variable X_i , makes the variable independent of its original causes. This is represented by adding an exogenous variable I which has two states (on/off) indicating that an intervention took place or not respectively. When I has value *on*, I makes X_i independent of its causes (parents) and determines the distribution of X_i . So the CPD $P(X_i|II_i)$ is replaced by $P(X_i|I = \textit{on})$.

Another type of interventions are parametric (soft) interventions which do not remove the influence of the original causes but modify the CPD. This can again be modeled by an exogenous variable I_{soft} with two states (on/off). When I_{soft} has value *on*, the CPD $P(X_i|II_i)$ is replaced by $P(X_i|II_i, I_{soft} = \textit{on})$.

The main result about parametric interventions by Eberhardt and Scheines (2006) is the following theorem:

Theorem 3.2. *One experiment is sufficient to discover the causal relations among a causally sufficient set of N variables if multiple variables can be simultaneously and independently subjected to a parametric intervention per experiment.*

As is discussed by Eberhardt and Scheines (2006) these results do not imply that parametric interventions are always better than structural interventions. First, more samples are needed because parametric interventions require more conditional independence tests with larger condition sets. Secondly, the theorem only holds for causally sufficient sets, however previous results by Eberhardt et al. (2005a) and Eberhardt et al. (2005b) on structural interventions also depend on causal sufficiency.

Uncertain Interventions

Eaton and Murphy (2007) further relax the requirements for interventions and study the setting in which the targets of the intervention are a priori unknown, these type of interventions are called uncertain. One exogenous intervention node I_{unc} may have multiple regular children (fat hand). It is shown that the dynamic programming algorithm proposed by Koivisto and Sood (20004) and Koivisto (2006) can be extended to the case where data is obtained from these types of experiments.

When there are no interventions, the marginal likelihood for a node and its parents is given by:

$$P(x_i^{1:N} | \pi_i^{1:N}) = \int \left[\prod_{n=1}^N P(x_i^n | \pi_i^n, \theta_i) \right] P(\theta_i) d\theta_i \quad (3.5)$$

where N is the number of data points.

The marginal likelihood in the case of uncertain interventions becomes:

$$P(x_i^{1:N} | \pi_i^{1:N}) = \prod_l \int \left[\prod_{n: I_{unc,i}=l} P(x_i^n | \pi_i^n, \theta_i^l) \right] p(\theta_i^l) d\theta_i^l \quad (3.6)$$

where $I_{unc,i}$ are the intervention parents of X_i and θ_i^l depicts the parameters for node X_i given that its parents have state l .

A Bayesian approach to learning the structure is used, however it is well known that computing the full posterior is intractable. Therefore sampling methods are used, the details thereof are outside the scope of this article, more information can be found in the papers of (Eaton and Murphy, 2007) and (Koivisto, 2006).

The main result in this article is that the extension of the algorithm of Koivisto and Sood (20004) has a time complexity of $O(d2^{d_X} + d^{k+1}C(N))$ where d_X is the number of regular nodes, d the number of total nodes (regular + intervention) and $C(N)$ is the cost of computing each local marginal likelihood term.

3.5 Overview of the Chapter

In this chapter we showed how probabilistic Bayesian networks can be extended to represent causal relationships between variables in a system.

Furthermore we showed that this new semantic interpretation of the arcs in a CBN allows to perform causal inference, which is the calculation of the effect of performing an intervention on the system.

The causal interpretation of the edges has as a consequence that learning the structure of a CBN no longer amounts to finding *A* member of the equivalence class but finding *THE* data generating network. We gave an overview of state-of-the art algorithms for handling this task.

In the next chapter we will introduce our algorithms for learning the complete structure of a CBN under certain assumptions.

Contributions on Causal Modeling

In this chapter we will introduce our contributions for Causal Bayesian Networks (CBNs). We propose two algorithms to learn the structure of CBNs using (a) available observational data and (b) data gathered from specific experiments. The key to both these contributions is our observation that available data is seldom sufficient to learn the complete causal structure. As mentioned in the previous chapters, a distribution can be modeled by different networks with different structures (Markov equivalence). Furthermore, sometimes just a limited amount of data is available which renders automatic learning techniques less reliable.

In the next two sections we introduce two algorithms MyCaDo (MY CAusal DiscOvery) and UnCaDo (UNsure CAusal DiscOvery). MyCaDo still assumes that the available observational data is sufficient (*perfect* data) to recover the representative of the correct Markov equivalence class of probabilistic networks modeling the data. The innovation of MyCaDo is that it tries to recover the complete causal structure using a minimal amount of experiments or with a minimal cost. The UnCaDo algorithm no longer expects that the observational data comprises all dependence information exactly. In this case we allow to explicitly model edges, representing probabilistic/causal dependencies, of which we are unsure. The goal is then to remove these *unsure* edges in a first phase and then complete the structure as for

4.1 Our Greedy Approach for Perfect Observational and Experimental Data

In this section we introduce our approach on learning a CBN from perfect observational and experimental data. We start with a general description of the algorithm and then discuss each part in more detail.

4.1.1 General Description

The general overview of the algorithm is given in Figure 4.1.

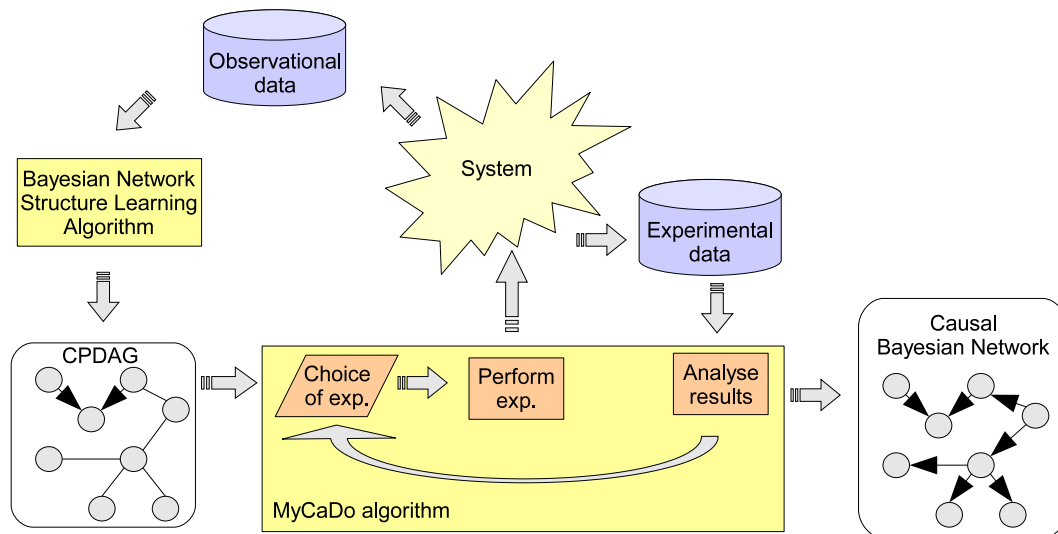


Figure 4.1. Main description of the MyCaDo approach.

We assume that we are given a dataset of perfect observational data from a system under study that can be modeled by a CBN. In the first phase we learn the CPDAG from this dataset using traditional structure learning techniques such as the ones proposed in Section 2.5.2.

The second phase is the main application of our approach, the MyCaDo (My Causal DiscOvery) algorithm, proposed by Meganck et al. (2006a), which consists of three parts. In the first part (Section 4.1.2) we have to decide which experiment we will perform and hence also which variables we will alter and measure. Secondly, we have to do the actual experiment (Section 4.1.4), and finally we have to analyze the results of our experiment (Section 4.1.5). The results of our experiments will allow us to direct a number of edges in the partially directed structure. If there are still some edges that are undirected we re-iterate over the second phase. When all edges are directed we have the correct CBN modeling the original data.

We have to note however that if some experiments are impossible to perform (too expensive / unethical) it is possible that the result of MyCaDo is not a completely directed graph cause the direction of some edges can not be resolved.

4.1.2 Choice of Experiment

We use elements from decision theory to decide which experiment we will perform at each step. In general a decision problem consists of three parts: values (symptoms, observables), actions and possible consequences. It is assumed that these are given in advance. It is possible to order the consequences by preference by using a utility function. Hence we can choose

the action that will lead to our preferred result based on some decision criteria such as least risk or optimistic estimation. We can also maximize the expected utility of each action by defining a probability distribution on the expected results of this action.

Utility Function

In general, our utility function $U(\cdot)$ will be a function of three variables: $gain(exp)$, $cost(exp)$, $cost(meas)$, respectively the gained information, the cost of performing an experiment and the cost of measuring other variables. The only restriction that is placed on the utility function is that it is proportional to $gain(exp)$ and inversely proportional to the sum of $cost(exp)$ and $cost(meas)$.

If we denote performing an action (=experiment) at X_i by A_{X_i} , and measuring the neighboring variables by M_{X_i} then the utility function we use is:

$$U(A_{X_i}) = \frac{\alpha gain(A_{X_i})}{\beta cost(A_{X_i}) + \gamma cost(M_{X_i})}$$

where α , β and γ are measures of importance for every term. We will assume $\alpha = \beta = \gamma$ unless stated otherwise, to simplify the notation.

Gain of an Experiment

Let's assume that we perform an experiment on X_i and that we can measure all neighboring variables $Ne(X_i)$. In this case we can direct all edges connecting X_i and $Ne(X_i)$ as a result of the experiment. So in this case the gain, $gain(A_{X_i})$, with A_{X_i} an experiment on X_i , is based entirely on the number of variables $Ne_U(X_i)$ that are connected to X_i by an undirected edge, which will be directed according to the result of the experiment.

However, it is possible that directing one edge can infer direction of other edges, see the final phase of the PC-algorithm in Algorithm 2. We can take into account the possibility of inferred edges in $gain(A_{X_i})$.

Note that the amount of edges of which the direction can be inferred after performing an experiment is entirely based on the instantiation of the undirected edges connected to the one being experimented on. Instantiating an undirected edge is assigning a direction to it, so for instance if we have an edge $X_i - X_j$, then $X_i \rightarrow X_j$ and $X_i \leftarrow X_j$ are the two possible instantiations of that edge. We denote $inst(A_{X_i})$ as the set of instantiation of the undirected edges connected to X_i . The number of inferred edges based on $inst(A_{X_i})$ is noted as $\#inferred(inst(A_{X_i}))$.

Note that two undirected substructures, which are subgraphs consisting of nodes connected by undirected edges, that are not linked in any way by undirected edges can not

be influenced by performing an experiment in the other substructure when the CPDAG is correct. Since we assume in this context that all discovered edges are correct no existing edges can change based on inference by experiments, and hence no new information can be inferred through a set of already directed edges. So the calculation of the utility of an experiment is only based on that part of the graph that is connected to the variable by undirected links.

The problem can hence be separated in sub-problems, each concerning a part of the graph connected by undirected edges. In the following we will introduce solutions for a single substructure that is entirely constituted of undirected links. This result can then be mimicked for the other undirected substructures.

Cost of Experiment and Measurement

The cost of an experiment can be the time needed, the amount of space it takes or simply the amount of money it costs to perform an experiment. It is dependent on the situation in which every experiment takes place and will typically be given by experts.

It is important to note that there are certain experiments that can not be performed, either because of ethical reasons (for example infecting people with HIV) or simply because it is impossible to do so (for example changing the season). These types of experiments will be assigned a cost value of infinity (∞) and thus the gain of performing such an experiment will be 0, therefore it will not add any new information.

In order to gain anything from an experiment, we have to perform measurements on the variables of interest. It is however important to note that measuring itself can be costly and can diminish the usefulness of an experiment although it does not directly concern the variable that is being altered. For instance, injecting someone with a certain medical fluid might not cost that much, but when the only way to check for changes is performing a CT-scan, measuring the results might add a huge cost factor.

4.1.3 Decision Criteria

In this section we will discuss a number of decision criteria for our learning problem. Our approach allows the possibility to maximize any of these criteria for the choice of the optimal experiment at each stage. Depending on the type of situation in which to perform the experiments it might be advantageous to choose a specific criterion.

Maximax

The *maximax* decision criterion is an optimistic one, which means that we choose the action that could give the best result, that is the one that might direct the most arrows. In our case

this means that we perform an experiment on X_{best} with:

$$X_{best} = \underset{X_i}{argmax} \left(\frac{\#Ne_U(X_i) + \max_{inst(A_{X_i})} (\#inferred(inst(A_{X_i})))}{cost(A_{X_i}) + cost(M_{X_i})} \right)$$

This is the sum of the number of undirected edges connected to X_i and the maximum number of inferred edges by any of the instantiations of the directions of the undirected edges connected to X_i , divided by the cost.

Maximin

The *maximin* decision criterion is a pessimistic one, which means that we assume that for each experiment at a variable X_i the least number of possible inferred edges can be found. This means that the minimum amount of edges is oriented by any instantiation of all edges connected to X_i . With this criterion, we perform an experiment on X_{best} with:

$$X_{best} = \underset{X_i}{argmax} \left(\frac{\#Ne_U(X_i) + \min_{inst(A_{X_i})} (\#inferred(inst(A_{X_i})))}{cost(A_{X_i}) + cost(M_{X_i})} \right)$$

The instantiation of edges that would induce the least inferred edges in general would be the one where all arrows are pointing at X_i , but this might create new v -structures and thus is not always possible. So if two neighbors of X_i are not directly connected, one of the links has to be out of X_i and hence possibly leads to inferred edges. If X_i is part of a clique then or is the sole neighbor of another node then having a link out of X_i might also not trigger inferring new edges.

Expected utility

The expected utility is based on a distribution of the directions of the edges, which can be used to calculate the probability, based on the members in the equivalence of the graph under study, of any instantiation of directions that might occur.

Instead of just assuming a uniform distribution of the edges we can look at all possible DAGs in the equivalence class of the discovered CPDAG and count for each pair $X_i - X_j$, the number of times $X_i \rightarrow X_j$ and $X_i \leftarrow X_j$ appears and hence we can assume that:

$$P_{eq}(X_i \rightarrow X_j) = \frac{\#(X_i \rightarrow X_j)}{\#\text{members of eq. class}}$$

$$P_{eq}(X_i \leftarrow X_j) = \frac{\#(X_i \leftarrow X_j)}{\#\text{members of eq. class}}$$

Note that in next steps of this learning phase we no longer have a CPDAG, because some new edges are directed at each iteration. We should then take into account all members of the original equivalence class that share the exact same directed edges, for convenience we will still refer to this set of DAGs as the *members of the equivalence class of the current PDAG*.

Using this approach it would mean that we perform the experiment on the variable X_{best} with:

$$X_{best} = \underset{X_i}{\operatorname{argmax}} \left(\frac{\#Ne_U(X_i) + \sum_{inst(A_{X_i})} \#inferred(inst(A_{X_i}))P_{eq}(inst(A_{X_i}))}{cost(A_{X_i}) + cost(M_{X_i})} \right) \quad (4.1)$$

where $P_{eq}(inst(A_{X_i}))$ is the number of times a certain instantiation is present in the equivalence class divided by the number of members in that class.

The problem with this approach is that we need to know the exact number of elements in the equivalence class. As far as we know there is no exact way of calculating the number of elements in the equivalence class of a certain DAG, except for enumerating them. We assume that any realistic problem for which our technique can be used requires a limited number of experiments otherwise the cost would be too high. Secondly, since we can work on undirected substructures separately, we do not need the all members of the equivalence class of the complete DAG but just the equivalence classes of the substructures. We assume that either number of DAGs in the equivalence class is practically calculatable by enumerating all members.

4.1.4 Performing Structural Experiments

When performing an experiment on X_i (cf. Section 3.3.2) we will force it to take on random values, hence removing all influence from any other variable in the system. Given that we know the correct CPDAG we know that the only variables that can be directly influenced by X_i are variables that are connected to X_i in the CPDAG by an undirected edge. So instead of measuring all variables in the system we will only measure the $Ne_U(X_i)$. $Ne_U(X_i)$ contains the union of the parents and children of X_i , since the connection between the parents of X_i and X_i is cut by the randomization, the value of X_i will have no influence on the value of the parents of X_i .

4.1.5 Result Analysis

After performing the experiment we have experimental data about all the variables in $Ne_U(X_i)$. This gives us the conditional distribution $P(X_j|do(x_i))$ for all X_j in $Ne_U(X_i)$, with $do(x_i)$ being the experiment on X_i where it takes on the value x_i .

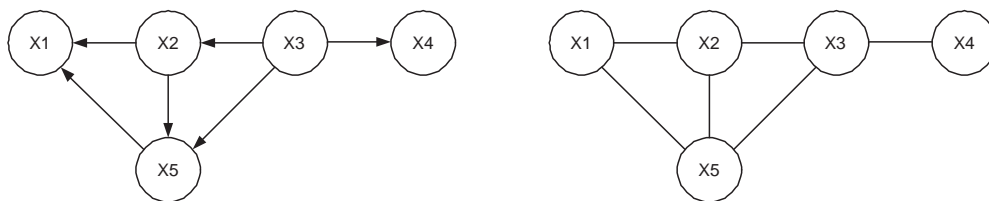


Figure 4.2. Toy CBN (left) and the corresponding CPDAG (right) used in the example.

If X_i has no influence on X_j in the mechanism under study then $P(X_j|do(x_i)) = P(X_j)$ for all values x_i . If X_i does influence X_j then at least for one value x_i the distributions $P(X_j)$ and $P(X_j|do(x_i))$ will differ.

To check whether $P(X_j) = P(X_j|do(x_i))$ we need to perform a statistical test like χ^2 to compare two sets of samples. One sample consists of the values of X_j in the original observational data and the other is the set of samples from the experiment.

Remember that any undirected edge $X_i - X_j$ in the partially directed structure represents either $X_i \rightarrow X_j$ or $X_i \leftarrow X_j$ because we work in a causally sufficient setting (no latent variables). So if the test shows that X_i has an influence on X_j then we will direct the edge $X_i \rightarrow X_j$ and when no change is found $X_i \leftarrow X_j$.

4.1.6 Detailed Learning Algorithm

All the different phases discussed above combine to form an adaptive algorithm, described in Algorithm 1, in which it is assumed that experiments are performed during learning. After an experiment is performed we gain information on the direction of certain edges, this may remove the need to perform certain other experiments. Remember that parts of the graph that are not connected by undirected links can be treated separately, so multiple instances of the algorithm can be applied in parallel to the substructures.

4.1.7 Toy Example

We will show by example the working mechanism of the different criteria.

Reference Model and Hypotheses

Assume the CBN modeling the system is given on the left in Figure 4.2, then the correct CPDAG, which is given as input to our algorithm is shown on the right in Figure 4.2.

To simplify the calculations in this example we will assume the following cost assignments:

$$Cost(A_{X_i}) = 1, \forall i$$

$$Cost(M_{X_i}) = \#Ne_U(X_i)$$

Algorithm 3 Adaptive learning of CBN from observational and experimental data.**Require:** Observational dataset D_O .**Ensure:** A CBN.

1. Apply a BN learning structure on D_O to obtain CPDAG G .
2. Compute $U(A_{X_i})$ for each node X_i and find $X_{best} = \text{argmax}(U(A_{X_i}))$
3. Perform an experiment on variable X_{best} .
4. Instantiate the orientation of all the undirected edges connected to X_{best}
 - For all $X_j \in Ne_U(X_{best})$
 - If distribution of X_j changed because of experiment,
 - then orient $X_{best} - X_j$ as $X_{best} \rightarrow X_j$
 - else orient $X_{best} - X_j$ as $X_{best} \leftarrow X_j$
 - end
5. Infer edges (Orientation rules of PC, see Algorithm 2)
 - Repeat
 - If $X_i \rightarrow X_j$ and X_j and X_k are adjacent, X_k and X_i are not and there is no arrow into X_k then orient $X_j - X_k$ as $X_j \rightarrow X_k$.
 - If there is a directed path from X_i to X_j and an edge between X_i and X_j then orient $X_i - X_j$ as $X_i \rightarrow X_j$.
 - until no more edges can be oriented.
6. Return to Step (2) until all links are directed.
7. Return CBN G .

Comparison of General Decision Criteria

We will assume that we want to perform an action on X_1 . In Figure 4.3 we give an overview of all possible instantiations of the edges $X_1 - Ne_U(X_1)$ and the edges that can be inferred. With this information we will be able to calculate the utility for this action for each decision criteria.

The cost is equal for all decision criteria, namely $Cost(A_{X_1}) + cost(M_{X_1}) = 1 + 2 = 3$.

The complexity of the calculation of the utilities for each decision criteria is dependent on two phases. First, we need to search among all instantiations of $X_i - Ne_U(X_i)$ for the set of possible instantiations in the particular PDAG. Second, we need to check which edge directions can be inferred based on this instantiation. Both these phases are simple searches for patterns in a graph and can be performed efficiently. All decision criteria operate in the same way and hence have the same type of complexity.

Maximax

The maximum number of inferred edges is 4, so the utility for Maximax is given by:

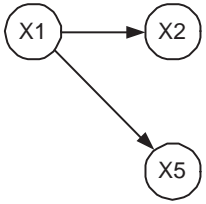
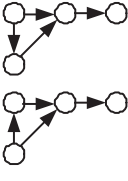
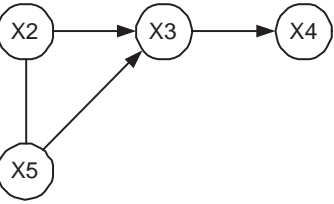
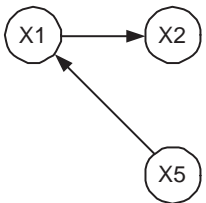
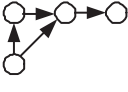
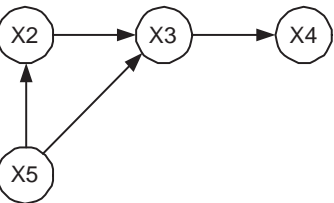
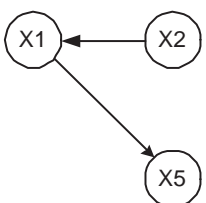
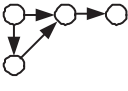
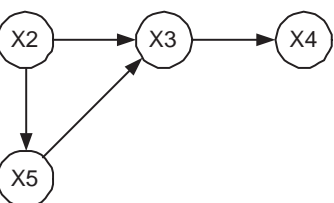
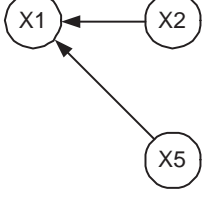
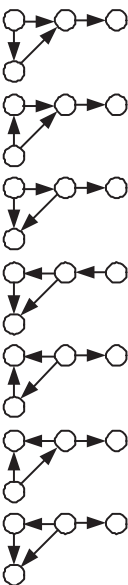
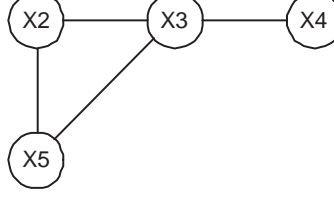
| Instantiations | Possible structures | Result of edge inference | #inferred | P(inst) |
|---|---|--|-----------|---------|
|  |  |  | 3 | 2/11 |
|  |  |  | 4 | 1/11 |
|  |  |  | 4 | 1/11 |
|  |  |  | 0 | 7/11 |

Figure 4.3. All possible instantiations for $X_i - Ne_U(X_i)$, the possible structures compatible with this instantiation and the result of inferring edges.

$$U(X_1) = \left(\frac{\#Ne_U(X_1) + \max_{inst(A_{X_1})} (\#inferred(inst(A_{X_1})))}{cost(A_{X_1}) + cost(M_{X_1})} \right) = \frac{2+4}{1+2} = \frac{6}{3} = 2$$

Maximin

The worst number of edges inferred is 0, so the utility for Maximin is given by:

$$U(X_1) = \left(\frac{\#Ne_U(X_1) + \min_{inst(A_{X_1})} (\#inferred(inst(A_{X_1})))}{cost(A_{X_1}) + cost(M_{X_1})} \right) = \frac{2+0}{1+2} = \frac{2}{3}$$

Expected utility

To use the Expected utility criterion, given in Equation 4.1, we need to calculate the number of graphs in the equivalence class of the PDAGs we encounter.

As shown in Figure 4.3, there are 11 DAGs in the equivalence class of our example (all the possible structures that can be inferred for all the instantiations). So the Expected utility for X_1 is given by:

$$\begin{aligned} U(X_1) &= \left(\frac{\#Ne_U(X_1) + \sum_{inst(A_{X_1})} \#inferred(inst(A_{X_1})) P_{eq}(inst(A_{X_1}))}{cost(A_{X_1}) + cost(M_{X_1})} \right) \\ &= \frac{2 + 3 \cdot \frac{2}{11} + 4 \cdot \frac{1}{11} + 4 \cdot \frac{1}{11} + 0 \cdot \frac{7}{11}}{1 + 2} \\ &= \frac{26}{33} \end{aligned}$$

4.1.8 Experiments and Results

In this section we show our experimental results. We implemented our algorithm in Matlab, using some existing features from the BNT toolbox¹ from *K. Murphy* and the Structure Learning toolbox² from *P. Leray* and *O. Francois*.

Random Problem (CPDAG) Generation

In order to test our algorithm on a large set of CBNs we implemented a random CBN generator **PMMixed**, proposed by Ide et al. (2004). This generator ensures a uniform distribution on the Bayes net structure and allows to add restrictions on maximum degree, induced width, etc.

¹ <http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html>

² <http://banquiseasi.insa-rouen.fr/projects/bnt-slp/>

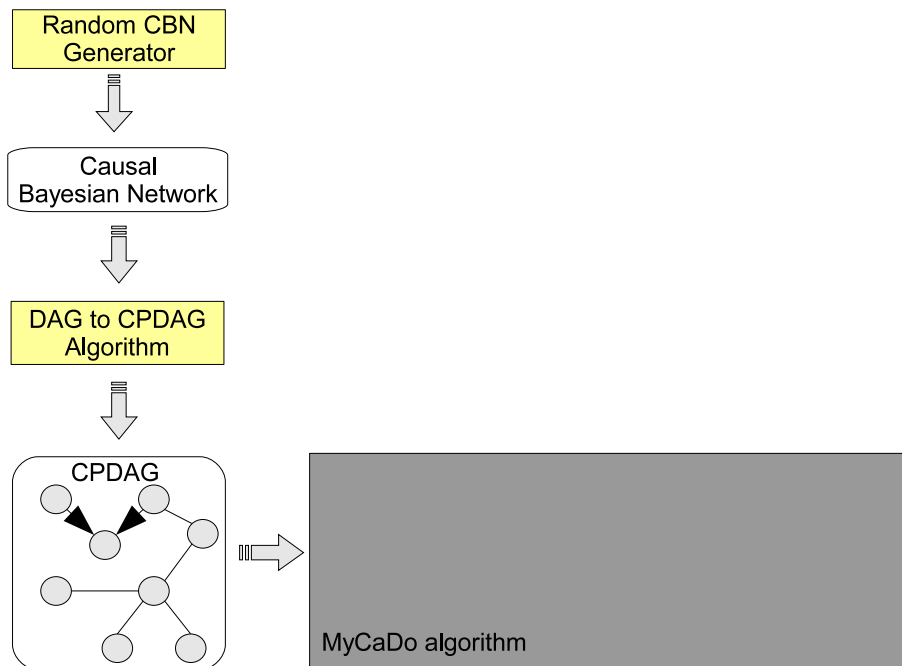


Figure 4.4. CPDAG generation, with an “oracle” providing the exact CPDAG from a known CBN.

Since we are mainly interested in testing our MyCaDo algorithm (Algorithm 3), we first constructed the CPDAG using the *dag_to_cpdag* algorithm from (Chickering, 2002a) that is implemented in BNT as an oracle, see Figure 4.4.

System Intervention

As we are working with random models, we do not have a corresponding system to intervene upon, so we have to simulate this experimentation directly in the previously generated CBN.

To perform the experiment determined in step 3 of our algorithm, we have to cut all influence of other variables on X_i or in other words we have to perform a surgical intervention on X_i . This is done by removing all incoming arrows into X_i from our original structure G_O giving us a post-intervention structure G_E . All parameters besides the one for the variable experimented on, $P(X_i|I_i)$, remain the same. We force X_i to take on random values and then sample the post-intervention distribution P_E for the variables in $Ne_U(X_i)$ using G_E to get our experimental data set D_E , see Figure 4.5.

Results

For all our results, we randomly created 50 DAGs for a given number of nodes 10, 12, ..., 50 variables with different densities. We assigned random costs to each experiment and

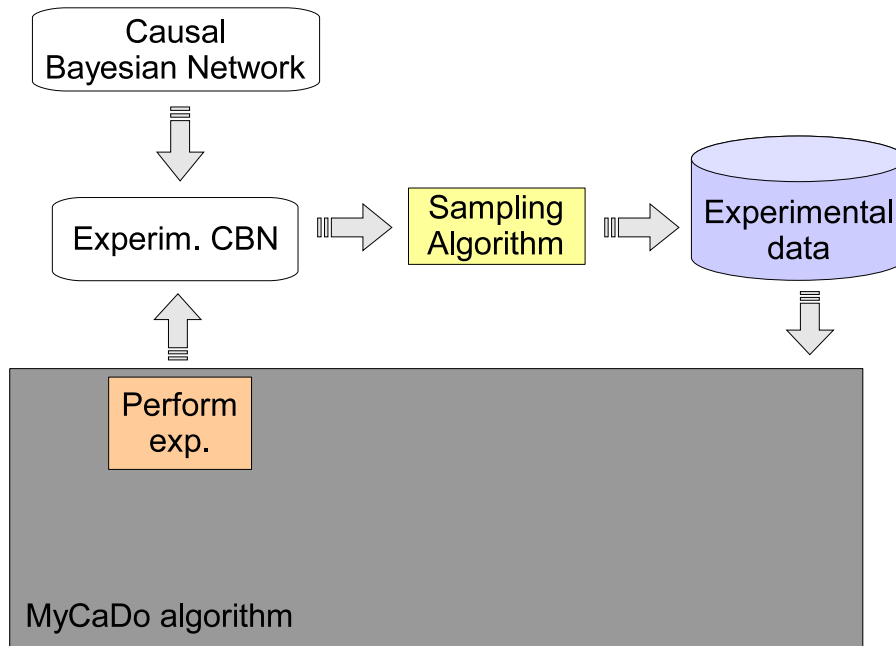


Figure 4.5. Experimental data generation from a known CBN.

measurement, uniformly distributed over 1 (cheap), 5 (normal), 100 (expensive). Our algorithm returns the completely directed CBN and we count the number of experiments needed, undirected substructures and undirected edges in the corresponding CPDAG and we calculate the total cost in order to evaluate our algorithm.

It is clear from our results that the theoretical upper bound of $N - 1$ experiments (when only manipulating one variable) derived by Eberhardt et al. (2005a) is much higher than the amount of experiments needed in the general case. Nonetheless we want to emphasize that in the worst case scenario, that is a clique graph, the upper bound will be attained.

The amount of experiments needed is very dependent on the situation, and not solely on the number of variables in the graph. We have to keep in mind that each individual undirected substructure requires at least one experiment to be completely oriented. So total amount of experiments needed is dependent on the number of undirected substructures and the individual complexity of each substructure.

We should note that the number of experiments is not the value we want to minimize directly. We hope to minimize it indirectly by maximizing the number of edges that will be directed by each experiment and minimizing the total cost of the experiments (which is related to the number of experiments).

For an arbitrary problem, we define the complexity by the maximal undirected clique in all the substructures. If we have a maximal undirected clique of M variables then we need $M - 1$ experiments in the worst case to solve this substructure. All other substructures will

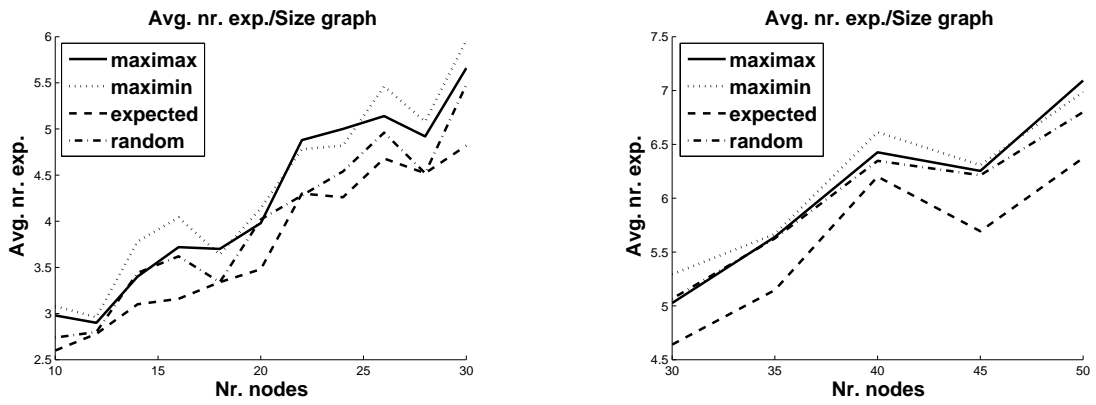


Figure 4.6. Plots of the number of experiments given the number of nodes in the graph, for relatively small graphs ($\#nodes \leq 30$) on the left, and bigger graphs ($30 \leq \#nodes \leq 50$) on the right.

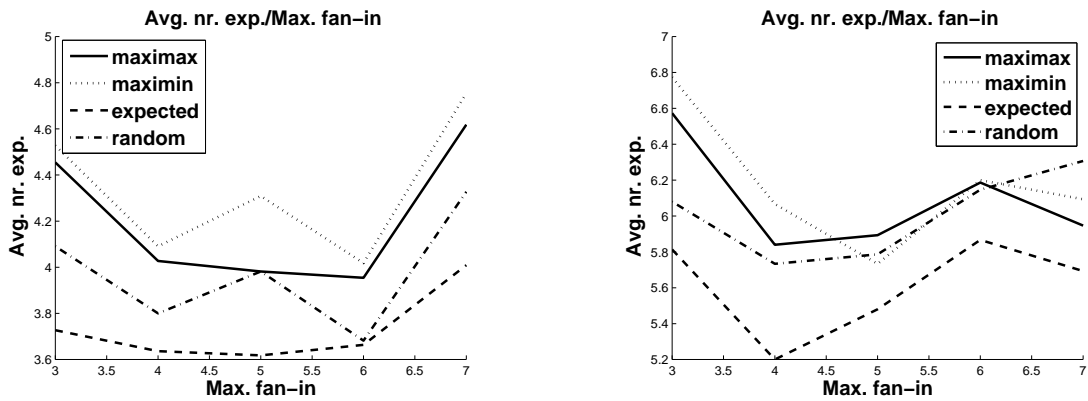


Figure 4.7. Plots of the number of experiments given the maximum number of parents for any node, for relatively small graphs ($\#nodes \leq 30$) on the left, and bigger graphs ($30 \leq \#nodes \leq 50$) on the right.

| alg. \ #nodes | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 35 | 40 | 45 | 50 |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| maximax | 24 | 30 | 24 | 25 | 23 | 27 | 16 | 16 | 26 | 19 | 19 | 23 | 22 | 22 | 17 |
| maximin | 22 | 26 | 19 | 16 | 24 | 20 | 21 | 22 | 17 | 16 | 18 | 20 | 20 | 17 | 18 |
| expected | 38 | 34 | 34 | 45 | 34 | 40 | 30 | 39 | 35 | 30 | 41 | 36 | 28 | 32 | 30 |
| random | 34 | 32 | 25 | 27 | 34 | 26 | 31 | 29 | 25 | 27 | 22 | 21 | 27 | 22 | 23 |

Table 4.1. Table presenting the exact results of our experiments comparing the different criteria based on the number of nodes in the graph. The values indicate the number of times that particular criteria needed the least amount of experiments to completely orient the graph. The total amount of graphs for each number of nodes is 50. Sometimes different criteria result in the same amount of experiments so the total does not add up to 50.

be solved using less experiments and can be handled in parallel. So the overall number of experiments is bounded by $\#(\text{undirected substructures}) * (M - 1)$.

In Figure 4.6 we plotted the average number of experiments needed to completely orient a dag with a given amount of nodes. The numerical results are given in Table 4.1. In Figure 4.7 we plotted the average number of experiments compared to the maximum number of

parents a node can have in the dag. The numerical results for this are given in Table 4.2. In both tables we indicated the best result in bold, and as can be seen this is always accomplished by the expected utility.

| alg. \ max. fan in | 3 | 4 | 5 | 6 | 7 |
|--------------------|-----------|-----------|-----------|-----------|-----------|
| maximax | 37 | 54 | 55 | 61 | 42 |
| maximin | 40 | 54 | 39 | 50 | 38 |
| expected | 83 | 78 | 84 | 74 | 81 |
| random | 54 | 74 | 58 | 70 | 56 |

Table 4.2. Table presenting the exact results of our experiments comparing the different criteria based on the maximum number of parents a node can have in the graph. The values indicate the number of times that particular criteria needed the least amount of experiments to completely orient the graph. The total amount of graphs for each maximum number of parents is 100. Sometimes different criteria result in the same amount of experiments so the total does not add up to 100.

It is clear that the Maximax and Maximin criterion perform quite poorly, we believe that the reason the use of these criteria performs less is that it is biased to choose particular parts of the graph. We even see that random experiments perform as well as those chosen by Maximax or Maximin. Expected utility tries to find an optimal solution that takes into account the probability of each instantiation and clearly outperforms the other algorithms.

We performed experiments with different cost assignments, but the results of all these experiments were similar to those shown in this dissertation.

4.1.9 Critical Discussion of MyCaDo

In this section, we proposed the MyCaDo algorithm, which is a greedy constraint based approach. It can use standard score-based or constraint-based learning methods to learn an initial CPAG and then use experiments to completely learn the structure of a CBN.

We use a mixture of experimental and observational data, which is assumed to perfectly reflect the (in)dependence relationships in the distribution. Furthermore we work under the assumption of causal sufficiency.

In the following section we discuss an algorithm that removes the requirement of perfect observational data. In the following chapters we will discuss setting without causal sufficiency.

4.2 Constraint-based Method for Imperfect Observational Data

Existing structure learning methods, including MyCaDo, are based on the assumption of perfect data. In this section we discuss a new algorithm, called *UnCaDo* (*Unsure Causal Dis-*

covery) which is a structural experiment strategy for settings in which the available observational data does not provide enough information to learn the correct CPDAG. We propose a constraint-based technique and assume causal sufficiency. Constraint-based methods are based on statistical tests which are linked to a notion of uncertainty about their results (i.e. p-value). Instead of using a fixed cut-off for the result of a statistical test we use this uncertainty explicitly. It is not immediately clear how these results can be extended to score-based algorithms but it is an interesting perspective to study this possibility.

4.2.1 General Description

The strategy consists of three phases. First an undirected dependence structure is learned using the observational data, possibly given rise to some unsure relations between nodes. Secondly all these unsure relations are removed from the graph. In the final phase, possible remaining undirected edges are oriented.

The general overview of the algorithm is given in Figure 4.8.

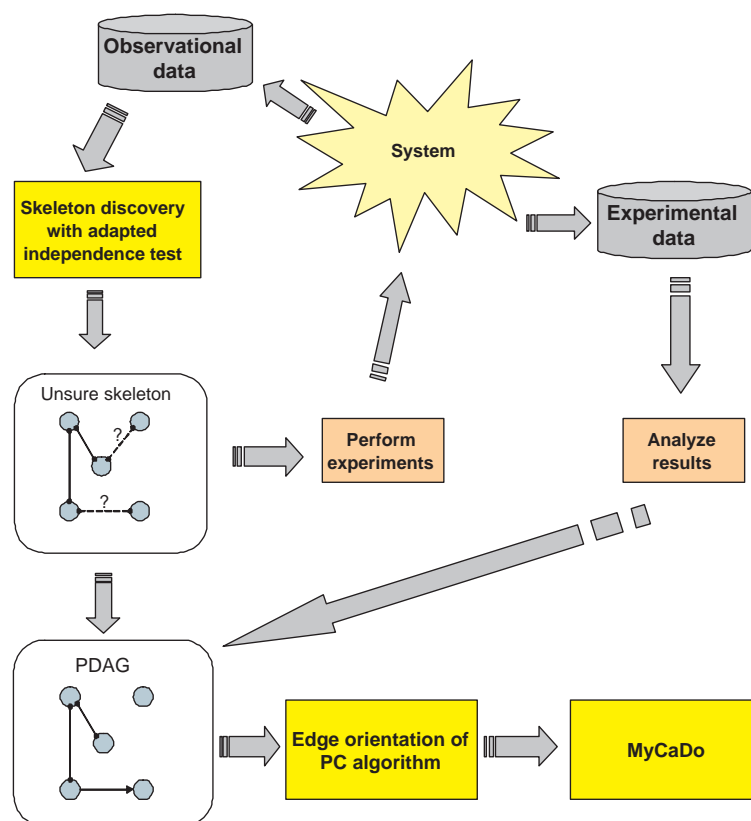


Figure 4.8. General overview of the UnCaDo algorithm.

4.2.2 Unsure Independence Test

The learning techniques we use are based on conditional (in)dependence tests. These tests need a certain amount of data in order to be reliable. If, for instance, not enough data was available then we can not draw a conclusion on the (in)dependence of two variables and thus are unsure whether an edge should be removed or added to the current graph. However most implementations of independence tests provide a standard answer in case there is not enough data. We propose an adapted independence test which in case there is not enough data or not enough evidence (this can be user dependent) for (in)dependence returns *unsure* as a result.

There are several ways to adapt existing independence tests, for instance:

- Using χ^2 , the number of data points has to be more than $10 * (\text{degree of freedom})$, otherwise the test can not be performed reliably as mentioned by Spirtes et al. (2000a). In most implementations "no conditional independence" is returned as default, while in our case *unsure* would be returned.
- There can be an interval for the significance level used to return *unsure*. Traditionally tests are done using $\alpha = 0.05$ significance level. We allow to set two parameters α_1 and α_2 with $\alpha_1 > \alpha_2$. We return independence if the test is significant for significance level α_2 and unsure if it is significant for α_1 and not for α_2 . For example *unsure* could be returned where the test is significant with $\alpha_1 = 0.05$, but no longer insignificant for $\alpha_2 = 0.02$.

We will use experiments to resolve these unsure relationships. If the current data we have does not obtain enough information to reliably perform the statistical tests we use experiments to gather more data hoping to resolve this uncertainty.

There is a parallel here with sequential hypothesis testing Wald (1945), where it is assumed that the observations arrive sequentially rather than all at once. Data is evaluated as it is collected, and further sampling is stopped in accordance with a pre-defined stopping rule as soon as significant results are observed. At each point during sequential hypothesis testing we need to determine whether a certain model is valid or that more data are required. As long as the provided data is not sufficient to either accept or decline a hypothesis we get more data prior to deciding.

4.2.3 Initial Phase: *Unsure* Observational Learning

We use the adapted independence test and modify the skeleton discovery phase of the PC algorithm, Steps 1 and 2 in Algorithm 2 in order to form an *unsure skeleton*. This conforms to Steps 1, 2 and 3 in Algorithm 4.

Definition 4.1. *An unsure skeleton is an undirected graph in which the nodes can have three graphical relations, based on the results of our adapted independence test:*

no edge: X_i and X_j are found to be independent conditional on some subset (possibly the empty set).

edge $X_i o-o X_j$: X_i and X_j are dependent conditional on all subset of variables and all conditional independence tests returned false. This corresponds to the traditional undirected edge $X_i - X_j$ and hence means either $X_i \leftarrow X_j$ or $X_i \rightarrow X_j$.

unsure edge $X_i o-?-o X_j$: we can not determine whether X_i and X_j are (in)dependent, i.e. there exists at least one subset of variables for which the independence test returned unsure and none that return independent. This means that either $X_i \not\sim X_j$, $X_i \rightarrow X_j$ or $X_i \leftarrow X_j$.

The semantics of the edges in an unsure skeleton allows us to replace the independence test used in PC with our adapted independence tests. The classic independence test returns either *true* or *false*, using our adaptation there is a third possible response *unsure*. When independence is found the edge between the two nodes is removed as usual, however when the relation between the two variables is unsure, we include a new type of edge $o-?-o$. In order to find sets to test for conditional independence, arrows of type $o-?-o$ are regarded as normal undirected edges. denoted as $o-o$.

Note that as the number of data points $N \rightarrow \infty$ the *unsure* edges will disappear, since we will work with perfect data. In general however when there are *unsure* edges more data is needed to distinguish between independence and dependence, therefore we are in need of experiments.

4.2.4 Experimentation Phase: Resolving Unsure Edges

In this section we show how we can resolve *unsure* edges using experiments. We denote performing an experiment at variable X_i by $exp(X_i)$.

In general if a variable X_i is experimented on and the distribution of another variable X_j is affected by this experiment, we say that X_j varies with $exp(X_i)$, denoted by $exp(X_i) \rightsquigarrow X_j$. If there is no variation in the distribution of X_j we note $exp(X_i) \not\rightsquigarrow X_j$.

If we find when comparing the observational with the experimental data that $exp(X_i) \rightsquigarrow X_j$ when we condition the test on the value of another variable Z , we denote this as $(exp(X_i) \rightsquigarrow X_j)|Z$, if conditioning on Z cuts the influence of the experiment we denote it as $(exp(X_i) \not\rightsquigarrow X_j)|Z$.

We introduce additional notation to indicate that two nodes X_i and X_j are either not connected or connected by an arc into X_j , we denote this by $X_i -?-o X_j$, where “|” indicates that there can be no arrow into X_i .

If we take a look at the simplest example, a graph existing of only two variables X_i and X_j for which our initial learning phase gives $X_i o-?o X_j$. After performing an experiment on X_i and studying the data we can conclude one of three things:

1. $X_i o-?o X_j$
2. $exp(X_i) \rightsquigarrow X_j \Rightarrow X_i \rightarrow X_j$
3. $exp(X_i) \not\rightsquigarrow X_j \Rightarrow X_i o-?o X_j$

The first case happens if the added experiments still do not provide us with enough data to perform an independence test reliably. We can repeat the experiment until sufficient data is available (χ^2) or the test can be performed at our desired significance level (i.e. $\alpha = 0.02$). If no sufficient experiments can be performed the link remains $X_i o-?o X_j$, this possibility is a part of future work. The second possibility is the ideal one, in which we immediately find an answer for our problem.

In the third case the only conclusion we can make is that X_i is not a cause of X_j and hence there is no arrow $>$ into X_j . To solve this structure completely we still need to perform an experiment on X_j . So in this case the results of performing an experiment at X_j are:

4. $exp(X_j) \rightsquigarrow X_i \Rightarrow X_i \leftarrow X_j$
5. $exp(X_j) \not\rightsquigarrow X_i + (3) \Rightarrow X_i \ X_j$

In a general graph there can be more than one path between two nodes, and we need to take them into account in order to draw conclusions based on the results of the experiments. This means that we have to generalize rules 2 to 5 presented above.

Therefore we introduce the following definition:

Definition 4.2. A *potentially directed path* (p.d. path) in an unsure PDAG is a path made of edges of types $o-?o$, \rightarrow and $-?o$, with all arrowheads in the same direction. A p.d. path from X_i to X_j is denoted as $X_i \dashrightarrow X_j$.

If in a general unsure PDAG there is an edge $X_i o-?o X_j$, the results of performing an experiment at X_i are:

6. $exp(X_i) \rightsquigarrow X_j \Rightarrow X_i \dashrightarrow X_j$, but since we want to find direct effects we need to block all p.d. paths of length ≥ 2 by a blocking set \mathbf{Z} (Generalization of (2)).
 - $(exp(X_i) \rightsquigarrow X_j) | \mathbf{Z} \Rightarrow X_i \rightarrow X_j$
 - $(exp(X_i) \not\rightsquigarrow X_j) | \mathbf{Z} \Rightarrow X_i \ X_j$
7. $exp(X_i) \not\rightsquigarrow X_j \Rightarrow X_i o-?o X_j$ (Generalization of (3)).

In the case that $exp(X_i) \not\rightsquigarrow X_j$ we have to perform an experiment at X_j too. The results of this experiment are:

8. $exp(X_j) \rightsquigarrow X_i \Rightarrow X_i \leftarrow X_j$, but since we want to find direct effects we need to block all p.d. paths of length ≥ 2 by a blocking set \mathbf{Z} (Generalization of (4)).
- $(exp(X_j) \rightsquigarrow X_i) | \mathbf{Z} \Rightarrow X_i \leftarrow X_j$
 - $(exp(X_j) \not\rightsquigarrow X_i) | \mathbf{Z} \Rightarrow X_i \rightarrow X_j$
9. $exp(X_j) \not\rightsquigarrow X_i + (7) \Rightarrow X_i \rightarrow X_j$ (Generalization of (5)).

After these experiments all *unsure* edges $X_i o \text{---} o X_j$ are transformed into either directed edges or are removed from the graph.

It has to be noted that, like in the simplest example, the experiments only provide us with more data and that this still might not be enough to give a reliable answer for our statistical test (χ^2). In this case the result of an experiment would leave the *unsure* edge and more experiments are needed until the test can be performed reliably.

4.2.5 Completion Phase

At this point there are only the original undirected edges $o \text{---} o$ and directed edges \rightarrow found by resolving *unsure* edges, we can hence use Steps 3 and 4 of Algorithm 2 to orient as many edges in the current PDAG, this conforms to Step 6 in Algorithm 4. If not all arcs are directed after this we need to perform another set of experiments. In order to complete this we use the MyCaDo algorithm discussed in Section 4.1, this conforms to Step 7 in Algorithm 4.

4.2.6 Complete Learning Algorithm

All the actions described above combine to form the Unsure Causal Discovery algorithm (UnCaDo). The complete algorithm is given in Algorithm 4. We define a couple of notions to simplify the notation. In an unsure graph, $Ne(X_i)$ are all variables connected to X_i either by a directed, undirected or unsure edge. Test of independence are performed using the unsure independence test introduced in Section 4.2.2, in Algorithm 4 this test is referred to as the test of independence.

4.2.7 Toy Example

We demonstrate the different steps of the UnCaDo algorithm on a simple example. If the unsure independence test returns "unsure" for a test between X_i and X_j conditioned on some set Z we note this as $(X_i \not\perp\!\!\!\perp X_j | Z)$. Assume the correct CBN is given in Figure 4.9(a). The algorithm starts with a complete undirected graph shown in Figure 4.9(b). Assuming that the first ordered pair of variables that will be checked is (X_1, X_3) and that we find the following (in)dependence information:

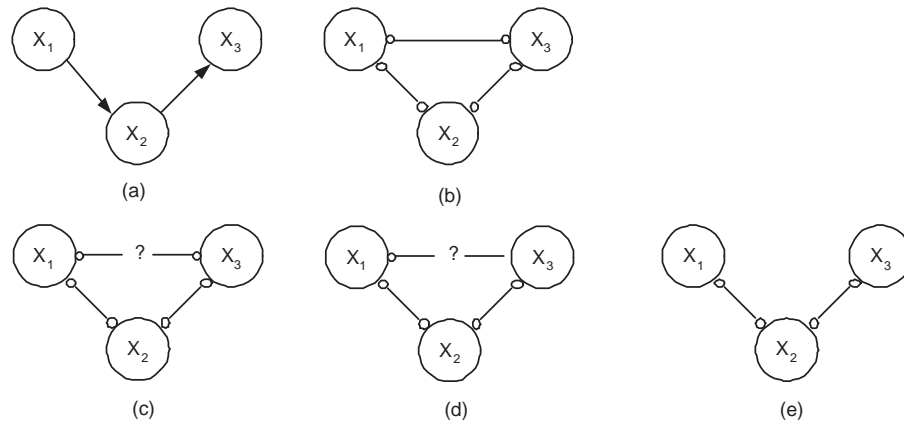


Figure 4.9. Simple example demonstrating the different steps in the UnCaDo algorithm.

- $(X_1 \perp\!\!\!\perp X_3)$
- $(X_1 \perp\!\!\!\perp X_3 | X_2)$

This means that the edge $X_1 \circ - \circ X_3$ will be replaced by $X_1 \circ - ? - \circ X_3$, cf. Figure 4.9(c). To check for (in)dependence between the other sets of variables (X_1, X_2) and (X_2, X_3) we regard the unsure edge $X_1 \circ - ? - \circ X_3$ as being a normal undirected edge. This means that we need to check for both the marginal as the conditional independence of these pairs. If the edge would be considered absent this might lead to missing necessary independence tests. Assume that we find the following independence information for (X_1, X_2) and X_2, X_3 :

- $(X_1 \perp\!\!\!\perp X_2)$
- $(X_1 \perp\!\!\!\perp X_2 | X_3)$
- $(X_2 \perp\!\!\!\perp X_3)$
- $(X_2 \perp\!\!\!\perp X_3 | X_1)$

So at the end of our non-experimental phase we end up with the structure given in Figure 4.9(c).

We now need to perform experiments in order to remove the unsure edge $X_1 \circ - ? - \circ X_3$. Assume we choose to perform an experiment on X_1 and gather all data D_{exp} . There is a p.d. path $X_1 \circ - \circ X_2 \circ - \circ X_3$ so we have to compare all conditional distributions to see whether there was an influence of $exp(X_1)$ at X_3 . Hence we find that:

- $(exp(X_1) \not\rightarrow X_3 | X_2)$

and thus we can replace the edge $X_1 \circ - ? - \circ X_3$ by $X_1 \circ - ? - X_3$ as shown in Figure 4.9(d). Now we need to perform an experiment on X_3 , taking into account the p.d. path $X_3 \circ - \circ X_2 \circ - \circ X_1$. We find that:

- $(exp(X_3) \not\rightarrow X_1 | X_2)$

and we can remove the edge $X_1 \circ \text{---} X_3$, leaving us the graph shown in Figure 4.9(e). Now that all unsure edges are resolved we can use the orientation rules of the PC-algorithm (Algorithm 2), including the search for v-structures which in some cases will immediately find the correct structure if enough data is available or we need to run the MyCaDo algorithm to complete the structure.

4.2.8 Critical Discussion of UnCaDo

Although the algorithm allows the use of imperfect observational data, it is assumed that the experiments we perform will quickly give us a definite answer about the edges in the graph. It is possible that certain experiments are hard or even impossible to perform and that we can not resolve all unsure edges.

A part of future work is to go further and allow unsure edges in the CPDAG and in the eventual DAG structure. We can test which conclusions about the structure can still be made reliably when we allow unsure edges.

Furthermore we would like to extend this approach to a setting in which latent variables are allowed. These causal latent models, as described in Section 1.4.1, can represent a much wider set of problems.

4.3 Overview of the Chapter

In this chapter, we discussed learning the structure of a CBN. In general, the structure can not be retrieved from observational data alone and hence experiments are needed. We gave an overview of state-of-the-art existing algorithms and their assumptions and properties.

We proposed the MyCaDo algorithm, which is a greedy constraint based approach, to learn causal Bayesian networks from a mixture of perfect experimental and observational data. We used the information of observational data to learn a complete partially directed graph and tried to discover the directions of the remaining edges by means of experiment. We used elements from decision theory to find the optimal experiment, at each step of the learning process. Our method supports the assignment of costs to experiments and measurements.

We extended the existing methods to include the possibilities of imperfect data, this is the case when observational data insufficient to learn the correct skeleton of the network. Therefore we proposed an adapted (in)dependence test which can return unsure if the (in)dependence can not be detected reliably. We suggested to change the skeleton discovery phase of the PC algorithm in order to be able to include the adapted (in)dependence test. We proposed a new graph, an unsure graph, which can represent the results of the new

discovery phase by means of unsure edges. We then showed how these unsure connections can be replaced either by a cause-effect relation or removed completely from the graph during an experimentation phase. Our UnCaDo strategy indicates which interventions need to be performed to transform the unsure DAG into a PDAG in which all directed links present direct causal influence. Using a combination of the orientation rules of (Meek, 1995) and, if necessary, some experiments, this PDAG can then be turned into the correct CBN.

Algorithm 4 Adaptive learning of CBN for imperfect observational data and experiments.**Require:** A set of samples from a probability distribution faithful to a CBN $P(V)$.**Ensure:** A CBN.

1. Initialization:
 - G is complete undirected graph on the vertex set V .
2. *Unsure* skeleton discovery:
 - $n=0$
 - repeat
 - repeat
 - select an ordered pair of variables X_i and X_j that are adjacent in G such that $Ne(X_i) \setminus X_j$ has cardinality greater than or equal to n , and a subset S of $Ne(X_i) \setminus X_j$ of cardinality n . If X_i and X_j are independent given S delete edge $X_i - X_j$ from G and record S in $Sepset(X_i, X_j)$ and $Sepset(X_j, X_i)$;
 - If the independence test returned *unsure*, record the tuple (X_i, X_j) into $PossibleUnsureEdges$.
 - until all ordered pairs of adjacent variables X_i and X_j such that $Ne(X_i) \setminus X_j$ has cardinality greater than or equal to n and all subsets S of $Ne(X_i) \setminus X_j$ of cardinality n have been tested for independence;
 - $n=n+1$;
 - until for each ordered pair of adjacent vertices X_i, X_j , $Ne(X_i) \setminus X_j$ is of cardinality less than n .
3. For each tuple (X_i, X_j) in $PossibleUnsureEdges$, if the edge $X_i - X_j$ is still present in G replace that edge by $X_i o-? o X_j$.
4. Resolving *unsure* edges and edge orientation from experiments:
 - For each unsure edge $X_i o-? o X_j$,
 - Perform experiment at X_i ,
 - If $exp(X_i) \rightsquigarrow X_j$,
 - Find a (possibly empty) set of variables Z that blocks all p.d. paths between X_i and X_j .
 - If $(exp(X_i) \rightsquigarrow X_j) | Z$ then orient $X_i o-? o X_j$ as $X_i \rightarrow X_j$, else remove the edge.
 - else replace $X_i o-? o X_j$ by $X_i o-?- X_j$.
5. For each edge $X_i o-?- X_j$,
 - Perform experiment at X_j ,
 - If $exp(X_j) \rightsquigarrow X_i$,
 - Find a (possibly empty) set of variables Z that blocks all p.d. paths between X_j and X_i .
 - If $(exp(X_j) \rightsquigarrow X_i) | Z \Rightarrow$ then orient $X_i o-?- X_j$ as $X_i \leftarrow X_j$, else remove the edge.
 - else remove the edge.
6. Edge orientation: inferred edges using the *Sepset* from Step 2.
 - Apply Steps 3, 4 of Algorithm 2.
7. Complete PDAG to get CBN:
 - Apply Steps 2 to 7 of Algorithm 3.

Causal Latent Networks

In this chapter we discuss causal networks with latent variables which are unmeasured variables that influence the system under study. In order to represent the presence of latent variables, we need a richer formalism than DAGs. There are two major paradigms used to model these systems, ancestral graphs (AG) (Richardson and Spirtes, 2002) and semi-Markovian causal models (SMCM) (Tian and Pearl, 2002c). We will discuss both techniques extensively and then show how they can be learned and used. If we again reflect on the requirements of a good causal model (Section 1.2), we have to describe what exactly can be represented by an AG and a SMCM (Section 5.1 and 5.2), which queries we can resolve within these frameworks (Section 5.3), and how they can be learned from observational or experimental data (Section 5.4).

5.1 Ancestral Graphs

Ancestral graphs (AGs) are an approach to modeling which can represent latent variables and unmeasured selection variables developed by (Richardson and Spirtes, 2002). Selection variables are variables that are conditioned upon but not measured in the study when modeling the system. This can lead to dependences between variables in the study, although they are actually marginally independent. For example given a v-structure $X_i \rightarrow X_m \leftarrow X_j$ in which X_m is unmeasured but conditioned upon, X_i will seem marginally dependent on X_j although this is not the case, this is called selection bias. In this dissertation we do not take into account selection bias, as is commonly done in literature, but only latent variables.

5.1.1 Definitions

An AG is defined in (Zhang and Spirtes, 2005b), (Tian, 2005), and (Ali et al., 2005) as follows:

Definition 5.1. An **ancestral graph** without conditioning is a graph with no directed cycle containing directed \rightarrow and bi-directed \leftrightarrow edges, such that there is no bi-directed edge between two variables that are connected by a directed path.

Ancestral graphs encode conditional independence relationships by graphical criterion called m -separation that generalize the d -separation criterion.

Definition 5.2. In an ancestral graph, a path p between two nodes X_i and X_j is m -connecting relative to a set of vertices Z ($X_i, X_j \notin Z$) if

- every non-collider on p is not a member of Z
- every collider on p is an ancestor of some member of Z

X_i and X_j are said to be m -separated by Z if there is no active path between X_i and X_j relative to Z . Let W_1 , W_2 and Z be three disjoint sets of vertices. W_1 and W_2 are said to be m -separated by Z if Z m -separates every member of W_1 from every member of W_2 .

Definition 5.3. An ancestral graph is said to be a **maximal ancestral graph** if, for every pair of non-adjacent nodes X_i, X_j there exists a set Z such that X_i and X_j are m -separated given Z .

Maximality corresponds to the pairwise Markov property which says that every missing edge corresponds to a conditional independence relation. A non-maximal AG can be transformed into a unique MAG by adding some bi-directed edges (indicating confounding) to the model. See Figure 5.1(c) for an example MAG representing the same model as the underlying DAG in (a).

5.1.2 Semantics

In this setting a directed edge represents an ancestral relation in the underlying DAG with latent variables. I.e. an edge from variable X_i to X_j represents that in the underlying causal DAG with latent variables, there is a directed path between X_i and X_j .

Bi-directed edges represent a latent common cause between the variables. However, if there is a latent common cause between two variables X_i and X_j , and there is also a directed path between X_i and X_j in the underlying DAG, then in the MAG the ancestral relation takes precedence and a directed edge will be found between the variables. $X_2 \rightarrow X_6$ in Figure 5.1(c) is an example of such an edge.

Furthermore, as MAGs are maximal, there will also be edges between variables that have no immediate connection in the underlying DAG, but that are connected via an inducing path. The edge $X_1 \rightarrow X_6$ in Figure 5.1(c) is an example of such an edge.

Definition 5.4. An **inducing path** is a path in a graph such that each observable non-endpoint node is a collider, and an ancestor of at least one of the endpoints.

Inducing paths have the property that their endpoints can not be separated by conditioning on any subset of the observable variables. For instance, in Figure 5.1(a), the path $X_1 \rightarrow X_2 \leftarrow L_1 \rightarrow X_6$ is inducing.

5.1.3 Advantages/Disadvantages of AG

The semantics of the edges in a MAG make some causal inferences in MAGs impossible. As we have discussed before the *manipulation theorem* (cf. Section 3.2.2) states that in order to calculate the causal effect of a variable C on another variable E , the immediate parents (i.e. the old causes) of C have to be removed from the model. In MAGs however, an edge does not necessarily represent an immediate causal relationship, but rather an ancestral relationship and hence in general the modeler does not know which are the real immediate causes of a manipulated variable.

An additional problem for finding the original causes of a variable in MAGs is that when there is an ancestral relation and a latent common cause between variables, that the ancestral relation takes precedence and that the confounding is absorbed in the ancestral relation.

The benefit of MAGs is that the structure of a MAG can be learned up to Markov equivalence from data, see Section 5.4.2.

5.2 Semi-Markovian Causal Models

The other central graphical modeling representation that we use are semi-Markovian causal models. They were first used by Pearl (2000), and Tian and Pearl (2002b) have developed causal inference algorithms for them.

5.2.1 Definitions

A SMCM is defined as follows:

Definition 5.5. A *semi-Markovian causal model* (SMCM) is an acyclic causal graph G with both directed and bi-directed edges. The nodes in the graph represent observable variables $\mathbf{V} = \{X_1, \dots, X_n\}$ and the bi-directed edges implicitly represent latent variables $\mathbf{L} = \{L_1, \dots, L_{n'}\}$.

See Figure 5.1(b) for an example SMCM representing the underlying DAG in (a).

The fact that a bi-directed edge represents a latent variable, implies that the only latent variables that can be modeled by a SMCM can not have any parents (i.e. is a root node) and has exactly two children that are both observed. This seems very restrictive, however it has been shown that models with arbitrary latent variables can be converted into SMCMs, while preserving the same independence relations between the observable variables (Tian and Pearl, 2002c).

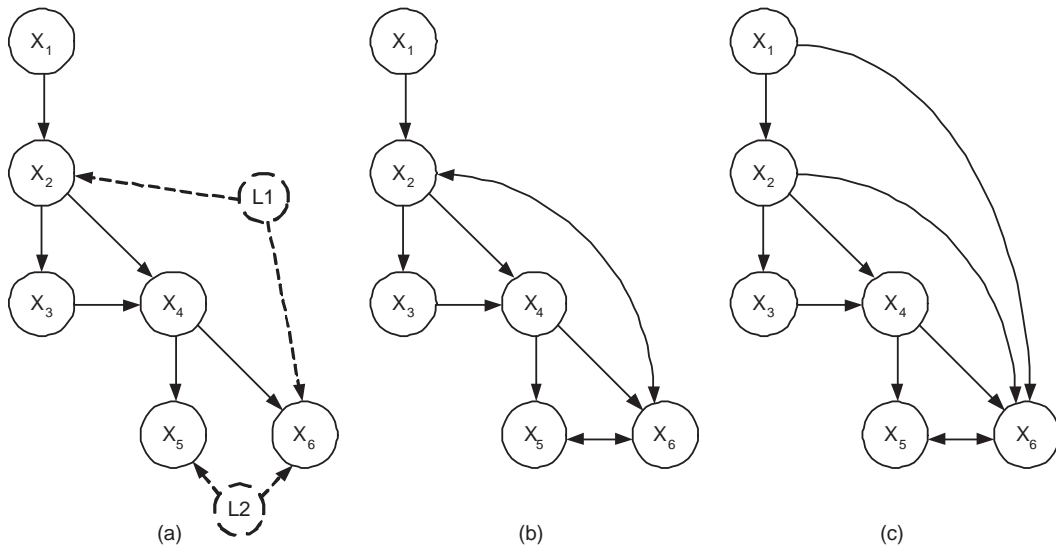


Figure 5.1. (a) A problem domain represented by a causal DAG model with observable and latent variables. (b) A semi-Markovian causal model representation of (a). (c) A maximal ancestral graph representation of (a).

5.2.2 Semantics

In a SMCM, each directed edge represents an immediate autonomous causal relation between the corresponding variables, just as was the case for causal Bayesian networks.

In a SMCM, a bi-directed edge between two variables represents a latent variable that is a common cause of these two variables.

The semantics of both directed and bi-directed edges imply that SMCMs are not maximal, meaning that not all dependencies between variables are represented by an edge between the corresponding variables. This is because in a SMCM an edge either represents an immediate causal relation or a latent common cause, and therefore dependencies due to inducing paths, will not be represented by an edge.

5.2.3 Advantages/Disadvantages of SMCM

Unlike in MAGs it is possible to perform causal inference in SMCMs. Tian and Pearl (2002b) provide a complete causal inference algorithm, which is the current state of the art. We will discuss this algorithm in Section 5.3.

However, prior to our work no algorithm to learn the structure of a SMCM existed. We constructed an algorithm that uses the learning algorithms for MAGs and experiments to recover the structure of a SMCM. The algorithm is introduced in Section 6.1.1.

5.3 Causal Inference

In AGs no complete causal inference algorithm exists due to the semantics of the edges. Causal inference is only possible in parts of the AG that are shared for all Markov equivalent AGs, see Section 5.4.2. Therefore we will only demonstrate causal inference for SMCMs.

The algorithm that we will discuss here, was first introduced by Tian and Pearl (2002b) and it represents the state-of-the-art in causal inference in SMCMs (and hence causal latent networks) at this moment. It is based on a new factorization of the JPD of an SMCM into so-called *c-factors*. Before going into the details of the algorithm we introduce some new concepts.

5.3.1 C-components and C-factors

We say that a path entirely composed of bi-directed edges is called a *bi-directed path*.

Definition 5.6. *In a semi-Markovian causal model, the set of observable variables can be partitioned into disjoint groups by assigning two variables to the same group iff they are connected by a bi-directed path. We call such a group a **c-component** (from "confounded component") (Tian and Pearl, 2002b).*

Variables that are not connected to any bi-directed edge are a c-component by themselves. All c-components of a SMCM are disjoint and constitute a partition.

Furthermore, when we say that unobservable variables belong to the same group if and only if they are part of bi-directed paths connecting two variables belonging to the same c-component, this also induces a partition on the unobserved variables.

Consider for example the SMCM of Figure 5.2, it is partitioned into 5 c-components:

$$\{\{X, X_4, X_6\}, \{X_1\}, \{X_2, X_5\}, \{X_3\}, \{Y\}\}. \quad (5.1)$$

The associated partition of the unobserved variables U_1, U_2, U_3 is as follows:

$$\{\{U_1, U_2\}, \emptyset, \{U_3\}, \emptyset, \emptyset\}. \quad (5.2)$$

Now assume that \mathbf{V} is partitioned into its k c-components $\mathbf{S}_1, \dots, \mathbf{S}_k$ and denote by \mathbf{N}_l the set of unobservable variables that are parents of variables in \mathbf{S}_l . As stated above, $\mathbf{N}_1, \dots, \mathbf{N}_k$ also form a partition of \mathbf{U} .

Definition 5.7. *In a SMCM with observable variables \mathbf{V} and unobservable variables \mathbf{U} , a **c-factor** $Q[\mathbf{S}]$ of a set of observable variables $\mathbf{S} \subset \mathbf{V}$ is the contribution of the variables in \mathbf{S} and the associated*

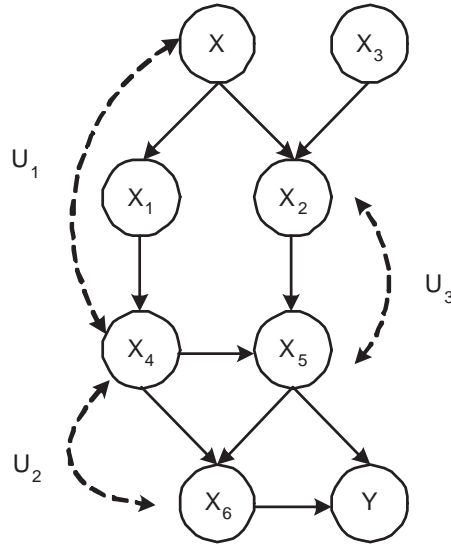


Figure 5.2. A SCMC with 5 c-components.

unobserved variables $\mathbf{N} \subset \mathbf{U}$ (those that are connected to variables in \mathbf{S}), to the mixture of products representing the JPD over \mathbf{V} :

$$Q[\mathbf{S}] = \sum_{\{u_k | U_k \in \mathbf{N}\}} \prod_{X_i \in \mathbf{S}} P(x_i | \Pi_i, UPa(x_i)) \prod_{U_j \in \mathbf{N}} P(u_j) \quad (5.3)$$

If we apply this definition to the c-components $\mathbf{S}_1, \dots, \mathbf{S}_k$ of an SCMC, then the disjointness of these c-components and their associated unobservable variables $\mathbf{N}_1, \dots, \mathbf{N}_k$ implies that the mixture of products for $P(\mathbf{V})$ can be decomposed into a product of c-factors $Q[\mathbf{S}_l]$ of the c-components:

$$\begin{aligned} P(\mathbf{v}) &= \sum_{\{u_m | U_m \in \mathbf{U}\}} \prod_{X_i \in \mathbf{V}} P(x_i | \pi_i, UPa(x_i)) \prod_{U_j \in \mathbf{U}} P(u_j) \quad (5.4) \\ &= \sum_{\{u_{N_1} | U_{N_1} \in \mathbf{N}_1\}} \prod_{X_i \in \mathbf{S}_1} P(x_i | \pi_i, UPa(x_i)) \prod_{U_j \in \mathbf{N}_1} P(u_j) \\ &\quad \sum_{\{u_{N_2} | U_{N_2} \in \mathbf{N}_2\}} \prod_{X_i \in \mathbf{S}_2} P(x_i | \pi_i, UPa(x_i)) \prod_{U_j \in \mathbf{N}_2} P(u_j) \\ &\quad \vdots \\ &\quad \sum_{\{u_{N_k} | U_{N_k} \in \mathbf{N}_k\}} \prod_{X_i \in \mathbf{S}_k} P(x_i | \pi_i, UPa(x_i)) \prod_{U_j \in \mathbf{N}_k} P(u_j) \\ &= \prod_{l=1}^k Q[\mathbf{S}_l] \quad (5.5) \end{aligned}$$

This result implies that the mixture of products can be transformed into a factorization. That is useful as a factorization allows a calculation to be split up into a product of different modular subcalculations, while when confronted with a mixture of products this is not necessary possible.

Furthermore, as a consequence of the *manipulation theorem* for SMCMs, the c-factor of a c-component \mathbf{S}_l is also the post-intervention distribution of the variables in \mathbf{S}_l , under an intervention that sets all other observable variables $\mathbf{V} \setminus \mathbf{S}_l$ to constants (Tian and Pearl, 2002c), or

$$Q[\mathbf{S}_l] = P(\mathbf{s}_l | do(\mathbf{v} \setminus \mathbf{s}_l)) \quad (5.6)$$

Finally, it follows from the definition that

$$P(\mathbf{V}) = Q[\mathbf{V}] \quad (5.7)$$

As an illustration, in the SMCM of Figure 5.2 the c-factors are¹:

$$Q[X, X_4, X_6] = \sum_{\{u_k | U_k \in \{U_1, U_2\}\}} P(x | u_1) P(x_4 | x_1, u_1, u_2) P(x_6 | x_4, x_5, u_2). \quad (5.8)$$

$$P(u_1) P(u_2)$$

$$Q[X_1] = P(x_1 | x)$$

$$Q[X_2, X_5] = \sum_{\{u_3 | U_3\}} P(x_2 | x, x_3, u_3) P(x_5 | x_2, x_4, u_3) P(u_3)$$

$$Q[X_3] = P(x_3)$$

$$Q[Y] = P(y | x_5, x_6)$$

5.3.2 Identification

Before demonstrating the causal inference algorithm we first discuss an identification algorithm which identifies when a causal query can be answered in a given SMCM. The causal identification algorithm that we discuss is based on three important lemmas proved by (Tian and Pearl, 2002b) that will briefly be introduced as properties in this section. We will give a more detailed description of the lemmas in the technical discussion of the identification

¹ For notational convenience we will write $Q[X_i, X_j, X_k]$ instead of $Q[\{X_i, X_j, X_k\}]$

algorithm. See (Tian and Pearl, 2002b) for a complete specification of the lemmas and their proofs.

We need to introduce some additional notation: given a SMCM with graph G and with observable variables \mathbf{V} and unobservable variables \mathbf{U} , then if $\mathbf{W} \subset \mathbf{V}$, $G_{\mathbf{W}}$ is the subgraph of G restricted to observable variables \mathbf{W} and to the unobservable variables \mathbf{H} , whose elements H_1, \dots, H_k are connected with exactly two elements in \mathbf{W} . See Figure 5.3 for an example reduction of the graph in Figure 5.2 to the observed variables $\mathbf{W} = \{X, X_2, X_4, X_6, Y\}$ and the associated unobserved variables $\{U_1, U_2\}$.

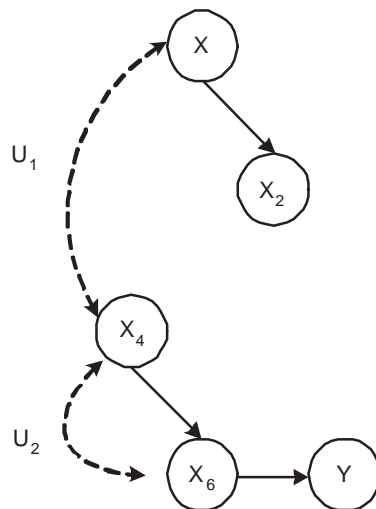


Figure 5.3. The SMCM of Figure 5.2 reduced to the observed variables $\mathbf{W} = \{X, X_2, X_4, X_6, Y\}$ and the associated unobserved variables $\{U_1, U_2\}$. If the original graph was denoted by G , we denote this graph by $G_{\mathbf{W}}$.

We denote by $Anc(X_i)_G$ with $X_i \in \mathbf{V}$ the observable ancestors of X_i in the graph G .

Definition 5.8. In a SMCM with graph G , observable variables \mathbf{V} and unobservable variables \mathbf{U} . A set of observable variables $\mathbf{W} \subset \mathbf{V}$ is said to be **ancestral** if the observable ancestors of \mathbf{W} is equal to \mathbf{W} itself, or formally: $Anc(\mathbf{W})_G = \mathbf{W}$.

We have seen before that the c-factor $Q[\mathbf{S}_l]$ of a c-component \mathbf{S}_l is the post-intervention distribution of the variables in \mathbf{S}_l when intervening upon all the other observable variables $\mathbf{V} \setminus \mathbf{S}_l$ (Equation (5.6)).

An important property of c-factors of c-components is that they are all identifiable, as stated in the following property.

Property 5.1. The c-factor $Q[\mathbf{S}]$ of every c-component \mathbf{S} in a SMCM with observable variables \mathbf{V} is identifiable from the JPD $P(\mathbf{V})$.

The next property provides a condition under which $Q[\mathbf{A}]$ can be computed from $Q[\mathbf{B}]$, where \mathbf{A} is a subset of \mathbf{B} , namely when \mathbf{A} is ancestral in \mathbf{B} .

Property 5.2. In a SMCM with graph G over observable variables \mathbf{V} , where $\mathbf{A} \subset \mathbf{B} \subset \mathbf{V}$, if \mathbf{A} is an ancestral set in the graph $G_{\mathbf{B}}$ ($\mathbf{A} = \text{Anc}(\mathbf{A})_{G_{\mathbf{B}}}$), then $Q[\mathbf{A}]$ is identifiable from $Q[\mathbf{B}]$.

Finally, the following property is a generalization of Property 5.1 to proper subgraphs of G .

Property 5.3. In a SMCM with graph G over observable variables \mathbf{V} , where $\mathbf{H} \subset \mathbf{V}$ and \mathbf{H} is partitioned in c -components $\mathbf{H}_1, \dots, \mathbf{H}_l$ in $G_{\mathbf{H}}$, then every $Q[\mathbf{H}_i]$ is identifiable from $Q[\mathbf{H}]$.

In the next section we will use these three properties to sketch the identification algorithm.

5.3.3 Sketch of Algorithm

We will treat the problem of calculating the effect of manipulating X_i on \mathbf{T} , $P(\mathbf{t}|\text{do}(x_i))$, where X_i is a single variable and \mathbf{T} is a set of variables, from a SMCM with observable variables \mathbf{V} and from the JPD over those variables, $P(\mathbf{V})$. Extending the algorithm to the case where more than one variable is being manipulated (i.e. where X_i is a set of variables) creates some extra problems which will not be treated here. See (Tian and Pearl, 2002b) for all the details.

Let the observable variables \mathbf{V} of an SMCM be partitioned into c -components $\mathbf{S}_X, \mathbf{S}_1, \dots, \mathbf{S}_k$, where $X \in \mathbf{S}_X$.

First of all, we will show why the set of variables D_x , defined as:

$$\mathbf{D}_X = \text{Anc}(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}} \cap \mathbf{S}_X \quad (5.9)$$

are the crucial variables to check whether $P(\mathbf{t}|\text{do}(x))$ is identifiable.

- $\text{Anc}(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}}$, or the ancestors of \mathbf{T} in the graph without variable X , because the ancestor of \mathbf{T} are the only variables that can causally influence \mathbf{T} . Only the ancestors in the graph **without** X , because due to the intervention, the old parents (i.e. the old immediate causes) of X no longer influence \mathbf{T} via X .
- \mathbf{S}_X , or the c -component of X , because it is the c -factor $Q[\mathbf{S}_X]$ that contains the CPD of variable X conditional on its causes before the intervention: $P(X|\mathbf{II}, \text{UPa}(X))$. It is this CPD that has to be removed from the JPD as stated by the *manipulation theorem*.

As an example, in Figure 5.4, we respectively see: (a) \mathbf{S}_X (b) $\text{Anc}(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}}$ (c) \mathbf{D}_X depicted as rectangles instead of circles, applied to the SMCM of Figure 5.2.

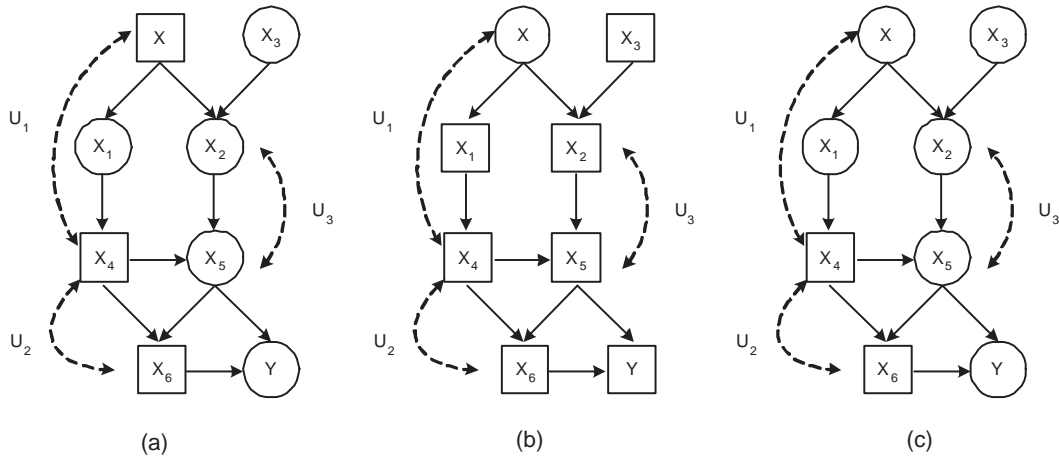


Figure 5.4. In the SMCM of Figure 5.2 the subsets of variables (a) \mathbf{S}_X (b) $\text{Anc}(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}}$ (c) $\mathbf{D}_X = \text{Anc}(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}} \cap \mathbf{S}_X$ are respectively depicted by rectangles.

The original graph reduced to these variables \mathbf{D}_X and \mathbf{S}_X is sufficient to know whether a causal effect $P(\mathbf{t}|\text{do}(x))$ can be computed from the JPD over the observed variables \mathbf{V} . To actually compute $P(\mathbf{t}|\text{do}(x))$, all observable variables \mathbf{V} of the JPD are needed, but, as we will see in the next section, for the variables $\mathbf{V} \setminus \mathbf{D}_X$, the calculations are trivial. Therefore, in the rest of this subsection we will focus on the calculation of $Q[\mathbf{D}_X]$.

In Figure 5.5, a conceptual sketch of the identification algorithm is depicted. The rest of this subsection will be devoted to explain its different steps.

- As \mathbf{S}_X is a c-component in the SMCM, $Q[\mathbf{S}_X]$ can be calculated from the JPD $P(\mathbf{V})$ using Property 5.1.
- The important variables are $\mathbf{D}_X = \text{Anc}(Y)_{G_{\mathbf{V} \setminus \{X\}}} \cap \mathbf{S}_X$
- Next, we will try to calculate $Q[\mathbf{D}_X]$ from $Q[\mathbf{S}_X]$.

If \mathbf{D}_X is an ancestral set in $G_{\mathbf{S}_X}$, its c-factor $Q[\mathbf{D}_X]$ can be calculated from $Q[\mathbf{S}_X]$ by applying Property 5.2.

As \mathbf{D}_X is not always an ancestral set in $G_{\mathbf{S}_X}$ we have to find a more general way to calculate $Q[\mathbf{D}_X]$ from $Q[\mathbf{S}_X]$. Therefore, we will apply Property 5.3 to rewrite $Q[\mathbf{D}_X]$ as a product of the c-factors of its l c-components $\mathbf{D}_{X,1}, \dots, \mathbf{D}_{X,l}$ in the graph $G_{\mathbf{D}_X}$:

$$Q[\mathbf{D}_X] = \prod_{j=1}^l Q[\mathbf{D}_{X,j}]$$

- Now if we can calculate each $Q[\mathbf{D}_{X,j}]$ our problem is solved.
- We first introduce a notation for the ancestors of $\mathbf{D}_{X,j}$ in the graph $G_{\mathbf{S}_X}$: $\mathbf{A} = \text{Anc}(\mathbf{D}_{X,j})_{G_{\mathbf{S}_X}}$. Then, each of these $\mathbf{D}_{X,j}$ is either:

given: $P(v)$ $S_X = c\text{-component of } X$
 wanted: $P(t|do(x))$ $D_X = Anc(T)_{G_{V \setminus \{x\}}} \cap S_X$
 $D_{X,j}$ for $j = \{1, \dots, l\} = c\text{-components of } G_{D_X}$

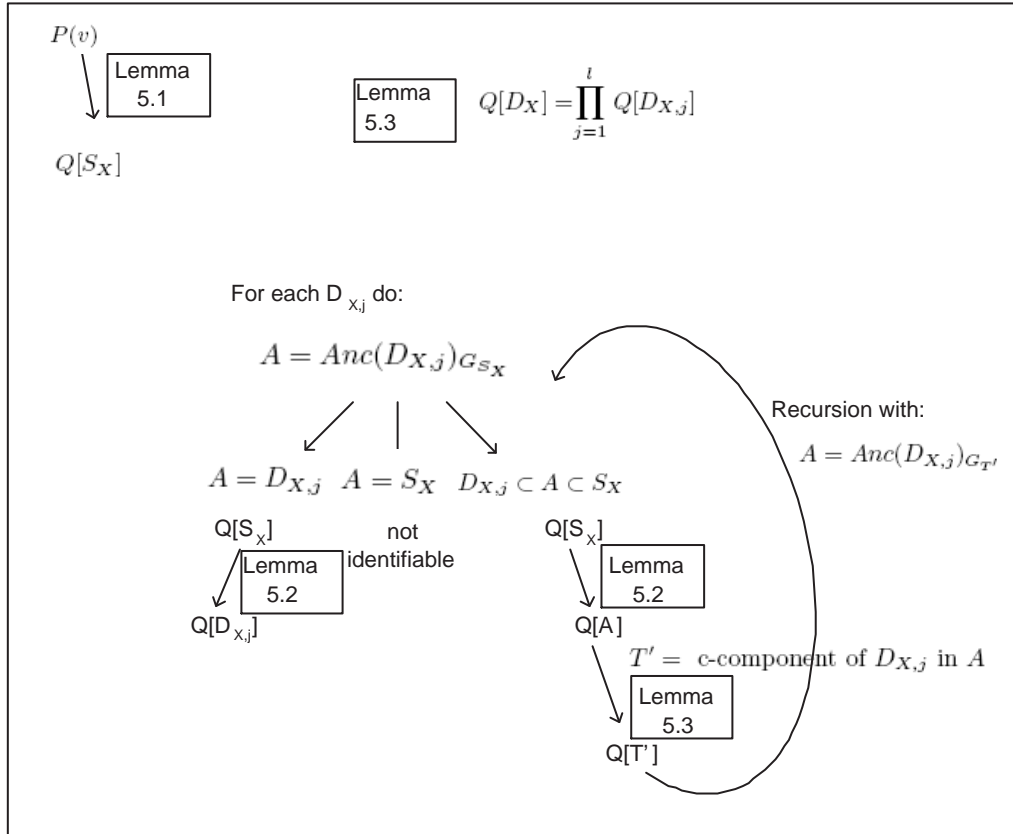


Figure 5.5. A conceptual sketch of the identification algorithm.

1. ancestral in G_{S_X} , or $\mathbf{A} = \mathbf{D}_{X,j}$.
2. its ancestral set is equal to \mathbf{S}_X itself, or $\mathbf{A} = \mathbf{S}_X$.
3. its ancestral set is strictly in between itself and \mathbf{S}_X , or

$$\mathbf{D}_{X,j} \subset \mathbf{A} \subset \mathbf{S}_X.$$

In the first case we can use Property 5.2 to obtain $Q[\mathbf{D}_{X,j}]$.

In the 2nd case, $Q[\mathbf{D}_{X,j}]$ is not identifiable in this SMCM and consequently, $P(\mathbf{t}|do(x))$ is not identifiable.

In the final case, we will initiate a recursive call of the above process. We start by calculating $Q[\mathbf{A}]$ from $Q[\mathbf{S}_X]$. This can be done by applying Property 5.2, as $Q[\mathbf{A}]$ is ancestral in G_{S_X} by definition.

- When all recursive calls are successful we can calculate $P(\mathbf{t}|do(x))$.
- When we perform the recursive call we know that $\mathbf{D}_{X,j}$ will always belong to a single c-component \mathbf{H}_X in $G_{\mathbf{A}}$, the graph reduced to its ancestral set \mathbf{A} . Furthermore $G_{\mathbf{A}}$ can be partitioned into c-components $\mathbf{H}_X, \mathbf{H}_1, \dots, \mathbf{H}_m$. Now by applying Property 5.3 we get:

$$Q[\mathbf{A}] = \mathbf{H}_X \prod_{i=1}^m \mathbf{H}_i$$

Property 5.3 also states that $Q[\mathbf{H}_X]$ can be obtained from $Q[\mathbf{S}_X]$. When $Q[\mathbf{H}_X]$ is obtained the process restarts, but now we want to calculate $Q[\mathbf{D}_{X,j}]$ from $Q[\mathbf{H}_X]$ instead of from $Q[\mathbf{S}_X]$.

These steps have to be repeated until we reach one of the first 2 cases for each $Q[\mathbf{D}_{X,j}]$. When all $Q[\mathbf{D}_{X,j}]$ are identifiable (i.e. each process ends in case 1), the desired quantity $P(\mathbf{t}|do(x))$ is identifiable. As soon as 1 of the $Q[\mathbf{D}_{X,j}]$ is not identifiable (i.e. the process ends in step 2), $P(\mathbf{t}|do(x))$ is not identifiable in the given SMCM.

Summary of the Algorithm

As a roundup we come back to the conceptual sketch of the identification algorithm of Figure 5.5. It consists of the following steps:

1. Calculate $Q[\mathbf{S}_X]$ via Property 5.1.
2. Calculate those $Q[\mathbf{D}_{X,j}]$ where $\mathbf{D}_{X,j}$ is an ancestral set in $G_{\mathbf{S}_X}$ via Property 5.2.
3. When $\mathbf{A} = \mathbf{S}_X$, then $Q[\mathbf{D}_{X,j}]$ cannot be calculated.
4. When $\mathbf{D}_{X,j} \subset \mathbf{A} \subset \mathbf{S}_X$, a recursive call tries to calculate $Q[\mathbf{D}_{X,j}]$ from $Q[\mathbf{H}_X]$ in the same way, where \mathbf{H}_X is the c-component of $\mathbf{D}_{X,j}$ in $G_{\mathbf{A}}$.

5.3.4 Technical introduction

In this section we will introduce the actual lemmas proposed as properties in the Section 5.3.2 and together with the properties of c-factors, they will be used to effectively develop an equation for calculating the effect of performing an intervention on a variable X on some others.

Lemma 5.1. *The c-factor $Q[\mathbf{S}]$ of every c-component \mathbf{S} in a SMCM with observable variables \mathbf{V} is identifiable from the JPD $P(\mathbf{v})$.*

Let $X_1 < \dots < X_n$ be a topological order over \mathbf{V} , and let $X^{(i)} = \{X_1, \dots, X_i\}$, $i = 1, \dots, n$ and $X^{(0)} = \emptyset$.

$$Q[\mathbf{S}] = \prod_{X_i \in \mathbf{S}} P(x_i | (\mathbf{T}_i \cup Pa(\mathbf{T}_i)) \setminus \{X_i\}) \quad (5.10)$$

where \mathbf{T}_i is the c -component of $G_{X^{(i)}}$ that contains X_i .

This lemma implies that every c -factor $Q[\mathbf{S}]$, and thus the post-intervention distribution of S after manipulating $\mathbf{V} \setminus \mathbf{S}$, can always be calculated from the JPD $P(\mathbf{v})$.

Consider for example the SMCM of Figure 5.2. As seen before it consists of 5 c -components. Then, if we choose the topological order

$$X < X_3 < X_1 < X_2 < X_4 < X_5 < X_6 < Y$$

we get

$$\begin{aligned} Q[X, X_4, X_6] &= P(x)P(x_4|x, x_1)P(x_6|x, x_1, x_4, x_5) \\ Q[X_1] &= P(x_1|x) \\ Q[X_2, X_5] &= P(x_2|x, x_3)P(x_5|x, x_3, x_2, x_4) \\ Q[X_3] &= P(x_3) \\ Q[Y] &= P(y|x_5, x_6) \end{aligned}$$

The next lemma provides a condition under which summing a c -factor $Q[\mathbf{B}]$ over some variables is equivalent to removing the corresponding factors. It also provides a condition under which we can compute $Q[\mathbf{A}]$ from $Q[\mathbf{B}]$, where \mathbf{A} is a subset of \mathbf{B} , by simply summing $Q[\mathbf{B}]$ over the remaining variables $\mathbf{B} \setminus \mathbf{A}$.

Lemma 5.2. *In a SMCM with graph G over observable variables \mathbf{V} , where $\mathbf{A} \subset \mathbf{B} \subset \mathbf{V}$, if \mathbf{A} is an ancestral set in the graph $G_{\mathbf{B}}$ ($\mathbf{A} = Anc(\mathbf{A})_{G_{\mathbf{B}}}$), then $Q[\mathbf{A}]$ is identifiable from $Q[\mathbf{B}]$ as follows*

$$Q[\mathbf{A}] = \sum_{\{v_i | V_i \in \mathbf{B} \setminus \mathbf{A}\}} Q[\mathbf{B}] \quad (5.11)$$

The lemma implies that when we have the c -factor $Q[\mathbf{B}]$ over some set \mathbf{B} (e.g. via Lemma 5.1), and a subset \mathbf{A} of \mathbf{B} is ancestral in the graph reduced to variables \mathbf{B} , then $Q[\mathbf{A}]$ can be calculated from $Q[\mathbf{B}]$, and this simply by marginalizing over $Q[\mathbf{B}]$.

This is the case, because when \mathbf{A} is an ancestral set in $G_{\mathbf{B}}$, this also means that in the set $\mathbf{B} \setminus \mathbf{A}$ there are no parents of \mathbf{A} (as all ancestors of \mathbf{A} are in \mathbf{A} itself). This implies that the remaining variables $\mathbf{B} \setminus \mathbf{A}$ do not causally influence variables \mathbf{A} , as they are necessarily non-ancestors of \mathbf{A} .

Furthermore, the influence of variables $\mathbf{B} \setminus \mathbf{A}$ can be removed from the equation of $Q[\mathbf{B}]$ by marginalization, on one hand because in the contributions of variables \mathbf{A} to the mixture

of products that constitutes $Q[\mathbf{B}]$, there figure no variables from $\mathbf{B} \setminus \mathbf{A}$, precisely because the variables in this set are non-ancestors of \mathbf{A} . Furthermore, the unconditional parts of contributions of variables $\mathbf{B} \setminus \mathbf{A}$ to $Q[\mathbf{B}]$ are exactly the variables $\mathbf{B} \setminus \mathbf{A}$, and thus by marginalizing over them, we remove exactly the contributions of $\mathbf{B} \setminus \mathbf{A}$ and keep the contributions of \mathbf{A} .

For example in Figure 5.6 we see the SMCM of Figure 5.2 reduced to \mathbf{S}_X , the c-component of X . We can see that the set (X_4, X_6) is ancestral in this graph, as $Anc(X_4, X_6)_{G_{S_X}} = (X_4, X_6)$, and thus Lemma 5.2 implies that $Q[X_4, X_6]$ can be calculated from $Q[X, X_4, X_6]$ via:

$$Q[X_4, X_6] = \sum_{\{x|X\}} Q[X, X_4, X_6]$$

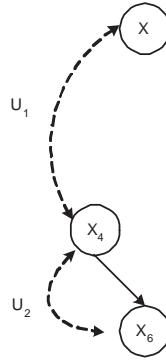


Figure 5.6. The SMCM of Figure 5.2 reduced to S_X , the c-component of X . Lemma 5.2 implies that $Q[X_4, X_6]$ can be calculated from $Q[X, X_4, X_6]$.

When we apply this to the c-factor of $Q[X, X_4, X_6]$, as calculated previously (Equation (5.8)), we get:

$$\begin{aligned} \sum_{\{x|X\}} Q[X, X_4, X_6] &= \sum_{\{x|X\}} \sum_{\{u_k|U_k \in \{U_1, U_2\}\}} P(x|u_1)P(x_4|x_1, u_1, u_2)P(x_6|x_4, x_5, u_2) \cdot \\ &\quad P(u_1)P(u_2) \\ &= \sum_{\{u_k|U_k \in \{U_1, U_2\}\}} P(x_4|x_1, u_1, u_2)P(x_6|x_4, x_5, u_2)P(u_1)P(u_2) \\ &= Q[X_4, X_6] \end{aligned}$$

We see that the factor $P(x|u_1)$ can be marginalized away, because that is the only occurrence of X , and then we obtain exactly $Q[X_4, X_6]$.

On the other hand, when we try to calculate $Q[X_6]$ from $Q[X, X_4, X_6]$ in such a way, we get:

$$\begin{aligned} Q[X_6] &= \sum_{\{x_i | X_i \in \{X, X_4\}\}} Q[X, X_4, X_6] \\ &= \sum_{\{x_i | X_i \in \{X, X_4\}\} \{u_i | U_i \in \{U_1, U_2\}\}} \sum P(x|u_1)P(x_4|x_1, u_1, u_2)P(x_6|x_4, x_5, u_2). \\ &\quad P(u_1)P(u_2) \end{aligned}$$

As X_6 is no ancestral set in G_{S_X} , X_4 occurs in the conditional part of $P(x_6|x_4, x_5, u_2)$, and thus the factor $P(x_4|x_1, u_1, u_2)$ cannot be marginalized away.

Finally, the following lemma is a generalization of the factorization of Equation 5.5 and Lemma 5.1 to proper subgraphs of G .

Lemma 5.3. *In a SMCM with graph G over observable variables V , where $H \subset V$ and H is partitioned in c -components H_1, \dots, H_l in G_H , then*

$$Q[H] = \prod_{i=1}^l Q[H_i] \quad (5.12)$$

Furthermore, every $Q[H_i]$ is identifiable from $Q[H]$ as follows. Let $\#H = k$ and let X_{h_1}, \dots, X_{h_k} be a topological order of the variables in G_H . Let $H^{(i)} = \{X_{h_1}, \dots, X_{h_i}\}$ be the set of variables in H ordered before X_{h_i} including X_{h_i} , $i = 1, \dots, k$, and $H^{(0)} = \emptyset$. Each $Q[H_j]$, $j = 1, \dots, l$, is given by

$$Q[H_j] = \prod_{\{i | X_{h_i} \in H_j\}} \frac{Q[H^{(i)}]}{Q[H^{(i-1)}]}, \quad (5.13)$$

where each $Q[H^{(i)}]$, $i = 0, 1, \dots, k$, is given by

$$Q[H^{(i)}] = \sum_{\{x_{h_i} | X_{h_i} \in (H \setminus H^{(i)})\}} Q[H] \quad (5.14)$$

This lemma resembles Lemma 5.1, but instead of treating c -components of entire graphs it can be used to calculate the c -components of a subgraph G_H . Remark that this lemma only states that the $Q[H_i]$ are computable from $Q[H]$, implying that $Q[H]$ has to be known for this lemma to be applicable.

For example consider the SMCM of Figure 5.2 reduced to the variables $H = \{X, X_1, X_4, X_6\}$. G_H can be seen in Figure 5.7.

By Lemma 5.3

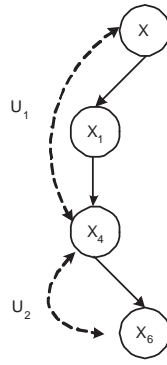


Figure 5.7. The SCMC of Figure 5.2 reduced to $\mathbf{H} = \{X, X_1, X_4, X_6\}$.

$$Q[\mathbf{H}] = Q[X, X_4, X_6]Q[X_1] \quad (5.15)$$

The only topological order over $G_{\mathbf{H}}$ is $X < X_1 < X_4 < X_6$. Then

$$Q[X, X_4, X_6] = Q[X] \frac{Q[X, X_1, X_4]}{Q[X, X_1]} \frac{Q[X, X_1, X_4, X_6]}{Q[X, X_1, X_4]} \quad (5.16)$$

and

$$Q[X_1] = \frac{Q[X, X_1]}{Q[X]}. \quad (5.17)$$

As $Q[\mathbf{H}]$ is given (assumption of Lemma 5.3), the remaining c-factors in the righthandside of the equation can be obtained as follows

$$Q[X] = \sum_{\{x_i | X_i \in (\mathbf{H} \setminus X)\}} Q[\mathbf{H}] \quad (5.18)$$

$$= \sum_{\{x_i | X_i \in \{X_1, X_4, X_6\}\}} Q[\mathbf{H}], \quad (5.19)$$

and

$$Q[X, X_1] = \sum_{\{x_i | X_i \in (\mathbf{H} \setminus \{X, X_1\})\}} Q[\mathbf{H}] \quad (5.20)$$

$$= \sum_{\{x_i | X_i \in \{X_4, X_6\}\}} Q[\mathbf{H}]. \quad (5.21)$$

And likewise for the other c-factors.

Identification Formula

In this subsection, we use the information discussed above to develop an equation for calculating the effect of intervening on a variable X on some other variables.

Let the observable variables \mathbf{V} of an SMCM be partitioned into c -components $\mathbf{S}_X, \mathbf{S}_1, \dots, \mathbf{S}_k$, where $X \in \mathbf{S}_X$, and let $\mathbf{V}' = \mathbf{V} \setminus \{X\}$. We are interested in calculating $P(\mathbf{t}|do(x))$, where X is a single variable and \mathbf{T} is a set of variables.

As seen before (Equation (5.5)), the c -factors factorize the JPD $P(\mathbf{v})$:

$$P(\mathbf{v}) = Q[\mathbf{S}_X] \prod_{i=1}^k Q[\mathbf{S}_i] \quad (5.22)$$

$$= Q[\mathbf{V}] \quad (5.23)$$

By applying the *manipulation theorem* and the definition of c -factors we get

$$P(\mathbf{v}'|do(x)) = Q[\mathbf{S}_X \setminus \{X\}] \prod_{i=1}^k Q[\mathbf{S}_i] \quad (5.24)$$

$$= Q[\mathbf{V}'] \quad (5.25)$$

To obtain $P(\mathbf{t}|do(x))$ with $\mathbf{T} \subseteq \mathbf{V}'$ we marginalize over $\mathbf{V}' \setminus \mathbf{T}$:

$$P(\mathbf{t}|do(x)) = \sum_{\{x_i | X_i \in (\mathbf{V}' \setminus \mathbf{T})\}} P(\mathbf{v}'|do(x)) \quad (5.26)$$

$$= \sum_{\{x_i | X_i \in (\mathbf{V}' \setminus \mathbf{T})\}} Q[\mathbf{V}'] \quad (5.27)$$

Now if we define $\mathbf{D} = Anc(\mathbf{T})_{G_{V'}}$ (the ancestors of \mathbf{T} in the graph without variable X), we can rewrite Equation (5.27) as follows:

$$P(\mathbf{t}|do(x)) = \sum_{\{x_i | X_i \in (\mathbf{D} \setminus \mathbf{T})\}} \sum_{\{x_j | X_j \in (\mathbf{V}' \setminus \mathbf{D})\}} Q[\mathbf{V}'] \quad (5.28)$$

$$= \sum_{\{x_i | X_i \in (\mathbf{D} \setminus \mathbf{T})\}} Q[\mathbf{D}] \quad (5.29)$$

The step from equation 5.28 to 5.29 is made by using Lemma 5.2, as \mathbf{D} is by definition an ancestral set in $G_{V'}$. Now, if we define $\mathbf{D}_X = \mathbf{D} \cap \mathbf{S}_X$, and $\mathbf{D}_i = \mathbf{D} \cap \mathbf{S}_i$ for $i = 1, \dots, k$, then from Equation (5.24) we can write $Q[\mathbf{D}]$ as

$$Q[\mathbf{D}] = Q[\mathbf{D}_X] \prod_{i=1}^k Q[\mathbf{D}_i] \quad (5.30)$$

All \mathbf{D}_i are ancestral sets in $G_{\mathbf{S}_i}$ by definition, and thus Lemma 5.2 can be applied:

$$Q[\mathbf{D}_i] = \sum_{\{x_l | X_l \in (\mathbf{S}_i \setminus \mathbf{D}_i)\}} Q[\mathbf{S}_i], \text{ for } i = 1, \dots, k. \quad (5.31)$$

Although $\mathbf{D}_X = \text{Anc}(\mathbf{T})_{G_{\mathbf{V}}} \cap \mathbf{S}_X$ is ancestral in $\mathbf{S}_X \setminus \{X\}$, it may not be ancestral in \mathbf{S}_X , since X could be an ancestor of \mathbf{D}_X . Now, by combining Equations (5.29), (5.30) and (5.31), we get

$$P(\mathbf{t} | do(x)) = \sum_{\{x_n | X_n \in (\mathbf{D} \setminus \mathbf{T})\}} Q[\mathbf{D}_X] \prod_{i=1}^k \sum_{\{x_l | X_l \in (\mathbf{S}_i \setminus \mathbf{D}_i)\}} Q[\mathbf{S}_i] \quad (5.32)$$

We know from Lemma 5.1 that the c-factors $Q[\mathbf{S}_i]$ of all c-components \mathbf{S}_i are identifiable, so at this point the remaining question is how to calculate $Q[\mathbf{D}_X]$.

Now we can say that the graph $G_{\mathbf{D}_X}$ is partitioned into m c-components $\mathbf{D}_{X,1}, \dots, \mathbf{D}_{X,m}$. Then, Lemma 5.3 implies that $Q[\mathbf{D}_X] = \prod_{j=1}^m Q[\mathbf{D}_{X,j}]$ and it follows that:

$$P(\mathbf{t} | do(x)) = \sum_{\{x_n | X_n \in (\mathbf{D} \setminus \mathbf{T})\}} \prod_{j=1}^m Q[\mathbf{D}_{X,j}] \prod_{i=1}^k \sum_{\{x_l | X_l \in (\mathbf{S}_i \setminus \mathbf{D}_i)\}} Q[\mathbf{S}_i] \quad (5.33)$$

After obtaining this equation we know that $P(\mathbf{t} | do(x))$ is identifiable if all $Q[\mathbf{D}_{X,j}]$ are identifiable. Now, the next question is whether these $Q[\mathbf{D}_{X,j}]$ are identifiable from $Q[\mathbf{S}_X]$.

If we denote the ancestors of a $\mathbf{D}_{X,j}$ in the graph $G_{\mathbf{S}_X}$ by $\mathbf{A} = \text{Anc}(\mathbf{D}_{X,j})_{G_{\mathbf{S}_X}}$, there are 3 possibilities:

1. $\mathbf{A} = \mathbf{D}_{X,j}$, meaning that $\mathbf{D}_{X,j}$ is an ancestral set in $G_{\mathbf{S}_X}$, and thus by Lemma 5.2, $Q[\mathbf{D}_{X,j}]$ is identifiable from $Q[\mathbf{S}_X]$ as follows:

$$Q[\mathbf{D}_{X,j}] = \sum_{\{x_i | X_i \in (\mathbf{S}_X \setminus \mathbf{D}_{X,j})\}} \mathbf{S}_X \quad (5.34)$$

2. $\mathbf{A} = \mathbf{S}_X$, meaning that the ancestral set of $\mathbf{D}_{X,j}$ in \mathbf{S}_X is \mathbf{S}_X itself. In this case there is no way to calculate $Q[\mathbf{D}_{X,j}]$, consequently the desired $P(\mathbf{t} | do(x))$ is not identifiable from the SMCM.
3. $\mathbf{D}_{X,j} \subset \mathbf{A} \subset \mathbf{S}_X$, in this case $Q[\mathbf{A}]$ is identifiable from $Q[\mathbf{S}_X]$ via Lemma 5.2, as \mathbf{A} is ancestral in $G_{\mathbf{S}_X}$ by definition:

$$Q[\mathbf{A}] = \sum_{\{x_i | X_i \in (\mathbf{S}_X \setminus \mathbf{A})\}} Q[\mathbf{S}_X] \quad (5.35)$$

Because $\mathbf{D}_{X,j}$ is defined as a c-component in $G_{\mathbf{D}_X}$ and $\mathbf{D}_{X,j} \subset \mathbf{A}$, $\mathbf{D}_{X,j}$ is still part of a single c-component \mathbf{H}_X in $G_{\mathbf{A}}$.

Now, Lemma 5.3 states that $Q[\mathbf{H}_X]$ is identifiable from $Q[\mathbf{A}]$ and thus the problem is reduced to whether $Q[\mathbf{D}_{X,j}]$ is identifiable from $Q[\mathbf{H}_X]$. At this point, we again have the same 3 possibilities as in this enumeration, but \mathbf{S}_X is replaced by the smaller set \mathbf{H}_X . This process can be repeated until we reach one of the first 2 cases.

To conclude, $P(\mathbf{t}|do(x))$ is identifiable in a SMCM, when in Equation (5.33),

$$P(\mathbf{t}|do(x)) = \sum_{\{x_n | X_n \in (\mathbf{D} \setminus \mathbf{T})\}} \prod_{j=1}^m Q[\mathbf{D}_{X,j}] \prod_{i=1}^k \sum_{\{x_l | X_l \in (\mathbf{S}_i \setminus \mathbf{D}_i)\}} Q[\mathbf{S}_i]$$

all $Q[\mathbf{D}_{X,j}]$ are identifiable by repeatedly applying the procedure with the 3 possibilities as explained above.

5.3.5 Algorithms

Finally, in this section we provide the identification algorithm and an auxiliary function.

We start with the function *Identify*($\mathbf{C}, \mathbf{F}, Q[\mathbf{F}]$) where $\mathbf{C} \subset \mathbf{F}$ are sets of variables. The goal of this function is to calculate $Q[\mathbf{C}]$ from $Q[\mathbf{F}]$ if possible. *Identify* can be seen in Function 5.

Function 5 Single agent *Identify*($\mathbf{C}, \mathbf{F}, Q[\mathbf{F}]$)

Require: sets of variables $\mathbf{C} \subset \mathbf{F} \subset \mathbf{V}$, $Q = Q[\mathbf{F}]$.

Ensure: Expression for $Q[\mathbf{C}]$ in terms of Q or fail to determine.

Let $\mathbf{A} = Anc(\mathbf{C})_{G_{\mathbf{F}}}$ {= the ancestors of \mathbf{C} in graph \mathbf{G} restricted to variables \mathbf{F} }.

1. IF $\mathbf{A} = \mathbf{C}$, return $Q[\mathbf{C}] = \sum_{\{x_i | X_i \in (\mathbf{F} \setminus \mathbf{C})\}} Q$.
 2. IF $\mathbf{A} = \mathbf{F}$, return FAIL.
 3. IF $\mathbf{C} \subset \mathbf{A} \subset \mathbf{F}$
 - a) Assume that in $G_{\mathbf{A}}$, \mathbf{C} is contained in a c-component \mathbf{F}' .
 - b) Compute $Q[\mathbf{A}] = \sum_{\{x_i | X_i \in (\mathbf{F} \setminus \mathbf{A})\}} Q$ by Lemma 5.2.
 - c) Compute $Q[\mathbf{F}']$ from $Q[\mathbf{A}]$ using Lemma 5.3.
 - d) return *Identify*($\mathbf{C}, \mathbf{F}', Q[\mathbf{F}']$).
-

The actual identification algorithm can be seen in Algorithm 6. Given a SMCM and a query $P(\mathbf{t}|do(x))$, this algorithm will calculate that causal effect if possible. In the first steps the graph is divided into c-components, their respective c-factors are calculated and the variable set \mathbf{D}_X is determined. Then, the necessary calls of the *Identify* function are performed, and finally, all calculations are combined in the final equation. Both the function and the algorithm are adapted from (Tian and Pearl, 2002b).

Algorithm 6 Single agent identification of $P(\mathbf{t}|do(x))$

Require: a SMCM with variables \mathbf{V} , graph G , JPD $P(\mathbf{v})$, a variable $X \in \mathbf{V}$, and a set $\mathbf{T} \subset \mathbf{V}$.

Ensure: the expression for $P(\mathbf{t}|do(x))$ or fail to determine.

1. Find the c-component of G : $\mathbf{S}_X, \mathbf{S}_1, \dots, \mathbf{S}_k$, where $X \in \mathbf{S}_X$.
2. Compute the c-factors $Q[\mathbf{S}_X], Q[\mathbf{S}_1], \dots, Q[\mathbf{S}_k]$ by Lemma 5.1.
3. Let $\mathbf{D} = Anc(\mathbf{T})_{G_{\mathbf{V} \setminus \{X\}}}$, and $\mathbf{D}_X = \mathbf{D} \cap \mathbf{S}_X$
4. Let the c-components of $G_{\mathbf{D}_X}$ be $D_{X,j}, j = 1, \dots, m$.

For each set $\mathbf{D}_{X,j}$: Compute $Q[\mathbf{D}_{X,j}]$ from $Q[\mathbf{S}_X]$ by calling the function *Identify*($\mathbf{D}_{X,j}, \mathbf{S}_X, Q[\mathbf{S}_X]$) given in Function 5. If the function returns FAIL, then stop and output FAIL. Otherwise output

$$P(\mathbf{t}|do(x)) = \sum_{\{x_n | X_n \in (\mathbf{D} \setminus \mathbf{T})\}} \prod_{j=1}^m Q[\mathbf{D}_{X,j}] \prod_{i=1}^k \sum_{\{x_i | X_i \in (\mathbf{S}_i \setminus \mathbf{D}_i)\}} Q[\mathbf{S}_i]$$

5.4 Learning Causal Latent Networks

In this section we discuss learning the parameters of causal latent networks.

5.4.1 Learning the Parameters

Prior to this dissertation not too much focus has been put on learning the parameters of AGs or SMCMs. We briefly discuss different options for both techniques.

Ancestral Graphs

It is possible to parametrize AGs but this is a very difficult task and not every distribution can be used because of the bi-directed links that can be present in AGs. Not much research has been done on this subject, but due to a transformation we propose in Chapter 6 we can change an AG into a SMCM with the same properties and hence parametrize the AG this way.

Semi-Markovian Causal Models

For SMCMs it is always assumed that the entire JPD is available and all calculations of all queries are done using the JPD. In Chapter 8 we will introduce a more concise representation of the JPD as a factorization of smaller CPDs and show that these CPD can be learned from data.

5.4.2 Learning the Structure

In this section we discuss learning the structure of causal latent networks.

Ancestral Graphs

There is a lot of recent research on learning the structure of MAGs from observational data. The Fast Causal Inference (FCI) algorithm is a constraint based learning algorithm proposed by (Spirtes et al., 1999). Together with the rules discussed in (Zhang and Spirtes, 2005a), the result of the FCI algorithm is a representation of the Markov equivalence class of MAGs.

This representative is referred to as a *complete partial ancestral graph* (CPAG) and in (Zhang and Spirtes, 2005a) it is defined as follows:

Definition 5.9. Let $[G]$ be the Markov equivalence class for an arbitrary MAG G . The *complete partial ancestral graph* (CPAG) for $[G]$, P_G , is a graph with possibly the following edges $\rightarrow, \leftrightarrow, o-o, o\rightarrow$, such that

1. P_G has the same adjacencies as G (and hence any member of $[G]$) does;
2. A mark of arrowhead ($>$) is in P_G if and only if it is invariant in $[G]$; and
3. A mark of tail ($-$) is in P_G if and only if it is invariant in $[G]$.
4. A mark of (o) is in P_G if not all members in $[G]$ have the same mark.

We show the FCI algorithm in Algorithm 7, but first we must define some notions. For a given CPAG G we define the following notions: *SepSet*, $D - SEP$, *Possible - D - SEP*(X_i, X_j) and $X_i * \underline{- * X_m * -} * X_j$.

Definition 5.10. A set of nodes Z is in *SepSet*(X_i, X_j) if $(X_i \perp\!\!\!\perp X_j | Z)$.

Definition 5.11. A variable X_k is in $D - SEP(X_i, X_j)$ if and only if $X_k \neq X_i$ and there is an undirected path between X_i and X_k on which every vertex except the endpoints is a collider and each vertex is an ancestor of X_i or X_j .

If $X_i \neq X_j$, X_m is in *Possible-D-SEP*(X_i, X_j) in G if and only if:

- $X_m \neq X_i$
- There is an undirected path between X_i and X_m such that for every three nodes $X_{p_1}, X_{p_2}, X_{p_3}$ on a subpath either X_{p_2} is a collider on the subpath, or X_{p_2} is not a definite noncollider and on the path $X_{p_1}, X_{p_2}, X_{p_3}$ form a triangle.

Definition 5.12. A triple $X_i - \underline{X_m} - X_j$ is a notation indicating that there can be no collider at X_m .

Recent work in the area consists of characterising the equivalence class of CPAGs and finding single-edge operators to create equivalent MAGs (Ali et al., 2005; Zhang and Spirtes, 2005a,b). One of the goals of these advances is to create methods that search in the space of Markov equivalent models (CPAGs) instead of the space of all models (MAGs), mimicking results in the case without latent variables (Chickering, 2002a).

Algorithm 7 FCI algorithm (Spirtes et al., 2000a).**Require:** A set of samples from a distribution $P(\mathbf{V})$ faithful to some underlying DAG.**Ensure:** CPAG representation of the underlying DAG.

1. Initialization:
 - G is complete undirected graph on the vertex set \mathbf{V} .
 2. Skeleton discovery:
 - $n=0$;
 - repeat
 - repeat
 - select an ordered pair of variables X_i and X_j that are adjacent in G such that $Ne(X_i) \setminus X_j$ has cardinality greater than or equal to n , and a subset \mathbf{S} of $Ne(X_i) \setminus X_j$ of cardinality n , and if X_i and X_j are independent given \mathbf{S} delete edge $X_i - X_j$ from G and record \mathbf{S} in $Sepset(X_i, X_j)$ and $Sepset(X_j, X_i)$;
 - until all ordered pairs of adjacent variables X_i and X_j such that $Ne(X_i) \setminus X_j$ has cardinality greater than or equal to n and all subsets \mathbf{S} of $Ne(X_i) \setminus X_j$ of cardinality n have been tested for independence;
 - $n=n+1$;
 - until for each ordered pair of adjacent vertices X_i, X_j , $Ne(X_i) \setminus X_j$ is of cardinality less than n .
 3. Let F be the undirected graph resulting from the previous step. Orient each edge as $o-o$. For each triple of vertices X_i, X_j, X_k such that the pair X_i, X_j and the pair X_j, X_k are each adjacent in F but the pair X_i, X_k are no adjacent in F , orient $X_i * - * X_j * - * X_k$ as $X_i * \rightarrow X_j \leftarrow * X_k$ if and only if X_j is not in $Sepset(X_i, X_k)$.
 4. For each pair of variables X_i and X_j adjacent in F , if X_i and X_j are independent given any subset \mathbf{S} of $Possible - D - SEP(X_i, X_j) \setminus \{X_i, X_j\}$ or any subset S of $Possible - D - SEP(X_j, X_i) \setminus \{X_j, X_i\}$ in F remove the edge between X_i and X_j , and record S in $Sepset(X_i, X_j)$ and in $Sepset(X_j, X_i)$.
 5. The algorithm then reorients an edge between any pair of variables X_i and X_j as $X_i o-o X_j$.
 6. Let F' be the resulting graph of the previous step. For each triple of vertices X_i, X_j, X_k such that the pair X_i, X_j and the pair X_j, X_k are each adjacent in F but the pair X_i, X_k are no adjacent in F , orient $X_i * - * X_j * - * X_k$ as $X_i * \rightarrow X_j \leftarrow * X_k$ if and only if X_j is not in $Sepset(X_i, X_k)$, and orient $X_i * - * X_j * - * X_k$ as $X_i * - * \underline{X_j} * - * X_k$ if and only if X_j is in $Sepset(X_i, X_k)$.
 7. repeat
 - If there is a directed path from X_i to X_j , and an edge $X_i * - * X_j$ orient $X_i * - * X_j$ as $X_i * \rightarrow X_j$, else if X_j is a collider along the path X_i, X_j, X_k, X_j is adjacent to X_l , and X_l is in $Sepset(X_i, X_k)$, the orient $X_j * - * X_l$ as $X_j \leftarrow X_l$.
 - else if U is a definite discriminating path between X_i and X_j for X_k , and X_l and X_m are adjacent to X_k on U , and $X_l - X_k - X_m$ is a triangle, then
 - if X_k is in $Sepset(X_i, X_j)$ then X_k is marked as a noncollider on subpath $X_l * - * \underline{X_k} * - * X_m$
 - else $X_l * - * X_k * - * X_m$ is oriented $X_l * \rightarrow X_k \leftarrow * X_m$.
 - else if $X_l * \rightarrow \underline{X_k} * - * X_m$ then orient as $X_l * \rightarrow X_k \rightarrow X_m$.
- until no more edges can be oriented.

As mentioned before for MAGs, in a CPAG the directed edges have to be interpreted as representing ancestral relations instead of immediate causal relations. More precisely, this means that there is a directed edge from X_i to X_j if X_i is an ancestor of X_j in the underlying DAG and there is no subset of observable variables \mathbf{Z} such that $(X_i \perp\!\!\!\perp X_j | \mathbf{Z})$. This does not necessarily mean that X_i has an immediate causal influence on X_j , it may also be a result of an inducing path between X_i and X_j . For instance in Figure 5.1(c), the link between X_1 and X_6 is present due to the inducing path X_1, X_2, L_1, X_6 shown in Figure 5.1(a).

Inducing paths may also introduce \leftrightarrow , \rightarrow , $o \rightarrow$ or $o-o$ between two variables, although there is no immediate influence in the form of an immediate causal influence or latent common cause between the two variables. An example of such a link is $X_3 o-o X_4$ in Figure 6.1(b).

Semi-Markovian Causal Models

In the literature no algorithm for learning the structure of an SMCM exists, in the next chapter we introduce techniques to perform that task, given some simplifying assumptions, and with the help of experiments.

5.5 Overview of the Chapter

In this chapter we introduced two commonly used paradigms for causal modeling with latent variables, namely ancestral graphs and semi-Markovian causal models. We described both their properties and their limitations. Both existing approaches only focus on one or a few of all the steps involved in a generic causal knowledge discovery approach.

We showed that AGs are specifically suited for structure learning in the presence of latent variables from observational data. However, the techniques only learn up to Markov equivalence and provide no clues on which additional experiments to perform in order to obtain the fully oriented causal graph. Furthermore, as of yet no parametrisation for discrete variables is provided for MAGs and no techniques for probabilistic inference have been developed. There is some work on algorithms for causal inference, but it is restricted to causal inference quantities that are the same for an entire Markov equivalence class of MAGs (Spirtes et al., 2000a), (Zhang, 2006).

We showed that SMCMs are specifically suited for performing quantitative causal inference in the presence of latent variables. However, at this time no efficient parametrisation of such models is provided and there are no techniques for performing efficient probabilistic inference. Furthermore there are no techniques to learn these models from data issued from observations, experiments or both.

The goal of the following chapter is to investigate the integral process from learning these models from observational and experimental data unto performing probabilistic or causal inference with them.

Contributions on Latent Variable Causal Modeling

In this chapter we introduce our contributions for causal latent networks. We propose an algorithm to learn the structure of a causal latent network and propose an alternative representation of an existing technique for modeling causal latent networks in order to allow probabilistic and causal inference with this technique.

The motivation for these contributions stem from the gap that existed in current research between the models being learned and the models being used. Once the causal latent network is given, there are extensive applications and many papers are written on performing causal inference in these models. However, when we made an overview of the available algorithms for learning these models we observed that the learned result was never as expressive as the models used to demonstrate the inference power. Furthermore, no discussion had been made on performing probabilistic inference in these models as such.

In the next sections we introduce an algorithm MyCaDo++ (MY CAusal DiscOvery ++) for learning the complete structure of a causal latent network. It is assumed, as for MyCaDo in Chapter 4, that there is perfect observational data available to learn the correct model up to Markov equivalence. MyCaDo++ uses a set of experiments to completely identify the causal latent model. Furthermore, we introduce the PR-representation of our causal latent model which allows to perform probabilistic inference.

6.1 Learning Causal Latent Networks

In this section we discuss learning causal latent networks. We do this by using existing techniques to learn the equivalence class of the AG that correctly represents the data and then transform this into the correct SMCM representing the data.

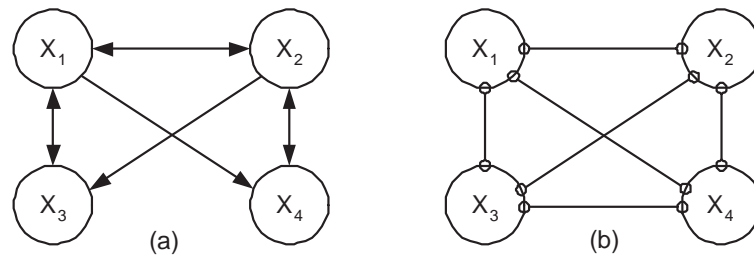


Figure 6.1. (a) A SMCM. (b) Result of FCI, with an i-false edge X_3o-oX_4 .

6.1.1 Preliminaries

Learning a SMCM, as is normal for probabilistic graphical models, consists of two parts: structure learning and parameter learning. Both can be done using data, expert knowledge and/or experiments. In this section we discuss only structure learning.

As mentioned above for MAGs, in a CPAG the directed edges have to be interpreted as being ancestral instead of causal. This means that there is a directed edge from X_i to X_j if X_i is an ancestor of X_j in the underlying DAG and there is no subset of observable variables D such that $(X_i \perp\!\!\!\perp X_j | D)$. This does not necessarily mean that X_i has an immediate causal influence on X_j : it may also be a result of an inducing path between X_i and X_j . For instance, in Figure 5.1(c), the link between X_1 and X_6 is present due to the inducing path X_1, X_2, L_1, X_6 shown in Figure 5.1(a).

Inducing paths may also introduce an edge of type $o \rightarrow$ or $o-o$ between two variables, indicating either a directed or bi-directed edge, although there is no immediate influence in the form of an immediate causal influence or latent common cause between the two variables. An example of such a link is X_3o-oX_4 in Figure 6.1(b).

A consequence of these properties of MAGs and CPAGs is that they are not suited for causal inference, since the immediate causal parents of each observable variable are not available, as is necessary according to the *manipulation theorem*. As we want to learn models that can perform causal inference, we will discuss how to transform a CPAG into a SMCM in the next sections. Before we start, we have to mention that we assume that the CPAG is correctly learned from data with the FCI algorithm, and the extended tail augmentation rules, i.e. we assume perfect observational data.

6.1.2 General Overview of the Algorithm

Our goal is to give an algorithm, MyCaDo++ (MyCaDo with latent variables), to transform a given CPAG in order to obtain a SMCM that corresponds to the corresponding DAG. Remember that in general there are three types of edges in a CPAG: \rightarrow , $o \rightarrow$, $o-o$, in which o means either a tail mark $-$ or a directed mark $>$. So one of the tasks to obtain a valid SMCM

is to disambiguate those edges with at least one o as an endpoint. A second task will be to identify and remove the edges that are created due to an inducing path.

A general overview of the algorithm is given in Figure 6.2.

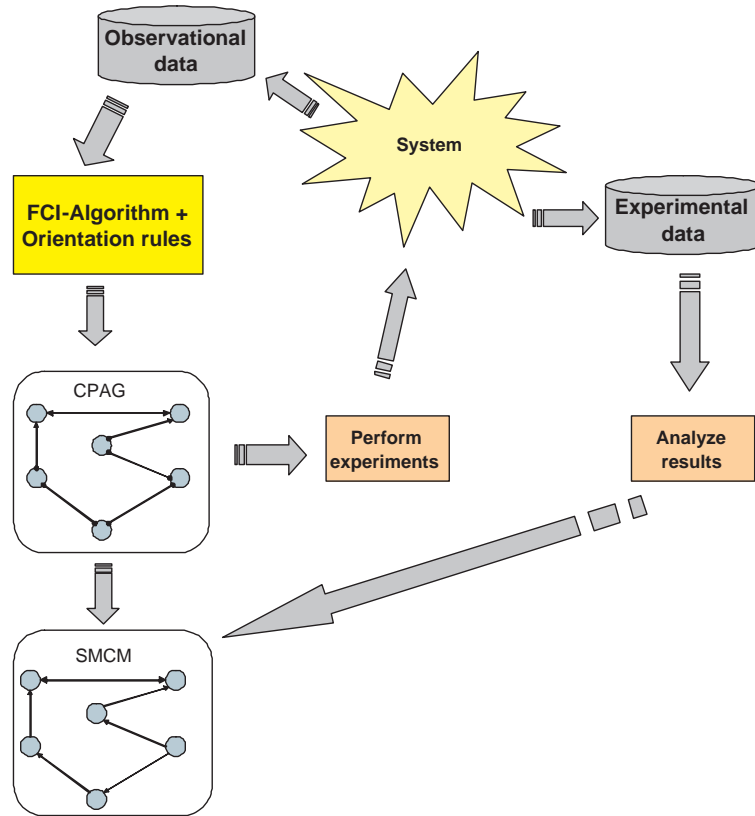


Figure 6.2. General overview of the MyCaDo++ algorithm.

In the next section we will first discuss exactly which information we obtain from performing an experiment. Then, we will discuss the two possibilities $o \rightarrow$ and $o-o$. Finally, we will discuss how we can find edges that are created due to inducing paths and how to remove these to obtain the correct SMCM.

6.1.3 Performing Experiments

The experiments discussed here play the role of the interventions that define a causal relation. An experiment on a variable X_i , i.e., a randomised controlled experiment, removes the influence of other variables in the system on X_i . The experiment forces a distribution on X_i , and thereby changes the joint distribution of all variables in the system that depend directly or indirectly on X_i but does not change the conditional distribution of other variables given values of X_i . After the randomisation, the associations of the remaining variables with X_i

provide information about which variables are influenced by X_i (Neapolitan, 2003). When we perform the experiment we cut all influence of other variables on X_i . Graphically this corresponds to removing all incoming edges into X_i from the underlying DAG.

All parameters besides those for the variable experimented on, (i.e., $P(X_i|I_i)$), remain the same. We then measure the influence of the intervention on variables of interest to get the post-interventional distribution on these variables.

To analyse the results of the experiment we compare for each variable of interest X_j the original distribution P and the post-interventional distribution P_E , thus comparing $P(X_j)$ and $P_E(X_j) = P(X_j|do(X_i = x_i))$.

We denote performing an experiment at variable X_i or a set of variables \mathbf{W} by $exp(X_i)$ or $exp(\mathbf{W})$ respectively, and if we have to condition on some other set of variables \mathbf{Z} while performing the experiment, we denote it as $exp(X_i)|\mathbf{Z}$ and $exp(\mathbf{W})|\mathbf{Z}$.

In general, if a variable X_i is experimented on and another variable X_j is affected by this experiment, we say that X_j varies with $exp(X_i)$, denoted by $exp(X_i) \rightsquigarrow X_j$. If there is no variation in X_j we note $exp(X_i) \not\rightsquigarrow X_j$.

Although conditioning on a set of variables \mathbf{Z} might cause some variables to become probabilistically dependent, conditioning will not influence whether two variables vary with each other when performing an experiment. I.e., suppose the following structure is given $X_i \rightarrow X_m \leftarrow X_j$, then conditioning on X_m will make X_i dependent on X_j , but when we perform an experiment on X_i and check whether X_j varies with X_i then conditioning on X_m will make no difference.

First we have to introduce the following definition:

Definition 6.1. A *potentially directed path* (p.d. path) in a CPAG is a path made only of edges of types $o \rightarrow$ and \rightarrow , with all arrowheads in the same direction. A p.d. path from X_i to X_j is denoted as $X_i \dashrightarrow X_j$.

6.1.4 Solving $o \rightarrow$

An overview of the different rules for solving $o \rightarrow$ is given in Table 6.1

For any edge $X_i o \rightarrow X_j$, there is no need to perform an experiment on X_j because we know that there can be no immediate influence of X_j on X_i , so we will only perform an experiment on X_i .

If $exp(X_i) \not\rightsquigarrow X_j$, then there is no influence of X_i on X_j , so we know that there can be no directed edge between X_i and X_j and thus the only remaining possibility is $X_i \leftrightarrow X_j$ (Type 1(a)).

If $exp(X_i) \rightsquigarrow X_j$, then we know for sure that there is an influence of X_i on X_j , we now need to discover whether this influence is immediate or via some intermediate variables.

| | |
|-----------|--|
| Type 1(a) | |
| Given | $X_i o \rightarrow X_j$ $exp(X_i) \not\rightarrow X_j$ |
| Action | $X_i \leftrightarrow X_j$ |
| Type 1(b) | |
| Given | $X_i o \rightarrow X_j$ $exp(X_i) \rightsquigarrow X_j$ \nexists p.d. path ($length \geq 2$) $X_i \dashrightarrow X_j$ |
| Action | $X_i \rightarrow X_j$ |
| Type 1(c) | |
| Given | $X_i o \rightarrow X_j$ $exp(X_i) \rightsquigarrow X_j$ \exists p.d. path ($length \geq 2$) $X_i \dashrightarrow X_j$ |
| Action | Block all p.d. paths by conditioning on blocking set \mathbf{Z} (a) $exp(X_i) \mathbf{Z} \rightsquigarrow X_j$; $X_i \rightarrow X_j$ (b) $exp(X_i) \mathbf{Z} \not\rightarrow X_j$; $X_i \leftrightarrow X_j$ |

Table 6.1. An overview of the different actions needed to complete edges of type $o \rightarrow$.

Therefore we make a difference whether there is a potentially directed (p.d.) path between X_i and X_j of length ≥ 2 , or not. If no such path exists, then the influence has to be immediate and the edge is found $X_i \rightarrow X_j$ (Type 1(b)).

If at least one p.d. path $X_i \dashrightarrow X_j$ exists, we need to block the influence of those paths on X_j while performing the experiment, so we try to find a blocking set \mathbf{Z} for all these paths. If $exp(X_i)|\mathbf{Z} \rightsquigarrow X_j$, then the influence has to be immediate, because all paths of length ≥ 2 are blocked, so $X_i \rightarrow X_j$. On the other hand if $exp(X_i)|\mathbf{Z} \not\rightarrow X_j$, there is no immediate influence and the edge is $X_i \leftrightarrow X_j$ (Type 1(c)).

6.1.5 Solving $o-o$

An overview of the different rules for solving $o-o$ is given in Table 6.2.

For any edge $X_i o-o X_j$, we have no information at all, so we might need to perform experiments on both variables.

If $exp(X_i) \not\rightarrow X_j$, then there is no influence of X_i on X_j so we know that there can be no directed edge between X_i and X_j and thus the edge is of the following form: $X_i \leftarrow o X_j$, which then becomes a problem of Type 1.

If $exp(X_i) \rightsquigarrow X_j$, then we know for sure that there is an influence of X_i on X_j , and as in the case of Type 1(b), we make a difference whether there is a potentially directed path between X_i and X_j of length ≥ 2 , or not. If no such path exists, then the influence has to be immediate and the edge must be turned into $X_i \rightarrow X_j$.

| | |
|-----------|---|
| Type 2(a) | |
| Given | $X_i o-o X_j$ $exp(X_i) \not\rightsquigarrow X_j$ |
| Action | $X_i \leftarrow o X_j (\Rightarrow \text{Type 1})$ |
| Type 2(b) | |
| Given | $X_i o-o X_j$ $exp(X_i) \rightsquigarrow X_j$ $\nexists \text{p.d. path (length} \geq 2) X_i \dashrightarrow X_j$ |
| Action | $X_i \rightarrow X_j$ |
| Type 2(c) | |
| Given | $X_i o-o X_j$ $exp(X_i) \rightsquigarrow X_j$ $\exists \text{p.d. path (length} \geq 2) X_i \dashrightarrow X_j$ |
| Action | Block all p.d. paths by conditioning on blocking set \mathbf{Z} (a) $exp(X_i) \mathbf{Z} \rightsquigarrow X_j: X_i \rightarrow X_j$ (b) $exp(X_i) \mathbf{Z} \not\rightsquigarrow X_j: X_i \leftarrow o X_j$ ($\Rightarrow \text{Type 1}$) |

Table 6.2. An overview of the different actions needed to complete edges of type $o-o$.

If at least one p.d. path $X_i \dashrightarrow X_j$ exists, we need to block the influence of those paths on X_j while performing the experiment, so we find a blocking set \mathbf{Z} like with Type 1(c). If $exp(X_i)|\mathbf{Z} \rightsquigarrow X_j$, then the influence has to be immediate, because all paths of length ≥ 2 are blocked, so $X_i \rightarrow X_j$. On the other hand if $exp(X_i)|\mathbf{Z} \not\rightsquigarrow X_j$, there is no immediate influence and the edge is of type: $X_i \leftarrow o X_j$, which again becomes a problem of Type 1.

6.1.6 Removing Inducing Path Edges

An inducing path between two variables X_i and X_j might create an edge between these two variables during learning because the two are dependent conditional on any subset of observable variables. As mentioned before, this type of edges is not present in SMCMs as it does not represent an immediate causal influence or a latent variable in the underlying DAG. We will call such an edge an *i-false* edge.

For instance, in Figure 5.1(a) the path X_1, X_2, L_1, X_6 is an inducing path, which causes the FCI algorithm to find an *i-false* edge between X_1 and X_6 , see Figure 5.1(c). Another example is given in Figure 6.1 where the SMCM is given in (a) and the result of FCI in (b). The edge between X_3 and X_4 in (b) is a consequence of the inducing path via the observable variables X_3, X_1, X_2, X_4 .

In order to be able to apply a causal inference algorithm we need to remove all *i-false* edges from the learned structure. We need to identify the substructures that can indicate

this type of edges. This is easily done by looking at any two variables that are connected by an immediate connection, and when this edge is removed, they have at least one inducing path between them. To check whether the immediate connection needs to be present we have to block all inducing paths by performing one or more experiments on an inducing path blocking set (i-blocking set) \mathbf{Z}^{ip} and block all other paths by conditioning on a blocking set \mathbf{Z} . If X_i and X_j are dependent, i.e., $X_i \nsim X_j$ under these circumstances the edge is correct and otherwise it can be removed.

In the example of Figure 5.1(c), we can block the inducing path by performing an experiment on X_2 , and hence can check that X_1 and X_6 do not covary with each other in these circumstances, so the edge can be removed.

In Table 6.3 an overview of the actions to resolve i-false edges is given.

| | |
|--------|--|
| Given | A pair of connected variables X_i, X_j A set of inducing paths X_i, \dots, X_j |
| Action | Block all inducing paths X_i, \dots, X_j by conditioning on i-blocking set \mathbf{Z}^{ip} . Block all other paths between X_i and X_j by conditioning on blocking set \mathbf{Z} . When performing all $exp(\mathbf{Z}^{ip}) \mathbf{Z}$: if $X_i \nsim X_j$: confounding is real else remove edge between X_i, X_j |

Table 6.3. Removing inducing path edges.

6.1.7 Example

We will demonstrate a number of steps to discover the completely oriented SMCM (Figure 5.1(b)) based on the result of the FCI algorithm applied on observational data generated from the underlying DAG in Figure 5.1(a). The result of the FCI algorithm can be seen in Figure 6.3(a). We will first resolve problems of Type 1 and 2, and then remove i-false edges. The result of each step is indicated in Figure 6.3.

- $exp(X_5)$
 - $X_5 o-o X_4$:
 $exp(X_5) \not\rightsquigarrow X_4 \Rightarrow X_4 o \rightarrow X_5$ (Type 2(a))
 - $X_5 o \rightarrow X_6$:
 $exp(X_5) \not\rightsquigarrow X_6 \Rightarrow X_5 \leftrightarrow X_6$ (Type 1(a))

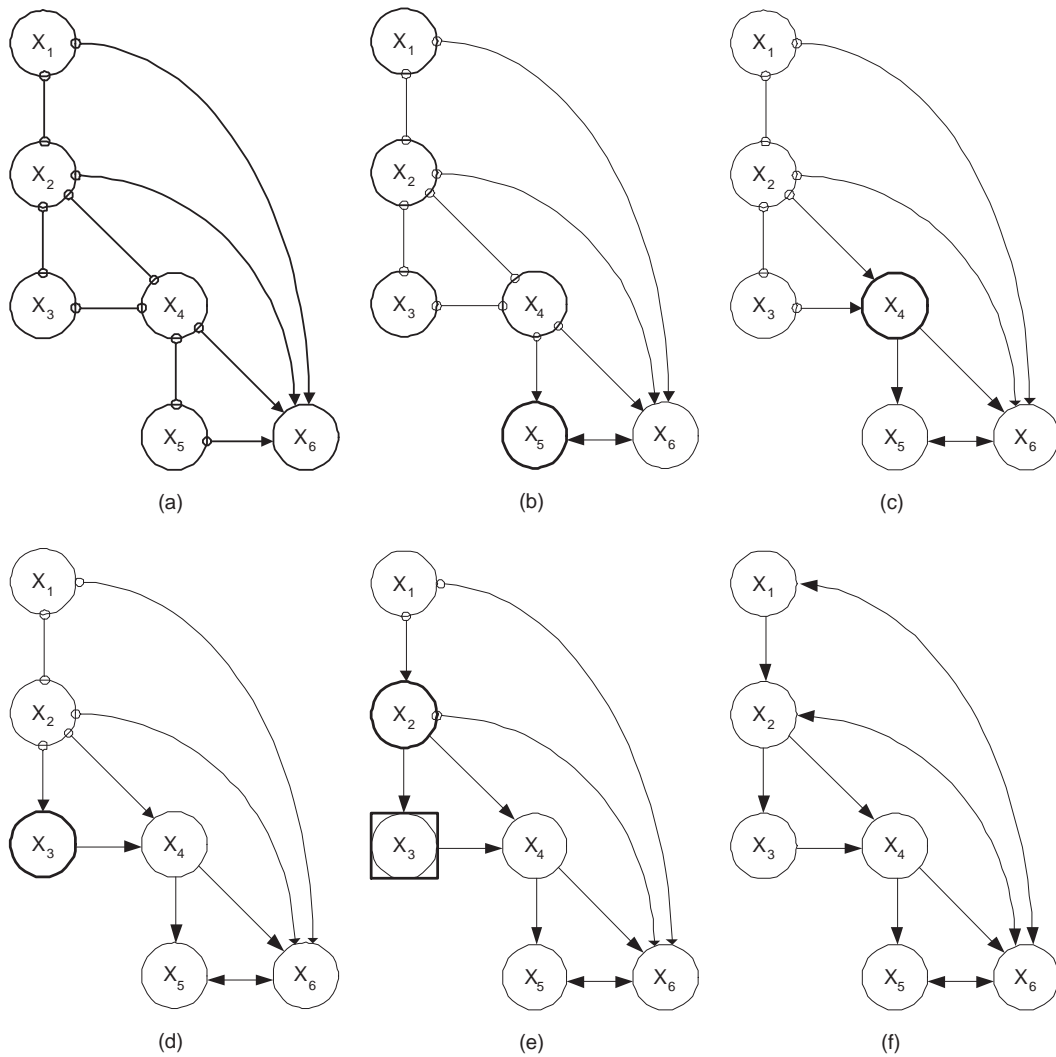


Figure 6.3. (a) The result of FCI on data of the underlying DAG of Figure 5.1(a). (b) Result of an experiment on X_5 . (c) After experiment on X_4 . (d) After experiment on X_3 . (e) After experiment on X_2 while conditioning on X_3 during the statistical test. (f) After resolving all problems of Type 1 and 2.

- $exp(X_4)$
 - $X_4 o-o X_2$:
 $exp(X_4) \not\rightsquigarrow X_2 \Rightarrow X_2 o \rightarrow X_4$ (Type 2(a))
 - $X_4 o-o X_3$:
 $exp(X_4) \not\rightsquigarrow X_3 \Rightarrow X_3 o \rightarrow X_4$ (Type 2(a))
 - $X_4 o \rightarrow X_5$:
 $exp(X_4) \rightsquigarrow X_5 \Rightarrow X_4 \rightarrow X_5$ (Type 1(b))
 - $X_4 o \rightarrow X_6$:
 $exp(X_4) \rightsquigarrow X_6 \Rightarrow X_4 \rightarrow X_6$ (Type 1(b))
- $exp(X_3)$

- $X_3 o-o X_2$:
 $exp(X_3) \not\rightsquigarrow X_2 \Rightarrow X_2 o \rightarrow X_3$ (Type 2(a))
- $X_3 o \rightarrow X_4$:
 $exp(X_3) \rightsquigarrow X_4 \Rightarrow X_3 \rightarrow X_4$ (Type 1(b))
- $exp(X_2)$ and $exp(X_2)|X_3$, because two p.d. paths between X_2 and X_4
 - $X_2 o-o X_1$:
 $exp(X_2) \not\rightsquigarrow X_1 \Rightarrow X_1 o \rightarrow X_2$ (Type 2(a))
 - $X_2 o \rightarrow X_3$:
 $exp(X_2) \rightsquigarrow X_3 \Rightarrow X_2 \rightarrow X_3$ (Type 1(b))
 - $X_2 o \rightarrow X_4$:
 $exp(X_2)|X_3 \rightsquigarrow X_4 \Rightarrow X_2 \rightarrow X_4$ (Type 1(c))

After resolving all problems of Type 1 and 2 we end up with the SMCM structure shown in Figure 6.3(f). This representation is no longer consistent with the MAG representation since there are bi-directed edges between two variables on a directed path, i.e., X_2, X_6 . There is a potentially i-false edge $X_1 \leftrightarrow X_6$ in the structure with inducing path X_1, X_2, X_6 , so we need to perform an experiment on X_2 , blocking all other paths between X_1 and X_6 (this is also done by $exp(X_2)$ in this case). Given that the original structure is as in Figure 5.1(a), performing $exp(X_2)$ shows that X_1 and X_6 are independent, i.e., $exp(X_2) : (X_1 \perp\!\!\!\perp X_6)$. Thus the bi-directed edge between X_1 and X_6 is removed, giving us the SMCM of Figure 5.1(b).

6.1.8 Critical Discussion of MyCaDo++

If all experiments that need to be performed are possible then MyCaDo++ will return the correct SMCM corresponding with the system. However, there is no attempt as to minimizing the cost of the experiments or the total number of experiments.

We want to research whether we can minimize the total cost of the experiments by using the orientation rules introduced in Zhang and Spirtes (2005a). The idea is to use a similar method as for MyCaDo (Section 4.1) where we intertwine the experiments and the orientation rules.

6.2 Parametrisation of SMCMs

As mentioned before in Section 5.3, Tian and Pearl provide an algorithm for performing causal inference given knowledge of the structure of an SMCM and the joint probability distribution (JPD) over the observable variables. However, they do not provide a parametrisation to efficiently store the JPD over the observables.

Algorithm 8 PR-representation algorithm.**Require:** A SMCM G and a topological ordering O over the variables V .**Ensure:** PR-representation R .

1. The nodes in R are \mathbf{V} , i.e., the observable variables of the SMCM.
2. The directed edges in R are the same as in the SMCM G .
3. The confounded edges in G are replaced by a number of directed edges in R as follows.
Add an edge from node X_i to node X_j iff:
 - a) $X_i \in (\mathbf{T}_j \cup Pa(\mathbf{T}_j))$, where \mathbf{T}_j is the c-component of G reduced to variables $\mathbf{X}^{(j)}$ that contains X_j ,
 - b) and there was not already an edge between nodes X_i and X_j .

6.2.1 Factorising with Latent Variables

Here we first introduce an additional representation for SMCMs, then we show how it can be parametrised and, finally we discuss how this new representation could be optimised.

PR-representation

Consider $X_{o_1} < \dots < X_{o_n}$ to be a topological order O over the observable variables \mathbf{V} , only considering the directed edges of the SMCM, and let $\mathbf{X}^{(i)} = X_{o_1} < \dots < X_{o_i}$, $i = 1, \dots, n$, and $\mathbf{X}^{(0)} = \emptyset$. We show how the parametrised (PR-) representation can be obtained from the original SMCM structure in Algorithm 8.

This way each variable becomes a child of the variables it would condition on in the calculation of the contribution of its c-component.

Figure 6.4(a) shows the PR-representation of the SMCM in Figure 5.1(a). The topological order that was used here is $X_1 < X_2 < X_3 < X_4 < X_5 < X_6$ and the directed edges that have been added are $X_1 \rightarrow X_5$, $X_2 \rightarrow X_5$, $X_1 \rightarrow X_6$, $X_2 \rightarrow X_6$, and, $X_5 \rightarrow X_6$.

The resulting DAG is an *I*-map (Pearl, 1988), over the observable variables of the independence model represented by the SMCM. This means that all the independences that can be derived from the new graph must also be present in the JPD over the observable variables. This property can be more formally stated as the following theorem.

Theorem 6.1. *The PR-representation PR derived from a SMCM S is an **I-map** of that SMCM.*

Proof. Consider two variables X_i and X_j in PR that are not connected by an edge. This means that they are not independent, since a necessary condition for two variables to be conditionally independent in a stable DAG is that they are not connected by an edge. PR is stable as we consider only stable problems.

Then, from the method for constructing PR we can conclude that S (i) contains no directed edge between X_i and X_j , (ii) contains no bi-directed edge between X_i and X_j , (iii)

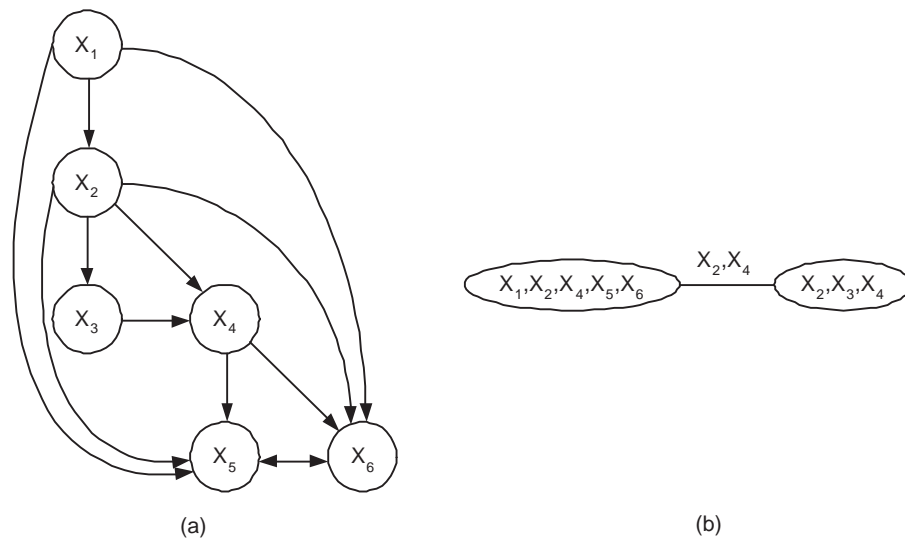


Figure 6.4. (a) The PR-representation applied to the SMCM of Figure 5.1(b). (b) Junction tree representation of the DAG in (a).

contains no inducing path between X_i and X_j . Property (iii) holds because the only inducing paths that are possible in a SMCM are those between a member of a c -component and the immediate parent of another variable of the c -component, and in these cases the method for constructing PR adds an edge between those variables. Because of (i),(ii), and (iii) we can conclude that X_i and X_j are independent in S .

Parametrisation

For this DAG we can use the same parametrisation as for classical BNs, i.e., learning $P(X_i|II_i)$ for each variable, where II_i denotes the parents in the new DAG. In this way the JPD over the observable variables factorises as in a classical BN, i.e., $P(\mathbf{V}) = \prod P(X_i|II_i)$, and these conditional probabilities map to the ones we need to calculate the c -factors. This follows immediately from the definition of a c -component and from the fact that each c -factor $Q[\mathbf{S}]$ can be calculated as follows (Tian and Pearl, 2002b):

Let $X_{o_1} < \dots < X_{o_n}$ be a topological order over \mathbf{V} , and let $\mathbf{X}^{(i)} = X_{o_1} < \dots < X_{o_i}$, $i = 1, \dots, n$, and $\mathbf{X}^{(0)} = \emptyset$.

$$Q[\mathbf{S}] = \prod_{X_i \in \mathbf{S}} P(X_i | (\mathbf{T}_i \cup Pa(\mathbf{T}_i)) \setminus \{X_i\}) \quad (6.1)$$

where \mathbf{T}_i is the c -component of the SMCM G reduced to variables $\mathbf{V}^{(i)}$, that contains \mathbf{V}_i . The SMCM G reduced to a set of variables $\mathbf{V}' \subset \mathbf{V}$ is the graph obtained by removing from the graph all variables $\mathbf{V} \setminus \mathbf{V}'$ and the edges that are connected to them.

Learning Parameters

As the PR-representation of SMCMs is a DAG as in the classical Bayesian network formalism, the parameters that have to be learned are $P(X_i|I_i)$. Therefore, techniques such as ML and MAP estimation, see Section 2.5.1, can be applied to perform this task.

6.2.2 Probabilistic Inference

One of the most famous existing probabilistic inference algorithm for models without latent variables is the *junction tree* algorithm (JT), see Section 2.4.2.

These techniques cannot immediately be applied to SMCMs for two reasons. First of all until now no efficient parametrisation for this type of models was available, and secondly, it is not clear how to handle the bi-directed edges that are present in SMCMs.

We have solved this problem by first transforming the SMCM into its PR-representation, which allows us to apply the junction tree inference algorithm. This is a consequence of the fact that, as previously mentioned, the PR-representation is an *I-map* over the observable variables. And as the JT algorithm is based only on independences in the DAG, applying it to an *I-map* of the problem gives correct results. See Figure 6.4(b) for the junction tree obtained from the parametrised representation in Figure 6.4(a).

Note that any other classical probabilistic inference technique that only uses conditional independences between variables could also be applied to the PR-representation.

6.2.3 Causal Inference

(Tian and Pearl, 2002b) developed an algorithm for performing causal inference, however as mentioned before in Section 5.4.1 they have not provided an efficient parametrisation.

Richardson and Spirtes (2003) show causal inference in AGs on an example, but a detailed approach is not provided and the problem of what to do when some of the parents of a variable are latent is not solved.

By definition in the PR-representation, the parents of each variable are exactly those variables that have to be conditioned on in order to obtain the factor of that variable in the calculation of the *c*-component, see Algorithm 8 and (Tian and Pearl, 2002b). Thus, the PR-representation provides all the necessary quantitative information, while the original structure of the SMCM provides the necessary structural information, for the algorithm by Tian and Pearl to be applied.

Critical Discussion of PR-representation

We have mentioned that the number of edges added during the creation of the PR-representation depends on the topological order of the SMCM.

As this order is not unique, choosing an order where variables with a lesser amount of parents have precedence, will cause less edges to be added to the DAG. This is because most of the added edges go from parents of c -component members to c -component members that are topological descendants.

By choosing an optimal topological order, we can conserve more conditional independence relations of the SMCM and thus make the graph more sparse, thus leading to a more efficient parametrisation.

In Section 8.1 we show how we can not only learn the parameters but also store them more efficiently by separating the SMCM into several smaller models.

6.3 Overview of the Chapter

We have discussed all classical steps in a modeling process such as learning the structure from observational and experimental data, model parametrisation, probabilistic and causal inference.

More precisely we showed that there is a big gap between the models that can be learned from data alone and the models that are used in causal inference theory. We showed that it is important to retrieve the fully oriented structure of a SMCM, and discussed how to obtain this from a given CPAG by performing experiments.

As the experimental learning approach relies on randomized controlled experiments, in general, it is not scalable to problems with a large number of variables, due to the associated large number of experiments. Furthermore, it cannot be applied in application areas where such experiments are not feasible due to practical or ethical reasons.

SMCMs have not been parametrised in another way than by the entire joint probability distribution, we showed that using an alternative representation, we can parametrise SMCMs in order to perform probabilistic as well as causal inference. Furthermore this new representation allows to learn the parameters using classical methods.

For future work, we want to study how to optimize the PR-representation, since the size of the CPDs is dependent on the topological order of the variables. We would like to see whether we can discover an optimal topological ordering.

Furthermore, we want to extend our MyCaDo algorithm presented in Section 4.1 to this setting with latent variables. Now we proposed an experimentation plan that shows all necessary experiments. We want to research whether we can minimize the total cost of the experiments or, maybe by using orientation rules during learning to minimize the total number of experiments.

Multi-Agent Causal Models

In this chapter we introduce multi-agent causal models (MACM) which were first proposed in (Maes et al., 2003). It is assumed that there is no longer one central controller having access to all the observable variables, but instead a collection of agents each having access to non-disjoint subsets \mathbf{V}_{M_i} of all the variables \mathbf{V} .

If we again reflect on the requirements of a good causal model (Section 1.2), we have to describe what exactly can be represented by a MACM (Section 7.1), which queries we can resolve within these frameworks (Section 7.2), and how they can be learned from observational or experimental data. Learning the structure of a MACM is part of our contribution and is discussed in the following chapter (Section 8.2).

7.1 Definition

We start by defining a multi-agent causal model:

Definition 7.1. A *multi-agent causal model* consists of n agents, each of which contains a semi-Markovian causal model M_i :

$$M_i = \langle \mathbf{V}_{M_i}, G_{M_i}, P_{M_i}(\mathbf{V}_{M_i}), \mathbf{K}_{M_i} \rangle \text{ for } i \in \{1, \dots, n\}.$$

- \mathbf{V}_{M_i} is the subset of variables agent- i can access.
- G_{M_i} is the causal graph over variables \mathbf{V}_{M_i} .
- $P_{M_i}(\mathbf{V}_{M_i})$ is the joint probability distribution over \mathbf{V}_{M_i} .
- \mathbf{K}_{M_i} stores the intersections $\{\mathbf{V}_{M_i} \cap \mathbf{V}_{M_j}\}$ with other agents j , denoted by \mathbf{V}_{M_i, M_j} . The variables in \mathbf{K}_{M_i} are called the **public variables** of an agent- i .

The set of variables of an agent- i without all the intersections he shares with other agents, or $\mathbf{V}_{M_i} \setminus \mathbf{K}_{M_i}$, is called the set of **private variables** of that agent.

Furthermore, for an MACM to be valid the following properties must hold:

- No edges (either directed or bi-directed) are allowed between private variables of different agents i and j .

- The intersection between two agents is modeled in the same way by each agent, i.e. the structure and distribution over the intersection should be the same.
- Combining the models of the individual agents does not introduce directed cycles.

Example 7.1. An example MACM adapted from Nadkarni and Shenoy (2001) is depicted in Figure 7.1. In this case $\mathbf{V}_{M_1} = \{X, X_1, \dots, X_9\}$ and $\mathbf{V}_{M_2} = \{X_7, \dots, X_{11}, Y\}$, while the intersection $\mathbf{V}_{1,2} = \{X_7, X_8, X_9\}$. For each agent the graph G_{M_i} is the part that is surrounded by an ellipse in the figure.

With this model a company wants to assess the influence of its *product pricing* strategy on *product decision*, this is whether a new product is launched or not. Furthermore, this company consists of two internal divisions, one modeled by agent1, roughly responsible for external issues such as the market situation or the competitive strategy, and one modeled by agent2 pertaining to internal issues such as research & development and the rate at which new products are launched.

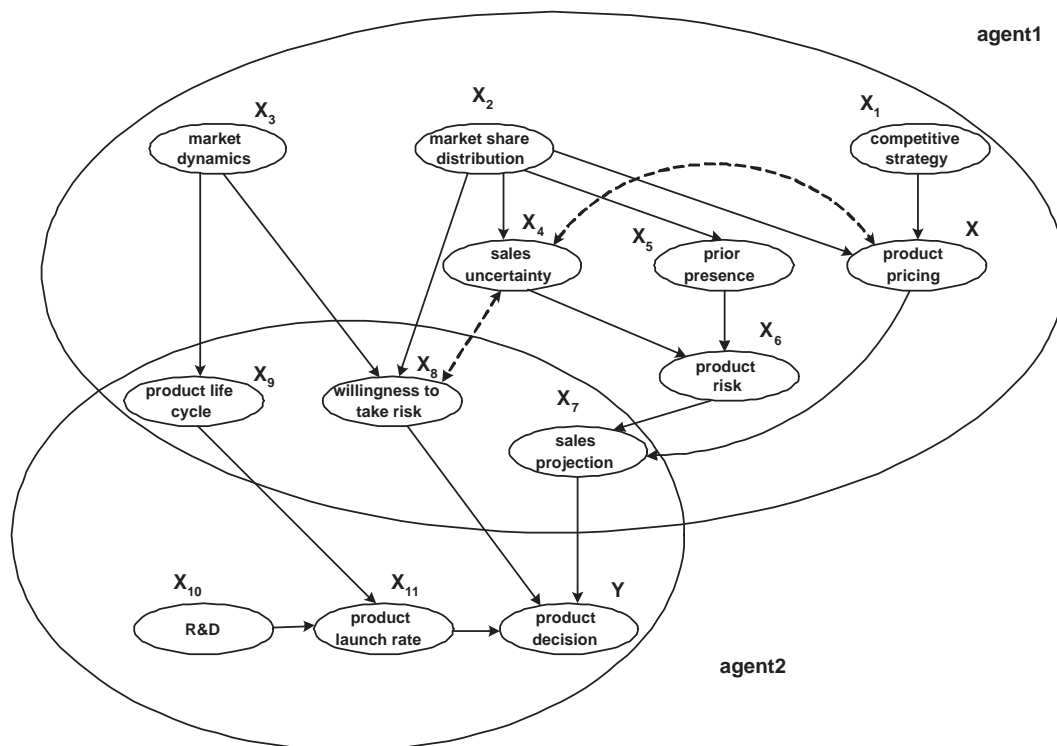


Figure 7.1. Example of a multi-agent causal model of a product decision model.

7.1.1 Additional Notations

The union of all the graphs G_{M_i} of all agents in the MACM is noted as $\cup G$, and it is the graph that would be associated with a single agent having access to the entire domain. Likewise, the union of all variables \mathbf{V}_{M_i} is denoted by $\cup \mathbf{V}$. For example, in Figure 7.1 $\cup \mathbf{V} = \{X, X_1, \dots, X_{11}, Y\}$ and $\cup G$ is the graph obtained when omitting the ellipses in that same figure.

We also introduce a notation for the variables of an *agent-i* without the intersection he shares with another *agent-j*: $\mathbf{V}_{M_i}^{\setminus M_j} = \mathbf{V}_{M_i} \setminus \mathbf{V}_{M_i, M_j}$. The private variables of an *agent-i*, i.e. the set of variables without all the intersections he shares with other agents are noted as $\mathbf{V}_{M_i}^{\setminus}$.

7.1.2 Properties of MACMs

In this section we will discuss the properties associated with the definition of multi-agent causal models.

JPD for each Agent

In Chapter 5 we have seen that at this time for semi-Markovian models the entire JPD over all observable variables is kept. In a MACM each *agent-i* stores the JPD over all variables \mathbf{V}_{M_i} that belong to its model. This has the advantage over a centralized semi-Markovian model, that we keep several JPDs over subsets of all the variables, instead of one JPD over all the variables. Thus this approach forms a more concise representation of the domain.

For example in the MACM of Figure 7.1, *agent1* would store $P(X, X_1, \dots, X_9)$ and *agent2* would store $P(X_7, \dots, X_{11}, Y)$, while in a single agent approach the JPD over all variables X, X_1, \dots, X_{11}, Y would be stored.

If a more efficient representation for semi-Markovian causal models is developed, this should also be incorporated in multi-agent causal models for them to keep the advantage of conciseness.

No Edges Between Different Agent Models

One of the properties of a valid MACM is that both directed and bi-directed edges between agent models are not allowed, except if both variables belong to the intersection. For example in Figure 7.1, a directed edge from variable X_6 to Y would not be allowed.

For a directed edge this simply means that no edge is allowed between private variables of different agents.

As a bi-directed edge represents an unobserved common cause, the fact that no bi-directed edges are allowed between private variables implies that unobserved variables

must be the common cause of variables belonging to the same agent for the multi-agent causal model to be valid. In domains where this is not the case, the intersection between agents has to be extended with some of the variables that are a child of an unobserved variables.

Intersection between Agents

Another property of multi-agent causal models is that both agents that share an intersection have to model those variables in the same way.

More specifically, this means that the structure and probabilities of the variables in the intersection must be the same in the model of both involved agents. Formally, consider *agent-i* and *agent-j* and the variables in the intersection are $\mathbf{I} = \mathbf{V}_{M_i, M_j}$. Then $G_{M_i, \mathbf{I}}$, or the graph G_{M_i} reduced to the variables in \mathbf{I} , must be equal to $G_{M_j, \mathbf{I}}$, and $P_{M_i}(\mathbf{I})$, or the joint probability distribution of *agent-i* over the variables in the intersection, must be equal to $P_{M_j}(\mathbf{I})$.

This is in accordance with the paradigm of cooperative multi-agent systems, where the agents do not hold back information concerning variables they consider as being shared.

Acyclicity

A property of multi-agent causal models is that combining individual agent models does not introduce directed cycles. The assumption of acyclicity is not always trivial to check in a distributed fashion as can be seen in Figure 7.2, where *agent1*, *agent2* and *agent3* are pairwise acyclic, but if we combine all three agent models there is the cycle $X_1, X_3, X_4, X_2, X_{14}, X_{11}, X_7, X_{10}, X_{12}, X_1$.

Individual Agent Models

The goal of a multi-agent causal model is to represent a complete domain correctly by several agents. Consequently, the semi-Markovian causal models in individual agent models M_i do not represent valid SMCs over their local subdomain consisting of the variables \mathbf{V}_{M_i} .

This is because in a multi-agent causal model each individual agent model M_i represents on one hand the independences and dependencies between its variables \mathbf{V}_{M_i} determined by subsets of these same variables (i.e. subsets of \mathbf{V}_{M_i} that make variables of \mathbf{V}_{M_i} independent). On the other hand, M_i also takes into account the independences found using sets of variables that are not subsets of \mathbf{V}_{M_i} , but some of which are modeled by other agents.

For example in the top model of Figure 7.3(a), the model of *agent1* takes into account that variables X_1 and X_2 are d -separated by private variables X_3 or X_4 in *agent2*. If *agent1* would

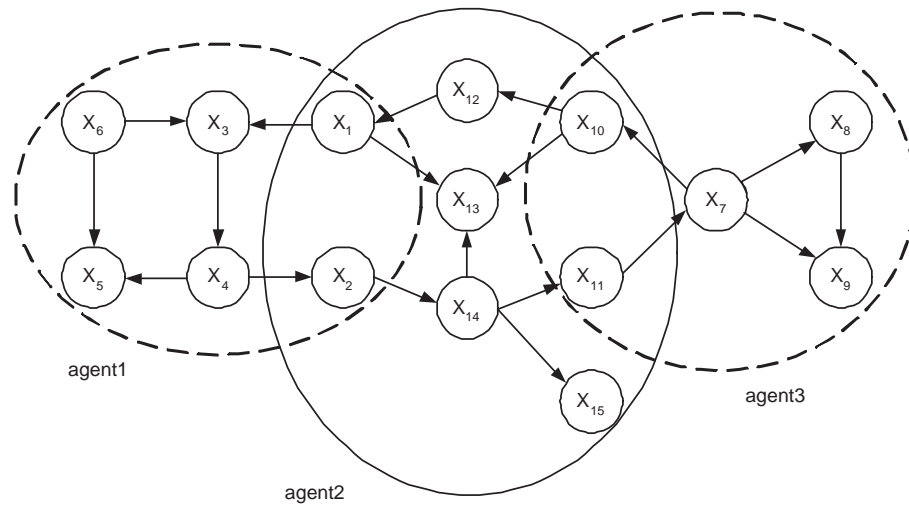


Figure 7.2. The agents in this MACM are pairwise acyclic, but the combination of all three agents introduces the directed cycle $X_1, X_3, X_4, X_2, X_{14}, X_{11}, X_7, X_{10}, X_{12}, X_1$.

only model independences that are determined by variables in V_{M_1} its correct model would be the bottom one of Figure 7.3(a).

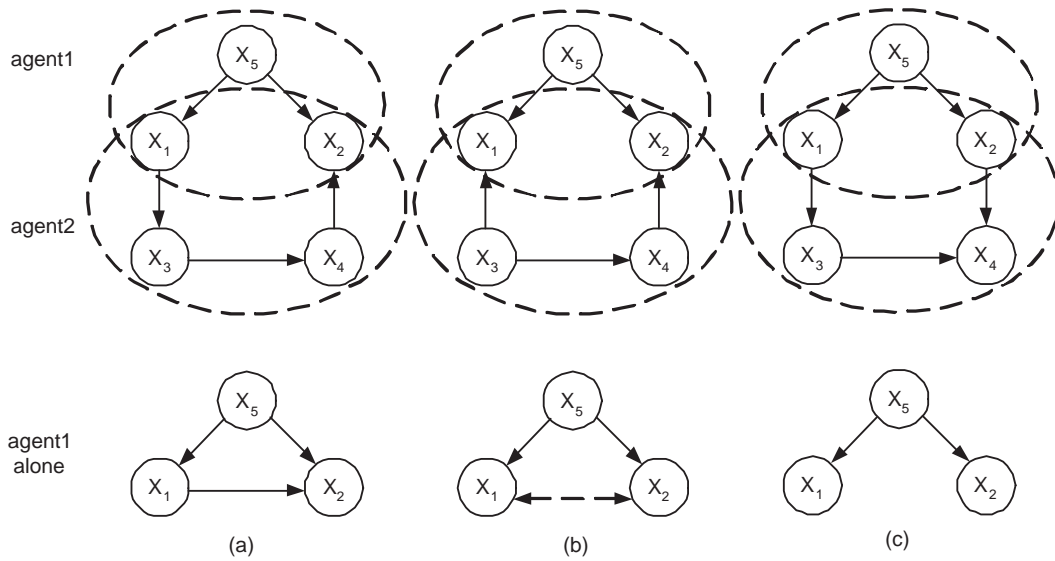


Figure 7.3. (a) A directed path between X_1 and X_2 in *agent2* is seen as a directed edge by *agent1*. (b) A path between X_1 and X_2 with a common cause in *agent2* is seen as a bi-directed edge by *agent1*. (c) A path between X_1 and X_2 that contains converging edges, is not seen by *agent1* as an edge of any type.

For an isolated individual agent model M_i of a MACM to constitute a valid semi-Markovian causal model of its subdomain, the private variables that are modeled by other

agents should be considered as unobserved variables to the agent model M_i for its local SMCM to be valid.

We are not interested in having a correct SMCM over individual agents' subdomain, but rather in a correct multi-agent causal model over the entire domain. Therefore, when learning a model it has to be taken into account that variables in one agent can be connected by a path of private variables in another agent, as we do not want an agent to model the private variables of another agent explicitly as unobserved variables.

Let the agent whose isolated model we are studying be *agent1* and the other agent *agent2*, the two connected variables are denoted by X_1 and X_2 . Then in a valid multi-agent causal model there are three possibilities for variables X_1 and X_2 to be connected via another agent:

- If the path between X_1 and X_2 in *agent2* is directed, then an isolated *agent1* would see a directed edge between the variables. See Figure 7.3(a) for an example.
- If the path between X_1 and X_2 in *agent2* contains at least one common cause of other variables on the path and no converging edges, then *agent1* would see a bi-directed edge between the variables if in isolation. This is because the common cause is hidden to *agent1* and thus it represents a confounding factor. See Figure 7.3(b) for an example. When there are converging edges on the path, it is handled by the next case.
- The path between X_1 and X_2 in *agent2* contains converging edges. In that case *agent1* does not see an edge between the variables, because X_1 and X_2 are marginally independent via that path. It is only when we condition on the variable where the edges converge that they become dependent, but as this variable is unobservable, conditioning on it is impossible from *agent1*'s point of view. As we can see in Figure 7.3(c), here there is no difference between the multi-agent case and the single agent case.

This property has important consequences for the learning of MACMs as will be discussed in Section 8.2.

Globally Correct Model

In the previous subsection we have seen that the model of an individual agent does not constitute a valid SMCM of its subdomain. On the other hand if the models of all individual agents are merged into one centralized model, then that model does constitute a valid SMCM of the complete domain.

For example, if in the top row MACMs of Figure 7.3, we omit the ellipses denoting the agent division we obtain semi-Markovian causal models that represent the complete domain correctly. We denote such models as *globally correct*.

Definition 7.2. In a MACM with models $M_i = \langle \mathbf{V}_{M_i}, G_{M_i}, P_{M_i}(\mathbf{V}_{M_i}), \mathbf{K}_{M_i} \rangle$ for $i \in \{1, \dots, n\}$, the semi-Markovian model associated with a single agent having access to the entire domain is called the **globally correct model**:

$$\cup M = \langle \cup \mathbf{V}, \cup G, P(\cup \mathbf{V}) \rangle \quad (7.1)$$

where

$$\begin{aligned} \cup \mathbf{V} &= \bigcup_{i \in \{1, \dots, n\}} \mathbf{V}_{M_i} \\ P(\cup \mathbf{V}) &= P\left(\bigcup_{i \in \{1, \dots, n\}} \mathbf{V}_{M_i}\right) \end{aligned}$$

so that $P(\mathbf{S}) = P_{M_i}(\mathbf{S})$ if $\mathbf{S} \subseteq P(\mathbf{V}_{M_i})$.

Finally, $\cup G$ denotes the graph obtained by merging all the individual local graphs G_{M_i} .

***d*-separation**

An important consequence of the previous property concerning individual agent models is that the *d*-separation criterion cannot be used in individual agent models alone to derive the independences implied by a model. In general, information from other agents is needed to obtain all the independences implied by a model.

More specifically, for a multi-agent causal model consisting of two agents, *agent-i* with model M_i and *agent-j* with model M_j , where

$$M_k = \langle \mathbf{V}_{M_k}, G_{M_k}, P_{M_k}(\mathbf{V}_{M_k}), \mathbf{V}_{M_i} \cap \mathbf{V}_{M_j} \rangle$$

for $k \in \{i, j\}$ and $\mathbf{V}_M = \mathbf{V}_{M_i} \cup \mathbf{V}_{M_j}$, we have the same property as for a single agent semi-Markovian causal model:

Property 7.1.

$$\begin{aligned} \forall X_i, X_j \in \mathbf{V}_M \text{ and } \exists \mathbf{S} \subset \mathbf{V}_M \setminus \{X_i, X_j\} : \\ X_i \perp X_j | \mathbf{S} \Leftrightarrow (X_i \perp\!\!\!\perp X_j | \mathbf{S}) \end{aligned} \quad (7.2)$$

Or in words, when two variables are *d*-separated by a set of variables in the union graph $\cup G$, this has to be mirrored by an equivalent statistical independence in the JPD $P(\cup \mathbf{V})$. Just as in the centralized case we work with faithful distributions, so the opposite relation should hold as well, i.e. every independence found in the JPD should be mirrored by a *d*-separation relationship in the graph.

The above property cannot be directly checked in multi-agent causal models, as it assumes that all variables are visible to every agent and for private variables of another agent in the MACM this is not the case. Note that even when all the variables in \mathbf{S} are in the private part of an *agent-i*, or $\mathbf{S} \subset \mathbf{V}_{M_i}^{\setminus}$, cooperation with another *agent-j* can be necessary to know whether $X_i \perp X_j | \mathbf{S}$.

Intersections of two Agents

In a multi-agent causal model, each intersection between agents consists of exactly two agents. In the case where more than two agents intersect at some point, this intersection is considered as multiple intersection between pairs of agents.

7.1.3 MACM and Cooperative MAS

The paradigm of cooperative multi-agent systems alleviates some of the problems associated with centralized approaches to modeling. In this section we will discuss how multi-agent causal models fulfill the properties we desire in a cooperative MAS.

Multi-Agent

MACMs are a multi-agent approach in the sense that the model of each individual agent has only access to its own local variables and that each individual agent cannot perform the task it is developed for, i.e. causal inference, without the help of other agents.

Autonomy

Although individual agent models do not constitute valid semi-Markovian causal models over their subdomains, the valid SMC can easily be obtained from its local data (it is even obtained as an intermediate step in the learning process of MACM). In that sense each individual agent model in a MACM is autonomous to a certain degree, because if communication with other agents in the MACM fails for some reason, the local agents can still model their local subdomain without much extra effort.

Cooperative

Multi-agent causal models are considered as a cooperative approach, because the agents do not hold back or provide wrong information concerning their public variables to other agents. In general once a valid MACM is obtained, information on private variables is not communicated to other agents, but in some cases, when the intersections between agents are too small and when all agents agree, there can be exceptions.

Furthermore, when learning a MACM from data, some information concerning private variables has to be disclosed, see Section 8.2 for more details.

Quantitative Causal Inference

We have seen earlier in this chapter that a centralized union of all individual agent models represents a valid semi-Markovian model of the complete domain and we have described an algorithm for performing causal inference in SMCMs. In a MACM we do not have explicit access to this SMCM of the complete domain, therefore in the following chapters we will introduce algorithms to perform quantitative causal inference in MACMs without violating the privacy of individual agents.

Unobserved Variables

As the agents use the paradigm of semi-Markovian causal models to model their local domain, unobserved variables can be taken into account in the modeling process.

7.2 Causal Inference in MACM

In this section, which describes joined work with Sam Maes presented in his dissertation (Maes, 2005), we start by introducing some auxiliary lemmas and proceed to an algorithm for calculating the causal effect $P(y|do(x))$ in a MACM consisting of 2 agents. We call this the *bi-agent causal model* (M_1, M_2) , where $M_i = \langle \mathbf{V}_{M_i}, G_{M_i}, P(\mathbf{V}_{M_i}), K_i \rangle$ for $i \in 1, 2$, where agent1 contains the intervention variable X and agent2 contains the variable to be studied Y .

7.2.1 Calculating c-factors

In this subsection we define c-factors and introduce a lemma stating that within some assumptions c-factors can be computed in a multi-agent manner, i.e. using only private information from an agent and its intersection with other agents.

We have previously explained how a semi-Markovian model can be partitioned into c-components. Assume that \mathbf{V} is thus partitioned into k c-components $\mathbf{S}_1, \dots, \mathbf{S}_k$ and denote by \mathbf{N}_j the set of unmeasured variables \mathbf{U} that are parents of those variables in \mathbf{S}_j . Then the c-factor $Q[\mathbf{S}_j]$ corresponding to a c-component \mathbf{S}_j is defined as (Tian and Pearl, 2002b):

$$Q[\mathbf{S}_j] = \sum_{\mathbf{n}_j} \prod_{\{i|X_i \in \mathbf{S}_j\}} P(x_i | \pi_i, \mathbf{u}^i) P(\mathbf{n}_j) \quad (7.3)$$

Due to the disjointness of $\mathbf{S}_1, \dots, \mathbf{S}_k$ and $\mathbf{N}_1, \dots, \mathbf{N}_k$, $P(\mathbf{v})$ can be decomposed into a product of $Q[\mathbf{S}_j]$'s:

$$P(\mathbf{v}) = \prod_{j=1}^k Q[\mathbf{S}_j] \quad (7.4)$$

For interpretations of c-factors and methods for calculating them in single agent models we refer the reader to (Tian and Pearl, 2002b) and the appendix. A method for computing c-factors in bi-agent causal models is introduced in the following lemma.

Lemma 7.1. (*Bi-agent c-factor calculation*)

Consider a bi-agent causal model (M_1, M_2) , where $M_i = \langle \mathbf{V}_{M_i}, G_{M_i}, P(\mathbf{V}_{M_i}), K_i \rangle$ for $i \in 1, 2$. Let a topological order over the union of the variables in both agents $\cup \mathbf{V}$ be $X_1 < \dots < X_n$ and let $X^{(i)} = \{X_1, \dots, X_i\}$.

If a c-component \mathbf{S}_j and its immediate parents are contained in exactly one agent, i.e. if $\mathbf{S}_j \cup Pa(\mathbf{S}_j) \subset \mathbf{V}_{M_1}$, then the c-factor $Q[\mathbf{S}_j]$ can be calculated using only information contained in that particular agent as follows:

$$Q[\mathbf{S}_j] = \prod_{\{i | X_i \in \mathbf{S}_j\}} P(x_i | pa(\mathbf{T}_i) \setminus \{x_i\}) \quad (7.5)$$

where \mathbf{T}_i is the c-component of $G_{X^{(i)}}$ that contains X_i .

Proof: Since $\mathbf{S}_j \cup Pa(\mathbf{S}_j) \subset \mathbf{V}_{M_1}$, all the elements involved in Equation (7.5) belong solely to agent-1 and thus the calculation can be done by that agent alone. \square

7.2.2 Recursive Domain Reduction

In the previous section, we proved that if the constituents of a c-component and its parents belong to the same agent, its c-factor can always be computed. Our approach to the identification problem will be to calculate the specific query from c-factors over c-components that are recursively becoming smaller, until the problem becomes trivially computable or unsolvable.

Therefore, we first introduce a lemma that formulates a way to compute the c-factor $Q[\mathbf{T}']$ from $Q[\mathbf{T}]$, where $\mathbf{T}' \subset \mathbf{T}$, and when some other assumptions hold.

Lemma 7.2. (*Calculation of $Q[\mathbf{T}']$*)

Given sets of variables $\mathbf{C} \subset \mathbf{A} \subset \mathbf{T}$, assume that $\mathbf{T}' \subset \mathbf{T}$ is the c-component of \mathbf{C} in $G_{\mathbf{A}}$ and that \mathbf{A}_i , $i = 1, \dots, l$ are the other c-components of $G_{\mathbf{A}}$. Also assume that $\mathbf{A} = Anc(\mathbf{C})_{G_{\mathbf{T}}}$. Furthermore, let $A_{h_1} < \dots < A_{h_k}$ be a topological order over the k variables of \mathbf{A} and $A^{(i)} = \{A_{h_1}, \dots, A_{h_i}\}$, the set of variables ordered before A_{h_i} including A_{h_i} .

Then, $Q[\mathbf{T}']$ is identifiable from $Q[\mathbf{T}]$ and given by

$$Q[\mathbf{T}'] = \frac{Q[\mathbf{A}]}{\prod_i Q[\mathbf{A}_i]} \quad (7.6)$$

Where,

$$Q[\mathbf{A}] = \sum_{\mathbf{T} \setminus \mathbf{A}} Q[\mathbf{T}] \quad \text{and}, \quad (7.7)$$

$$Q[\mathbf{A}_i] = \prod_{\{j | A_{h_j} \in \mathbf{A}_i\}} \frac{Q[A^{(j)}]}{Q[A^{(j-1)}]} \quad \text{and}, \quad (7.8)$$

$$Q[A^{(j)}] = \sum_{\mathbf{A} \setminus A^{(j)}} Q[\mathbf{A}] \quad (7.9)$$

Proof: We know from (Tian and Pearl, 2002b) that the c-factor $Q[\mathbf{A}]$ of a group of variables \mathbf{A} is the product of the c-factors of the c-components \mathbf{A}_i of \mathbf{A} ,

$$Q[\mathbf{A}] = \prod_i Q[\mathbf{A}_i] \quad (7.10)$$

If one of the \mathbf{A}_i is renamed to \mathbf{T}' , Equation (7.6) can be derived trivially from the equality above.

For Equation (7.7): we know from Lemma 4 in (Tian and Pearl, 2002b) that for any $\mathbf{W} \subseteq \mathbf{C}$, $Q[\mathbf{W}] = \sum_{\mathbf{C} \setminus \mathbf{W}} Q[\mathbf{C}]$, if \mathbf{W} is an ancestral set in the subgraph $G_{\mathbf{C}}$ (i.e., $\mathbf{W} = \text{Anc}(\mathbf{W})_{G_{\mathbf{C}}}$). From $\mathbf{A} = \text{Anc}(\mathbf{C})_{G_{\mathbf{T}}}$ follows that \mathbf{A} is an ancestral set in \mathbf{T} , i.e., $\mathbf{A} = \text{Anc}(\mathbf{A})_{G_{\mathbf{T}}}$ and thus $Q[\mathbf{A}] = \sum_{\mathbf{T} \setminus \mathbf{A}} Q[\mathbf{T}]$.

Equation 7.8 is an application of Lemma 5 in (Tian and Pearl, 2002b) and finally, since each $A^{(j)}$ is an ancestral set in \mathbf{A} by definition, Lemma 4 of (Tian and Pearl, 2002b) can be used to marginalize over the c-factor $Q[A]$ to obtain Equation 7.9.

□

Note that Lemma 7.2 uses no variables outside \mathbf{T} , so if \mathbf{T} is entirely contained in the model of one agent, the calculation of $Q[\mathbf{T}']$ from $Q[\mathbf{T}]$ in a setting as above, can happen without knowledge external to that agent.

In Algorithm 9 we introduce a recursive algorithm *Identify* for calculating $Q[\mathbf{C}]$ from $Q[\mathbf{T}]$ in general, if this is possible, by making use of Lemmas 7.1 and 7.2. If \mathbf{T} is completely contained in 1 agent ($\mathbf{T} \subset \mathbf{V}_{M_1}$ in this case), the algorithm respects the privacy of the two agents in the sense that it only uses variables from \mathbf{V}_{M_1} in its computation.

In the first case \mathbf{C} is an ancestral set in \mathbf{T} , and thus $Q[\mathbf{C}]$ can be calculated from $Q[\mathbf{T}]$ with the help of Lemma 7.1 as $\mathbf{T} \subset \mathbf{V}_{M_1}$. In the second case, when the ancestral set of \mathbf{C} in \mathbf{T} is

Algorithm 9 Bi-agent *Identify*($\mathbf{C}, \mathbf{T}, Q[\mathbf{T}]$)**Require:** sets of variables $\mathbf{C} \subset \mathbf{T} \subset \mathbf{V}_{M_1}$.**Ensure:** Expression for $Q[\mathbf{C}]$ in terms of $Q[\mathbf{T}]$ using only variables \mathbf{V}_{M_1} .Let $\mathbf{A} = \text{Anc}(\mathbf{C})_{G_T}$ {= the ancestors of \mathbf{C} in graph G restricted to variables \mathbf{T} }

1. IF $\mathbf{A} = \mathbf{C}$, output $Q[\mathbf{C}] = \sum_{\mathbf{T} \setminus \mathbf{C}} Q[\mathbf{T}]$
2. IF $\mathbf{A} = \mathbf{T}$, return *#false*
3. IF $\mathbf{C} \subset \mathbf{A} \subset \mathbf{T}$
 - a) In $G_{\mathbf{A}}$, \mathbf{C} is contained in a c -component \mathbf{T}' .
 - b) Compute $Q[\mathbf{T}']$ from $Q[\mathbf{A}] = \sum_{\mathbf{T} \setminus \mathbf{A}} Q[\mathbf{T}]$ by Lemma 7.2
 - c) return *Identify*($\mathbf{C}, \mathbf{T}', Q[\mathbf{T}']$)

equal to \mathbf{T} itself, we know of no way to calculate $Q[\mathbf{C}]$ from $Q[\mathbf{T}]$ at this time ¹. In the final case, when the ancestral set \mathbf{A} of \mathbf{C} in \mathbf{T} is a strict set in between \mathbf{C} and \mathbf{T} , we make use of Lemma 7.2 to formulate a recursive call of *Identify*, where \mathbf{T} is replaced by its subset \mathbf{T}' , which is the c -component of \mathbf{C} in \mathbf{T} . Specifically, Lemma 7.2 is used to calculate $Q[\mathbf{T}']$ from $Q[\mathbf{T}]$. *Identify* will always terminate in one of the two first cases after a finite number of steps.

7.2.3 Computation of $P(y|do(x))$

Here we present an algorithm for computing $P(y|do(x))$ in a bi-agent causal model, based on the lemmas and algorithms introduced earlier in this section.

Consider a bi-agent causal model (M_1, M_2) , as before. As stated before, X belongs to agent1 ($X \in \mathbf{V}_{M_1}$) and Y belongs to agent2 ($Y \in \mathbf{V}_{M_2}$). Then the underlying single agent semi-Markovian model has variables $\cup \mathbf{V} = \mathbf{V}_{M_1} \cup \mathbf{V}_{M_2}$ and as graph $\cup G$ the union of the graphs G_{M_1} and G_{M_2} . We assume that each of the c -components $\mathbf{S}_1, \dots, \mathbf{S}_k$ of $\cup \mathbf{V}$ and its parents $Pa(\mathbf{S}_i)$ are completely contained in either agent1 (\mathbf{V}_{M_1}) or agent2 (\mathbf{V}_{M_2}), or formally $\forall \mathbf{S}_i : (\mathbf{S}_i \cup Pa(\mathbf{S}_i) \subset \mathbf{V}_{M_1}) \vee (\mathbf{S}_i \cup Pa(\mathbf{S}_i) \subset \mathbf{V}_{M_2})$. This assumption will allow Lemma 7.1 to be applied.

Identification Algorithm

In Algorithm 10 we introduce a bi-agent algorithm for the identification of $P(y|do(x))$. In step 1 both agents calculate the c -factors of their c -components using Lemma 7.1. Note that for the variables in the intersection between the two agents, the c -factor is only calculated by the agent that contains its parents.

In step 2 the agents communicate information concerning their intersection $\mathbf{V}_{M_1} \cap \mathbf{V}_{M_2}$ to obtain the ancestors of Y in agent1, and next the intersection \mathbf{D}^X with \mathbf{S}^X is obtained.

¹ Do we know it can not be calculated in principle ?

Next, in step 3 the resulting \mathbf{D}^X is divided into c-components in agent1 and for each c-component \mathbf{D}_j^X Function 9 (*Identify*) is called.

Finally, the results of the *Identify* calls are combined to calculate $P(y|do(x))$. The part of the equation on line (7.12) pertains only to variables of agent1 and the part on line (7.13) only to variables of agent2. After their respective summations, distributions are obtained that only consist of $X \cup \mathbf{K}_1$ for agent1 and $Y \cup \mathbf{K}_2$ for agent2. Hence, the resulting distribution of agent2 can be sent to agent1 without disclosing information except the intersection and the variable Y that is being studied in this specific query. Agent1 multiplies this distribution with its own result and performs the summation over $\mathbf{D}_1 \cap (\mathbf{D}_2 \setminus Y)$ to obtain the final result.

Algorithm 10 Bi-agent identification of $P(y|do(x))$

Require: variables X and Y .

Ensure: $P(y|do(x))$ if it is identifiable, otherwise *#false*.

1. Calculate all c-factors $Q[\mathbf{S}_k^i]$ separately in each agent- $i \in 1, 2$ using Lemma 7.1. The c-factor of the c-component that contains X is identified by $Q[\mathbf{S}^X]$ and is contained in agent1. The c-factors of the variables in the intersection are calculated in the agent that contains their parents.
2. Find $\mathbf{D}_i = \text{Anc}(Y)_{G_{V_{M_i} \setminus \{X\}}}$ in each agent- $i \in 1, 2$ and $\mathbf{D}^X = \mathbf{D}_1 \cap \mathbf{S}^X$.
3. Let the c-components of $G_{\mathbf{D}^X}$ be $\mathbf{D}_j^X, j = 1, \dots, l$.

For each set \mathbf{D}_j^X :

Call *Identify*($\mathbf{D}^X, \mathbf{S}^X, Q[\mathbf{S}^X]$) given in Function 9 to calculate $Q[\mathbf{D}_j^X]$. If the algorithm returns *#false*, then stop and output *#false*.

Otherwise, output:

$$P(y|do(x)) = \sum_{\mathbf{D}_1 \cap (\mathbf{D}_2 \setminus Y)} \quad (7.11)$$

$$\sum_{\mathbf{D}_1 \setminus \mathbf{K}_1} \left[\prod_j Q[\mathbf{D}_j^X] \prod_k \sum_{\mathbf{S}_k^1 \setminus \mathbf{D}_1} Q[\mathbf{S}_k^1] \right] \quad (7.12)$$

$$\sum_{\mathbf{D}_2 \setminus (\mathbf{K}_2 \cup Y)} \left[\prod_k \sum_{\mathbf{S}_k^2 \setminus \mathbf{D}_2} Q[\mathbf{S}_k^2] \right] \quad (7.13)$$

Now we will prove equation (7.11,7.12,7.13).

Proof: This equation is based on Equation (106) in (Tian and Pearl, 2002b) for identification in single agent models, see the same article for a proof of that formula.

$Q[\mathbf{D}_j^X]$ and $Q[\mathbf{S}_k^1]$ consist only of elements in agent1 and $\mathbf{D}_2 \setminus (\mathbf{K}_2 \cup Y)$, consists only of elements in agent2, thus the summation over $\mathbf{D}_2 \setminus (\mathbf{K}_2 \cup Y)$ can be brought to the front. Likewise, $Q[\mathbf{S}_k^2]$ is calculated solely from elements in agent2 and $\mathbf{D}_1 \setminus \mathbf{K}_1$ is part of agent1's model. Combining the summations over $\mathbf{D}_1 \cap (\mathbf{D}_2 \setminus Y)$, $\mathbf{D}_1 \setminus \mathbf{K}_1$, and $\mathbf{D}_2 \setminus (\mathbf{K}_2 \cup Y)$ yields a summation over $\mathbf{D} \setminus Y$:

$$P(y|do(x)) = \sum_{\mathbf{D} \setminus Y} \prod_j Q[\mathbf{D}_j^X] \prod_k \sum_{\mathbf{S}_k \setminus \mathbf{D}} Q[\mathbf{S}_k] \quad (7.14)$$

This is the single agent formula from Equation (106) in (Tian and Pearl, 2002b). \square

So we have introduced an algorithm for the identification of $P(y|do(x))$ in a bi-agent causal model, where each agent combines confidential information stored in its local model with information concerning the intersection with the other agent and the variables being studied (in this case X and Y). In this algorithm no information concerning other variables than the intersection and the variables X and Y is being disclosed.

7.2.4 Example

If we apply our algorithm to the product decision model in Figure 7.1, with $X = \text{product pricing}$ and $Y = \text{product decision}$ we get $\mathbf{S}^X = \{X, X_4, X_8\}$ and $Q[\mathbf{S}^X] = P(x|x_1, x_2) P(x_4|x, x_1, x_2) P(x_8|x, x_1, x_2, x_3, x_4)$.

$\mathbf{D}_1 = \text{Anc}(Y)_{G_{V \setminus X}} = \{X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9\}$ and $\mathbf{D}^X = \{X_4, X_8\}$, then in the call $\text{Identify}(\{X_4, X_8\}, \mathbf{S}^X, Q[\mathbf{S}^X])$, $\mathbf{A} (= \text{Anc}(\{X_4, X_8\})_{G_{S^X}} = \{X_4, X_8\})$ will be equal to \mathbf{C} and $Q[X_4, X_8] = \sum_X Q[\mathbf{S}^X]$ will be returned.

Finally,

$$P(y|do(x)) = \sum_{X_7, X_8, X_9} \sum_{X_2, \dots, X_6} \left[\sum_X Q[\mathbf{S}^X] \prod_i \sum_{\mathbf{S}_k^1 \setminus \mathbf{D}_1} Q[\mathbf{S}_k^1] \right] \quad (7.15)$$

$$\sum_{X_{10}, X_{11}} \left[\prod_i \sum_{\mathbf{S}_k^2 \setminus \mathbf{D}_2} Q[\mathbf{S}_k^2] \right] \quad (7.16)$$

where the $Q[\mathbf{S}_k^A]$ can be calculated using Lemma 7.1 as follows. For agent1:

$$\begin{aligned} S_1^1 &= \{X_1\}; S_2^1 = \{X_2\}; S_3^1 = \{X_3\}; S_4^1 = \{X_5\} \\ S_5^1 &= \{X_6\}; S_6^1 = \{X_7\}; S_7^1 = \{X_9\} \end{aligned}$$

$$Q[X_1] = P(x_1)$$

$$Q[X_2] = P(x_2)$$

$$Q[X_3] = P(x_3)$$

$$Q[X_5] = P(x_5|x_2)$$

$$Q[X_6] = P(x_6|x_4, x_5)$$

$$Q[X_7] = P(x_7|x, x_6)$$

$$Q[X_9] = P(x_9|x_3)$$

For agent2:

$$S_1^2 = \{x_{10}\}; S_2^2 = \{x_{11}\}; S_3^2 = \{y\}$$

$$Q[X_{10}] = P(X_{10})$$

$$Q[X_{11}] = P(X_{11}|X_9, X_{10})$$

$$Q[Y] = P(Y|X_7, X_8)$$

7.3 Overview of the Chapter

In this chapter we introduced a new type of causal models that does not assume that all knowledge about the studied system is localized in a central site but dispersed over multiple agents. Furthermore, we proposed a causal inference algorithm for this setting. The algorithm assumes that the different agents are linked in a chain structure.

In the next section we will relax this final assumption by proposing a different causal inference algorithm. We also will propose a structure learning algorithm for MACM.

Contributions on MACM

In this chapter we propose our contributions for multi-agent causal models. We will introduce an alternative causal inference algorithm and demonstrate beginning steps to learn the structure of MACMs.

We start by showing how we can store the JPD corresponding to a SMCM more concisely by separating it into a multiplication of CPDs. Then, we demonstrate how this helps to divide a SMCM into a MACM in which each agent stores a c-component and the parents of all variables in the c-component.

The motivation for an alternative causal inference algorithm stems from the fact that the one proposed in the previous chapter is only applicable for agents that are linked in a tree structure. The algorithm proposed in this chapter only allows the calculation of the effect of an intervention on the entire set of variables that constitutes the MACM. It is viewed as a starting point for a more general inference algorithm.

In the later sections we introduce the basis of a structure learning algorithm for MACM, this is part of joined work with Sam Maes and had been discussed prior in his dissertation (Maes, 2005).

8.1 Separation of MACM

In (Tian and Pearl, 2002b,a) a decomposition of the joint probability distribution for a SMCM is proposed. We will introduce this decomposition here and will use it to explicitly factorize the JPD.

8.1.1 Factorization of the JPD

Remember that the set of variables \mathbf{V} of a SMCM can be partitioned into so called c-components \mathbf{S}_j by grouping variables that are connected by a bi-directed path. Denote by

\mathbf{N}_j the set of \mathbf{U} variables that are parents of variables in \mathbf{S}_j . The sets \mathbf{N}_j form a partition on \mathbf{U} . Define the c-factor $Q[\mathbf{S}_j]$ of a c-component \mathbf{S}_j as:

$$Q[\mathbf{S}_j] = \sum_{\mathbf{n}_j} \prod_{\{i|X_i \in \mathbf{S}_j\}} P(x_i | \pi_i, u^i) P(\mathbf{n}_j) \quad (8.1)$$

Remember that the JPD $P(\mathbf{v})$ can be written as (cf. Section 5.3.4):

$$P(\mathbf{v}) = \prod_j Q[\mathbf{S}_j] \quad (8.2)$$

Furthermore each $Q[\mathbf{S}_j]$ is computable from $P(\mathbf{v})$. Let a topological order over \mathbf{V} be $X_1 < \dots < X_n$, and let $X^{(i)} = X_1, \dots, X_i$ and $X^0 = \emptyset$, then:

$$Q[\mathbf{S}_j] = \prod_{\{i|X_i \in \mathbf{S}_j\}} P(x_i | x^{(i-1)}) \quad (8.3)$$

It can be shown that $Q[\mathbf{S}_j]$ is a function of $Pa(\mathbf{S}_j)$ with $Pa(\mathbf{S}_j) = \mathbf{S}_j \cup (\bigcup_{X_i \in \mathbf{S}_j} \Pi_i)$.

This factorization can be used to store the JPD more concise in a SMCM, while in (Tian and Pearl, 2002a) JPD was assumed to be stored completely. The way this can be done is by joining nodes in a c-component \mathbf{S}_j and the parents of the node in \mathbf{S}_j together (similar to making a j-tree) and storing probability distributions over subsets of \mathbf{V} .

We store $P(\mathbf{S}_j | Pa(\mathbf{S}_j) \setminus \mathbf{S}_j)$ for each group $Pa(\mathbf{S}_j)$, basically dividing the original SMCM into several smaller SMCMs that store the respective CPD over fewer variables. This can diminish the size needed to store the JPD over all variables depending on the structural properties of the original SMCM.

For instance in Figure 8.1 we would store the following probabilities:

- $Q[X_1] = P(X_1)$
- $Q[\{X_3, X_5\}] = P(X_5 | X_3, X_1, X_4) P(X_3 | X_1) = P(X_5 | X_1, X_3, X_4) P(X_3 | X_1, X_4) = P(X_3, X_5 | X_1, X_4)$
- $Q[\{X_2, X_4\}] = P(X_2, X_4)$
- $Q[X_6] = P(X_6 | X_4)$
- $Q[X_7] = P(X_7 | X_5, X_6)$

In Section 6.2 we have shown how to create a Bayesian network (BN) in which the parameters $P(X_i | \Pi_i)$ can be mapped to the *c-factors* of a given SMCM. The parameters of the constructed BN can obviously be learned from data and therefore we are able to learn the corresponding c-factors.

The factorization of the JPD and the division of the original SMCM allows us to extend the framework of multi-agent causal models (MACM) proposed in (Maes et al., 2007b). In this previous work we designed an algorithm for the identification of causal effects in a

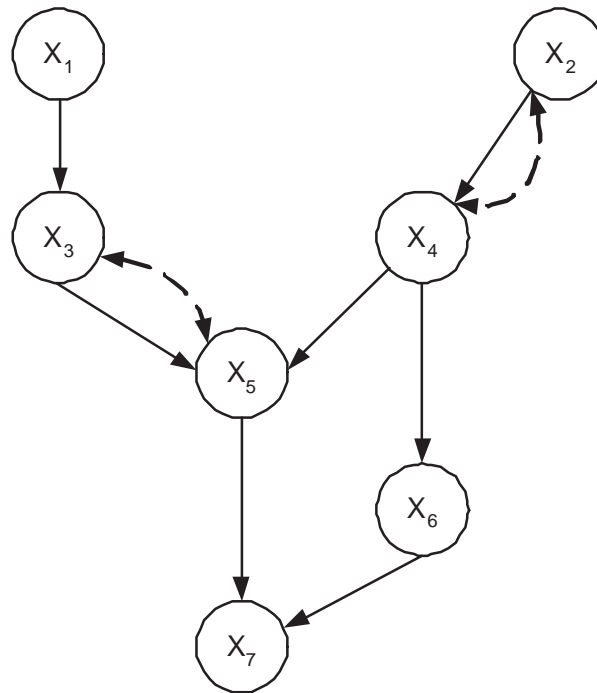


Figure 8.1. Example SMCM used to demonstrate decomposition into MACM.

multi-agent approach. The main advantages of the multi-agent solution is that the identification of causal effects can be assessed without disclosing sensitive information of a local model to other agents. It allows to perform causal inference in situations where parts of the model are kept confidential by their distributors. An algorithm was provided for tree-structured MACM in (Maes et al., 2007b).

MACM were studied mostly from the view of a distributed problem domains. Here we work in reverse and we decompose a centralized SMCM into several models using the factorization discussed in the previous section. Furthermore, we introduce an alternative algorithm to perform causal inference, which is the process of calculating the effect of an intervention on a (set of) variable(s) on some other variables, in the corresponding MACM by exchanging information between agents.

8.1.2 From SMCM to MACM

Based on the factorization and the discussion at the end of the previous section the division is straightforward. Assume we are given a SMCM M . We will create an agent for each component S_j in M . The model M_j of *agent* $- j$ consists of:

- $\mathbf{V}_{M_j} = Pa(\mathbf{S}_j)$
- $\mathbf{U}_{M_j} = N_j$

- $G_{\mathbf{v}U_{M_j}} = G_{Pa(\mathbf{S}_j)}$
- $P_{M_j}(x_i|\pi_i, u^i), x_i \in \mathbf{S}_j$
- $P_{M_j}(\mathbf{n}_j)$

For any pair of agents $agent - i$ and $agent - j$, the intersection of their variables is given by $Pa(\mathbf{S}_i) \cap Pa(\mathbf{S}_j)$.

The resulting agents models are exactly those substructures we identified in Section 8.1.1.

For instance in the SMCM shown in Figure 8.1, there are 5 c-components: $\{\{X_1\}, \{X_3, X_5\}, \{X_2, X_4\}, \{X_6\}, \{X_7\}\}$. The MACM division corresponding to the SMCM in Figure 8.1 is shown in Figure 8.2.

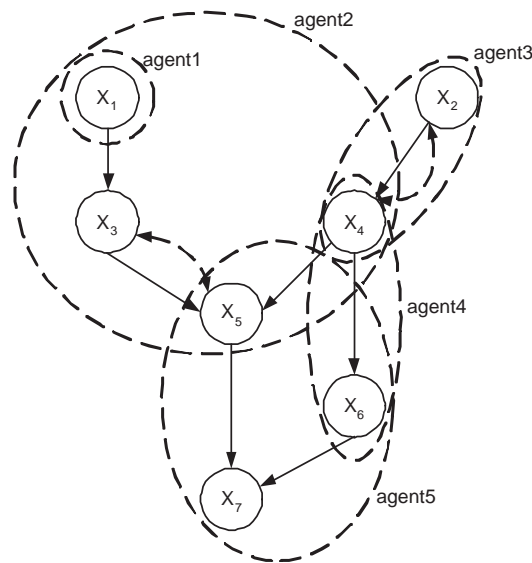


Figure 8.2. The resulting separating MACM from the original SMCM given in Figure 8.1.

8.1.3 Alternative Causal Inference

In this section we show how to check whether a causal effect $P(e|do(x))$ is identifiable in a MACM. We also show how to compute $P(\mathbf{v}|do(x))$ if possible. We proposed a causal inference algorithm in Section 7.2 however that algorithm was limited to bi-agent or chain-multi agent causal models. Here we present an algorithm that can perform causal inference where the agents are linked in a more complex structure. Furthermore the previous algorithm relied on communicating over shared variables while we now present an algorithm that works by passing information on local JPDs.

In a SMCM the following theorem holds (Tian and Pearl, 2002c):

Theorem 8.1. $P(v|do(x))$ is identifiable if and only if there is no bi-directed path connecting M to any of its children. When $P(v|do(x))$ is identifiable, it is given by:

$$P(v|do(x)) = Q_x[\mathbf{S}^X] \prod_j Q[\mathbf{S}_j] \quad (8.4)$$

with \mathbf{S}^X the c -component to which X belongs and $Q_x[\mathbf{S}^X]$ the c -factor $Q[\mathbf{S}^X]$ with the term $P(x|\pi_x, u^x)$ removed.

We assume that the agent that contains the c -component that contains X is the agent asking for the calculation of $P(v|do(x))$, and hence the necessary information has to be communicated to this agent. Remember that the causal effect is given by Equation (8.4), and that each $Q[\mathbf{S}_j]$ can be calculated by Equation (8.3).

Remember we can rewrite the latter equation as given in Equation 6.1.

Each factor $P(x_i | (\mathbf{T}_i \cup Pa(\mathbf{T}_i)) \setminus \{X_i\})$ can be calculated using only variables in \mathbf{S}_j since each $\mathbf{T}_i \subseteq \mathbf{S}_j$. This means that each agent that does not contain the c -component \mathbf{S}^X can calculate $Q[\mathbf{S}_j]$ independently and hence communicate this information to its neighbors and eventually to the agent containing X .

So all that rests is to calculate $Q_x[\mathbf{S}^X]$ and multiply this result with $\prod_j Q[\mathbf{S}_j]$. We can write $Q_x[\mathbf{S}^X]$ as:

$$Q_x[\mathbf{S}^X] = \sum_{n^X} \prod_{\{i|X_i \neq X, X_i \in \mathbf{S}^X\}} P(x_i | \pi_i, u^i) P(n^X) \quad (8.5)$$

Furthermore, since we assume that the causal effect is identifiable, there is no bi-directed path between X and any of its children and we can write (Tian and Pearl, 2002b):

$$Q_x[\mathbf{S}^X] = \sum_x Q[\mathbf{S}^X] \quad (8.6)$$

We know that $Q[\mathbf{S}^X]$ can be calculated by the agent containing \mathbf{S}^X and since it also contains X itself we can calculate $Q_x[\mathbf{S}^X]$. Together with the received information from the other agents we can now calculate $P(v|do(x))$ by using Equation 8.4.

8.1.4 Example

Assume that we are working in the MACM show in Figure 8.2 and that the query we want to calculate is $P(x_2, x_3, x_4, x_5, x_6, x_7 | do(x_1))$. We assume that is the agent that contains X_1 (agent1) which sends the request for the query and hence needs to receive all the necessary information. For the example we'll assume that the subresults $Q[\mathbf{S}_j]$ can be transmitted to X_1 directly. If for instance communicating was only possible between neighbors then partial

results should be forwarded and combined. Note that this is possible as each factor $Q[\mathbf{S}_j]$ can be calculated independently.

In a first step, agent1 forwards the instantiations $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ of all variables in the query to the respective agents. As a second step, when an agent receives the instantiation of all its variables it can calculate its proper $Q[\mathbf{S}_j]$. Let's look at the necessary calculations for agent2 and agent3, containing variables $\{X_1, X_3, X_4, X_5\}$ and $\{X_2, X_4\}$ respectively. We need to calculate $Q[\{X_3, X_5\}]$ (because this is the c-component used to create the model for agent-2) and $Q[\{X_2, X_4\}]$, which according to Equation (6.1) are equal to:

$$Q[\{X_3, X_5\}] = P(x_3|x_1)P(x_5|x_1, x_3, x_5) \quad (8.7)$$

$$Q[\{X_2, X_4\}] = P(x_4)P(x_2|x_4) = P(x_2, x_4) \quad (8.8)$$

Similar calculations are performed in every agent and in a third step this result is then forwarded to agent-1.

The final step that needs to be performed is calculating $Q_{x_1}[\mathbf{S}^{X_1}]$, which is trivial in this case, and then multiplying this by all the received $Q[\mathbf{S}_j]$ from the other agents.

8.1.5 Critical Discussion on Separation of MACM

The simple inference algorithm described here is not particularly effective since it only allows the calculation of the effect of an intervention on the probability of a joined state of all other variables $P(\mathbf{v}|do(x))$. Also the information that is communicated between the agent is sufficient to learn the distribution of the private variables of an agent by performing a sufficient number of queries.

The technique described above can be used as a starting point for a causal inference algorithm to answer each causal query for agents linked in an arbitrary manner. This is however a whole research area in itself that falls outside the scope of this dissertation.

8.2 Learning MACMs

In this section we will discuss how a multi-agent causal model can be obtained. Just as in the single agent case, models can be obtained from data, experts or a combination of both. However, in this section we will only focus on learning MACM from data.

We will start by mentioning some related approaches in the literature and then move on to discuss our setting for learning MACMs and the assumptions associated with it. Then we will discuss the problems that are characteristic to the learning of MACM. Finally, we will present some preliminary solutions to these problems.

8.2.1 Related Approaches

Chen et al. (2003), and Chen et al. (2004) treat the distributed learning of Bayesian networks from distributed heterogeneous data. It is assumed that there are no latent variables in the global model and that the given observational data is *perfect* (see Section 3.4.1).

In that setting there are different sites that each have data concerning subsets of all the domain variables. An assumption of their work is that the data is strictly heterogeneous, i.e. that there is no overlap between the variables at different sites.

Their approach consists of an agent associated to each site, that learns a local Bayesian network at each site based on the local data. Then each agent identifies the observations that are most likely to be evidence of coupling between variables local to that site and non-local variables. After that the agent transmits a representative subset of these observations to a central agent.

The central agent learns another BN using the data transmitted from the different local sites. The local and central BNs are combined to obtain a collective BN, that models the entire data.

For the central agent to learn a model based on correct data, it is necessary that each data point that is sent to it has a key that links it to associated data points at other sites. Without such a key, no correct data on the joint instantiation of variables from different sites would be available and consequently no model on these models can be learned.

An important difference with our setting is that in this case there is no overlap between the sets of variables associated with each site and that this approach learns a single collective BN that represents the entire distributed domain.

Because the overlap of variable sets is a keystone of our approach and a MACM is a distributed representation, we extend their approach to incorporate these properties in the rest of this section.

8.2.2 Setting and Assumptions

In this subsection we will discuss the general setting in which we try to learn a multi-agent causal model and the assumptions associated with it.

First of all, we only study a setting where there are only two different sites S_{M_i} and S_{M_j} each containing measurements from sets of variables \mathbf{V}_{M_i} and \mathbf{V}_{M_j} respectively. For extensions to more than two agents, the principles of learning a MACM as introduced would remain the same, only the exchange of private information between the agents would become more intricate.

Furthermore we assume that in the global model there are no latent variables.

The samples are taken from a distribution $\cup P$ over a set of variables $\cup \mathbf{V} = \mathbf{V}_{M_i} \cup \mathbf{V}_{M_j}$ faithful to some CBN $\cup G$ without unobserved variables. The intersection between the sets of variables at both sites is non-empty, or $\mathbf{V}_{M_i, M_j} = \mathbf{V}_{M_i} \cap \mathbf{V}_{M_j} \neq \emptyset$.

There is an agent *agent-i* at each site which learns a model over its local variables, this can be done using a combination of learning algorithms and expert knowledge.

We assume that all agents use the same *constraint based* learning technique such as PC discussed earlier in Chapter 2 to learn the structure from the available local observational data. We say that each such learned model M_k is *locally correct* with respect to the variables \mathbf{V}_{M_k} local to that site.

Definition 8.1. We say that a model is *locally correct* with respect to \mathbf{V}_{M_i} , when it learns the model that we expect when the \mathbf{V}_{M_i} are considered as observed variables and the private variables of other agents *agent-j*, $\mathbf{V}_{M_j}^{\setminus}$ as unobserved variables.

For example, consider Figure 8.3. In (a) we see the underlying correct semi-Markovian causal model of the domain, where the circles indicate the division per site. With each site an agent is associated that learns a locally correct model and in (b) we show such a model for *agent-i*. Because that agent cannot observe variable X_4 , it concludes that there is a directed edge from variables X_3 to X_2 .

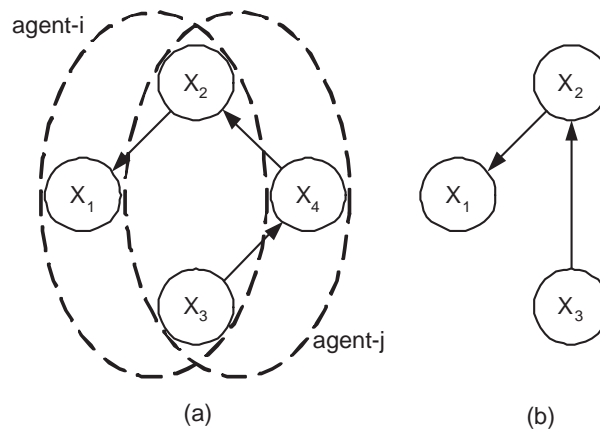


Figure 8.3. (a) The underlying SMCM of the domain, with the division per site. (b) The locally correct model that an *agent-i* would learn, based on $\mathbf{V}_{M_i} = \{X_1, X_2, X_3\}$ alone. It concludes that there is a directed edge between variables X_3 and X_2 .

However, as we have seen before in Section 7.1.2, in a MACM the individual agent models do not constitute locally correct models, but they have to take into account independences induced by variables in other agents. Therefore, the agents have to cooperate to discover such independences.

To recapitulate, our learning approach is based on the following assumptions:

1. We learn a MACM consisting of two agents *agent-i* and *agent-j*, with respective models $M_k = \langle \mathbf{V}_{M_k}, G_{M_k}, P_{M_k}(\mathbf{V}_{M_k}), \mathbf{K}_{M_k} \rangle$ for $k \in \{i, j\}$. In this case $\mathbf{K}_{M_k} = \mathbf{V}_{M_i, M_j}$ for both agents.
2. The samples are taken from a distribution $\cup P = P(\mathbf{V}_{M_i} \cup \mathbf{V}_{M_j})$ faithful to a causal Bayesian network, $\cup G$, without hidden variables.
3. There are no edges from variables of one site that are not in the intersection to variables that are not in the intersection of the other site, or between $\mathbf{V}_{M_i} \setminus$ and $\mathbf{V}_{M_j} \setminus$ in $\cup G$.
4. At both sites, each sample is marked with a unique key (e.g. a time stamp) so the observations of the two sites can be linked together.

The second assumption is used because faithful distribution have some favorable characteristics, such as the following property:

Property 8.1. If a distribution P over variables \mathbf{V} is faithful to a CBN with DAG G , and $X_i, X_j \in V$, then:

If X_i and X_j are d-separated by any subset of $\mathbf{V} \setminus \{X_i, X_j\}$, then they are d-separated either by Π_i or Π_j (Spirtes et al., 2000a).

The third assumption gives us the possibility to identify variables whose neighbors are all in the same agent, these variables will be the *private* variables in the MACM. A consequence of this assumption is that all variables with neighbors in different agents are situated in the intersection.

The last assumption is also present in the related approach that was discussed in the previous section. Just like there it is essential because some steps in the algorithm may require to perform tests on variables from different agents, and thus it has to be possible to obtain joint data on variables from different sites.

In this general setting the following properties hold:

Theorem 8.2. *Dependencies in the underlying global distribution $P(\cup \mathbf{V})$ between private variables $\mathbf{V}_{M_i} \setminus$ of an agent- i can be correctly obtained using only variables in \mathbf{V}_{M_i} .*

Proof:

Using Property 8.1 and the third assumption, it is clear that all parents of variables in $\mathbf{V}_{M_i} \setminus$ are in \mathbf{V}_{M_i} , and so the correct dependencies can be found.

Theorem 8.3. *The following dependencies in the underlying global distribution $P(\cup \mathbf{V})$ can be obtained correctly using all the variables in $\mathbf{V}_{M_i} \cup Ne(\mathbf{V}_{M_i, M_j})$ in $\cup G$, where $Ne(\mathbf{V}_{M_i, M_j})$ stands for the neighbors of \mathbf{V}_{M_i, M_j} in the underlying global graph $\cup G$.*

Dependencies between:

- variables in the intersection V_{M_i, M_j} , and,
- variables in $V_{M_i}^{\setminus}$ of an agent- i and the intersection V_{M_i, M_j} .

Proof:

Again using Property 8.1 it is clear that $V_{M_i} \cup Ne(V_{M_i, M_j})$ must contain the variables that d -separate the variables belonging to these sets.

However, the variables necessary in Theorem 8.3, $V_{M_i} \cup Ne(V_{M_i, M_j})$ in UG , are not visible by a single agent. So to check for independences concerning the variables mentioned in Theorem 8.3, observations from variables of both sites and a way to link them together is needed.

Note that we have not studied algorithms to combine samples from different sites to form a minimal sufficient set, instead we assume that all samples concerning a variable will be disclosed.

8.2.3 Problems

In this section we show by use of examples some of the problems that can arise when initiating the task of learning the structure of MACMs based on conditional independence tests.

All discussed problems are a consequence of the fact that the individual agents learn locally correct models. Instead in a MACM we want the individual agent models to take into account that independences can be induced by private variables of other agents.

For the moment, the problems that we take into account here will not focus on the direction of the edges or their causal interpretation, but rather on the presence or absence of certain edges.

Definition 8.2. *An edge is **g-false** (globally false) if it is present in the locally correct model but not in the correct global one.*

For example, in Figure 8.3(b), the edge between C and B is g -false.

In general, there are two main sorts of g -false edges:

- g -false edges in the intersection, i.e. between variables that are in the intersection, and,
- g -false edges outside the intersection, i.e. between variables that are not both in the intersection.

We will treat both types next.

g -false Edges in the Intersection

In this case there are two sorts of g -false edges that can emerge in the intersection:

- only one agent locally creates the g -false edge, and,
- both agents locally create the g -false edge.

Below we give an example of both types.

In Figure 8.4 a situation is shown in which only *agent2* would conclude that there is a link between two variables $X2$ and $X3$ in the intersection. This is due to the path $X2 \leftarrow X1 \rightarrow X3$ in which $X1$ is unobservable for *agent2*.

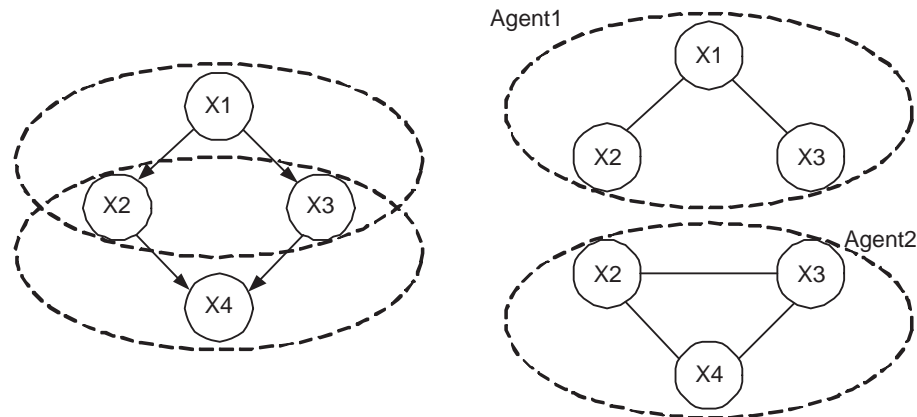


Figure 8.4. Example of the edges found when the global network (left) is divided into two agents (right). *Agent2* detects a g -false edge between $X2$ and $X3$ using only local information in his independence tests.

In Figure 8.5 a situation is shown in which both agents would conclude that there is an edge between two variables $X4$ and $X5$ in the intersection, due to the paths $X4 \leftarrow X2 \leftarrow X1 \rightarrow X3 \rightarrow X5$ in *agent1* and $X4 \rightarrow X6 \rightarrow X5$ in *agent2*.

Both in Figure 8.4 and Figure 8.5 the g -false edges are a direct consequence of a path in the other agent that can not be blocked by the first agent. To solve this problem one of the agents should disclose information about a set of private variables that would block all paths inducing this g -false edge.

g -false Edges outside the Intersection

It might seem that under our assumptions there can only be g -false edges in the intersection, however this is not the case. For example, in Figure 8.6 where the path $X4 \leftarrow X3 \leftarrow X1 \rightarrow X2$ induces the g -false edge $X2 - X4$.

8.2.4 Multi-Agent Causal Learning Algorithm

In this section we will introduce a simple algorithm for learning the structure of MACMs. Before going to the actual algorithm we introduce some crucial theorems.

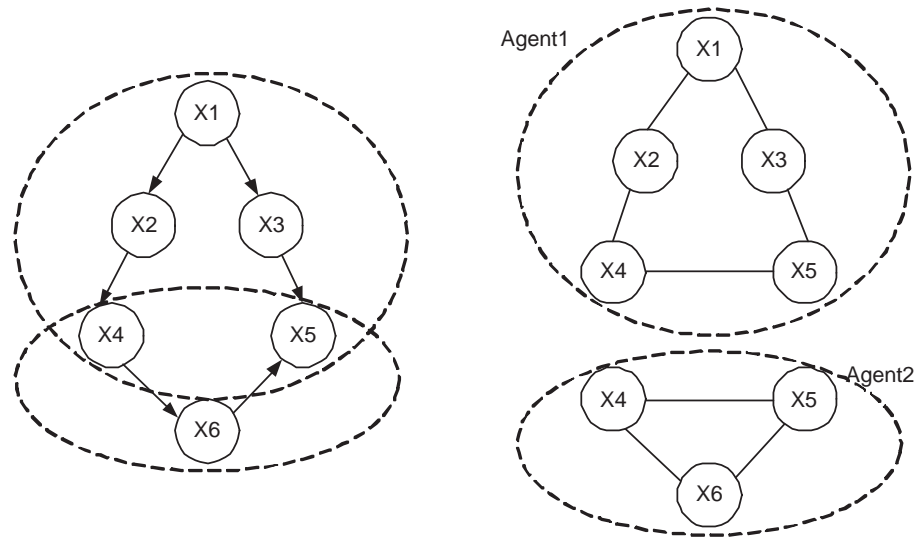


Figure 8.5. Both agents detect a g -false edge between $X4$ and $X5$ using only local information.

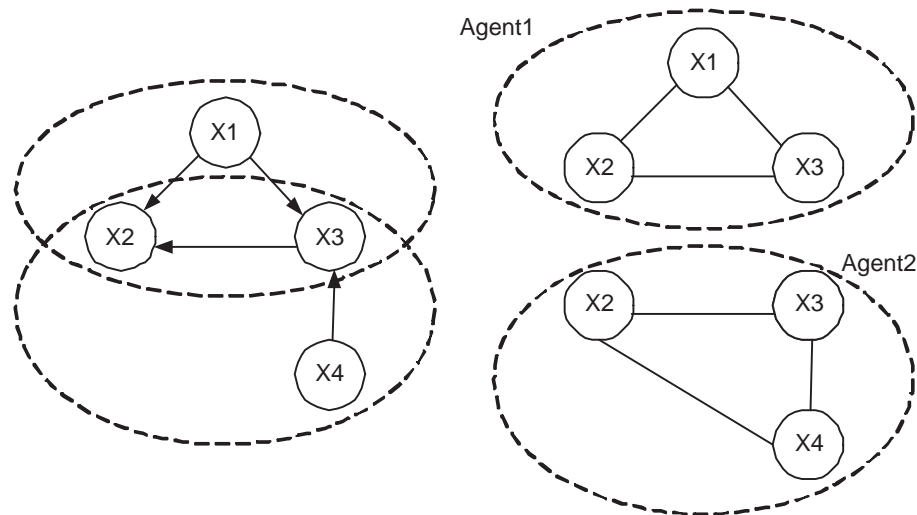


Figure 8.6. Problem with g -false edges concerning one variable in the intersection $X2$ and one private variable of $agent2$, $X4$.

Theorems

The first theorem states that locally correct models contain at least all the desired edges.

Theorem 8.4. *When the individual agents learn locally correct models, then each of these models contains at least all the edges that are desired for it to constitute a valid agent model of a MACM.*

Proof:

Without loss of generality, we can consider an $agent1$ with variables \mathbf{V}_{M_1} , and an agent $agent2$ with variables \mathbf{V}_{M_2} .

In Section 7.1.2 we have seen that when in a MACM we omit the division in agents, by making the union of all individual agents models of a valid MACM, we obtain the semi-Markovian causal model (SMCM) associated with the domain where a central single agent would be able to observe all the variables of the domain $\cup \mathbf{V} = \mathbf{V}_{M_1} \cup \mathbf{V}_{M_2}$.

From Definition 8.1, we know that an individual agent model is locally correct when it learns based on the observed variables \mathbf{V}_{M_1} and considers all the other variables as unobserved variables. Furthermore, constraint based learning techniques only add an edge between two variables $X_i, X_j \in \mathbf{V}_{M_1}$, when no set of variables \mathbf{S} is found that makes X_i and X_j independent.

Consequently, when in the centralized SMCM associated with the valid MACM there is an edge between X_i and X_j , then there is no set $\mathbf{S} \subset \cup \mathbf{V}$, that makes X_i and X_j independent. From that it follows that in $\mathbf{V}_{M_1} \subset \cup \mathbf{V}$ we can also not find a set to make X_i and X_j independent, and therefore there will also be an edge between X_i and X_j in the locally correct model.

For example in Figure 8.6 we can see that all the edges in the valid MACM (on the left), are present in the locally correct models (on the right).

The consequence of Theorem 8.4 for a multi-agent learning algorithm is that all edges that are necessary in a MACM are present by learning locally correct models. This implies that the only problem that can arise in our setting is that too many edges are present in the locally correct models.

Therefore we introduce the second theorem, it states that when an agent does not find an edge between two variables, then there also is no edge in the valid MACM, even if other agents do find an edge between the same variables.

Theorem 8.5. *When the individual agents learn locally correct models, and when one agent finds an edge between variables in the intersection and another agents does not find it, then that edge is not present in the desired MACM.*

Proof:

Without loss of generality, we can consider two agents *agent1* and *agent2* with respective variable sets \mathbf{V}_{M_1} and \mathbf{V}_{M_2} . The two agents share a non-empty intersection $\mathbf{V}_{M_1, M_2} = \mathbf{V}_{M_1} \cap \mathbf{V}_{M_2}$.

When *agent1* finds no edge between two variables $X_i, X_j \in \mathbf{V}_{M_1, M_2}$ in its locally correct model, this means that there is a set $\mathbf{S}_{M_1} \subset \mathbf{V}_{M_1}$ that makes X_i and X_j independent.

If *agent2* does find an edge between X_i and X_j in its locally correct model, this means that there is no set $\mathbf{S}_{M_2} \subset \mathbf{V}_{M_2}$ that makes X_i and X_j independent.

Now, as soon as an agent finds a set of variables \mathbf{S}_{M_1} to make X_i and X_j independent in its locally correct model, this means that there is a way to make X_i and X_j independent in the

domain. Therefore there can never be an edge between X_i and X_j in the SMCM associated with the valid MACM with variable set $\cup \mathbf{V} = \mathbf{V}_{M_1} \cup \mathbf{V}_{M_2}$.

The fact that *agent2* does find an edge between the variables in its locally correct model, merely means that it has no access to variables that can make both variables independent.

See Figure 8.7 for an example. In (a) we see the valid MACM, in (b) the locally correct model of *agent1* where no edge is found between X_1 and X_2 , since conditioning on X_3 makes them independent. In (c) the locally correct model of *agent2* can be seen, and there is an edge between X_1 and X_2 since *agent2* cannot condition on variable X_3 of *agent1* to make them independent.

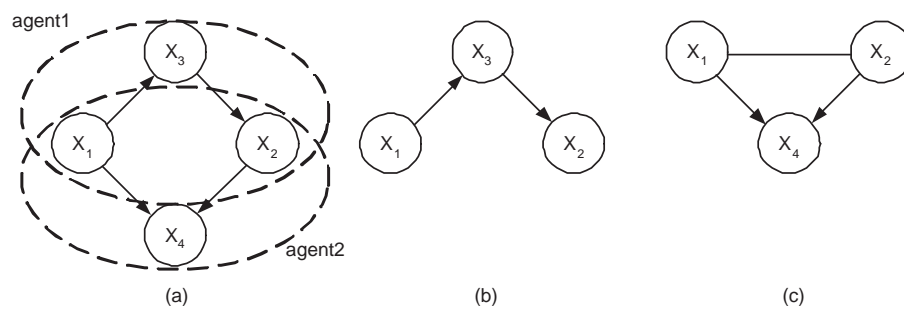


Figure 8.7. (a) the valid multi-agent causal model, (b) the locally correct model of *agent1*, (c) the locally correct model of *agent2*.

Now to recapitulate, Theorem 8.4 stated that there can be no missing edges, but only edges too many. Theorem 8.2 states that the presence of edges between private variables can be learned locally. For variables that are in the intersection, Theorem 8.5 determines what to do if the locally correct models of both agents do not agree whether there should be an edge between the variables.

Then there are still two cases left that are not handled by the above:

- both agents falsely see an edge between two variables in the intersection in their locally correct models, and,
- an agent falsely sees an edge between a private variable and a variable in the intersection.

To solve these problems the agents will have to exchange private knowledge, i.e. each agents' neighbors of the intersection, or $Ne(\mathbf{V}_{M_i, M_j})$, as stated in Theorem 8.3.

Algorithm

The multi-agent causal discovery algorithm (M-A CaDo) consists of three phases, first each agent learns a local model, then the agents exchange sample data, and finally local knowl-

edge is combined with knowledge received from other agents. The complete algorithm is given in Algorithm 11.

Algorithm 11 M-A CaDo

Require: A group of agents A_i each having access to an overlapping subset of variables \mathbf{V}_i .

Ensure: A MACM.

Phase 1: Local learning Using Theorem 8.2 we know that in the locally learned model the dependencies are correct for all variables in $\mathbf{V}_{M_i}^{\setminus}$ and possibly some independences between $\mathbf{V}_{M_i}^{\setminus}$ and \mathbf{V}_{M_i, M_j} are found as well. Note that these are removed correctly, since no two variables can be dependent in the global model if they are independent in the locally correct model.

Phase 2: Negotiation Each *agent-i* sends a message to the other demanding the samples from the private variables in $Ne(\mathbf{V}_{M_i, M_j})$. If the other agent is not willing to disclose this information the algorithm returns FAIL, otherwise *agent-i* now has knowledge on all the necessary variables to compute the full dependence structure of \mathbf{V}_{M_i} as stated by Theorem 8.3.

Alternate Phase 2: Communication with neutral agent Each *agent-i* sends the samples to a specific neutral agent that will perform the statistical tests and communicate to each agent the partial result. This way no private information is freed to an agent active in the domain and no direct communication is necessary, however one needs to assure the existence of a trustworthy external agent.

Phase 3: Post-processing If in one agent an edge is found during the previous phases and in another it is not, then this edge is removed, as stated by Theorem 8.5.

Example

Here we will discuss an example run of our multi-agent learning algorithm. Consider the network in Figure 8.8. We assume that the samples from this network are stored at two different sites S_1 and S_2 , containing

$$\mathbf{V}_{M_1} = \{X, X1, X2, X3, X4, X5, X6, X7\}$$

and

$$\mathbf{V}_{M_2} = \{X4, X5, X6, X7, X8, X9, Y\}$$

respectively.

Applying *M-A CaDo* to this model gives:

Phase 1:

The locally correct models are given in Figure 8.9.

As you can see there are some edges that do not appear in the global model, $X6 - X7$ and $X4 - X8$.

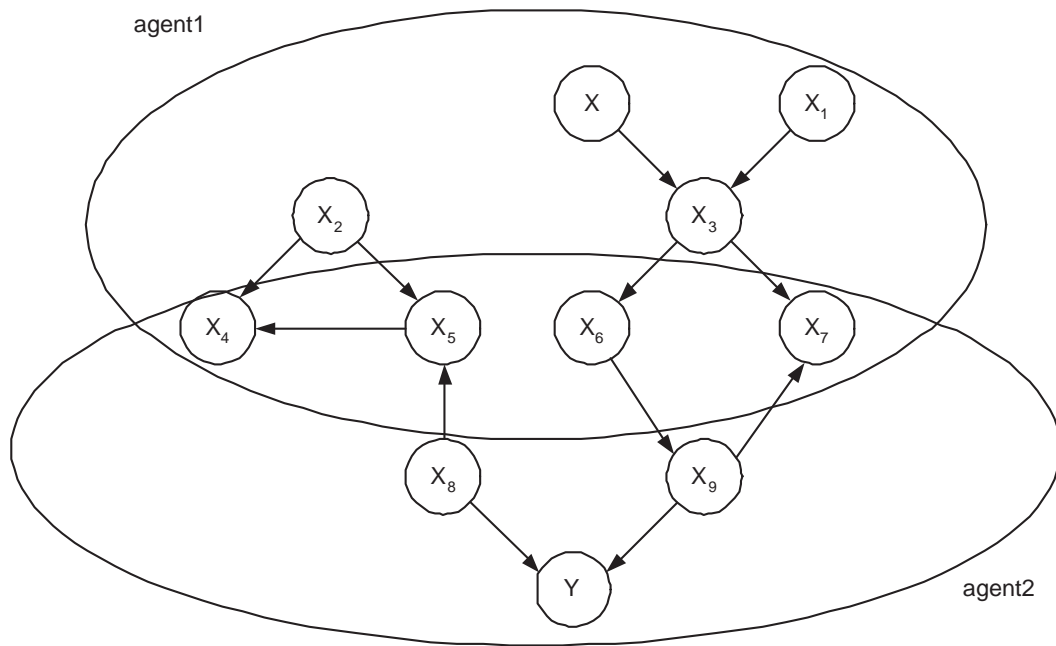


Figure 8.8. Global model and its subdivision over the different sites.

Phase 2:

Agent1 demands the samples of the neighbors of \mathbf{V}_{M_1, M_2} in *agent2* and receives the samples from X_8 and X_9 . Using standard independence tests, *agent1* finds $(X_6 \perp\!\!\!\perp X_7 | X_3, X_9)$.

Agent2 asks for the neighbors of \mathbf{V}_{M_1, M_2} in *agent1* and receives the samples from X_2 and X_3 and finds that $(X_6 \perp\!\!\!\perp X_7 | X_3, X_9)$ and $(X_8 \perp\!\!\!\perp X_4 | X_5, X_2)$.

Phase 3:

Remove the edges $X_6 - X_7$ and $X_4 - X_8$ in the substructures.

The algorithm retrieves the correct structure as shown in Figure 8.8.

8.2.5 Critical Discussion of M-A CaDo

In this section we have developed an approach for learning MACM from distributed data of two sites, where only samples concerning the neighbors of the intersection have to be disclosed to the other agent.

A possible extension would be to study the possibility of not disclosing all the neighbors of the intersection, but only subsets of that.

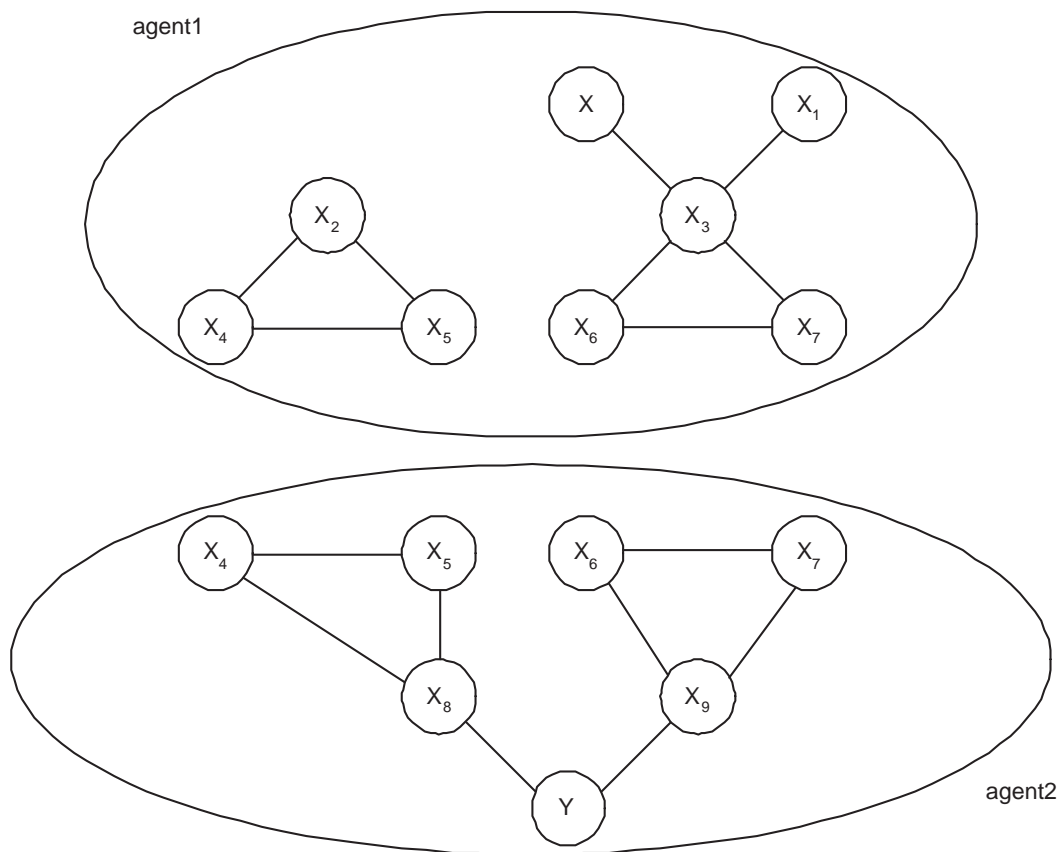


Figure 8.9. The correct local models from the global structure in Figure 8.8.

Furthermore, instead of disclosing all the samples, only a statistically sufficient set could be disclosed to the other agent. Finally, extending this approach to more agents, would use the same principle, but would only make the disclosure of information more intricate.

If experiments are allowed and can be performed perfectly by each agent, then we can treat the variables in other agents as latent variables. This allows us to use MyCaDo++ to learn the local parts of the global model. Assuming that there are no latent variables in the global model we can then just paste the local models together while removing the bi-directed edges found using MyCaDo++ taking into account the properties described earlier.

8.3 Validation of Assumptions

In this section we will introduce some procedures to check whether a given MACM is valid, i.e. whether it exhibits all the properties mentioned in section 7.1.2. This is useful when a MACM is not learned, but derived from experts.

8.3.1 Intersection

First of all, the agents check whether they model the intersections they share in the same way.

This is done by each agent sending the variables, structure and distribution of the intersection to the other agent. If the agents do not agree about the intersection, the model is not a valid MACM and it cannot be used to perform inference.

8.3.2 Acyclicity

Secondly, a check is enforced, testing whether combining models from different agents does not introduce cycles going through their intersection.

For this task we use an adaptation of Xiang's algorithm for the distributed verification of acyclicity (Xiang, 2002). It is based on the alternate removing of private roots and leaves and public roots and leaves from the model. If at the end of such a process all the nodes are marked, the model is acyclic, otherwise it is cyclic.

We will not give a full description of the algorithm here, instead we give an illustrative example run. For the full details we refer to Chapter 9 in (Xiang, 2002).

In Figure 8.10(a), we see a MACM consisting of three agents of which we want to check whether the underlying associated SMCM is acyclic.

In (b), each agent locally marks the private roots and leaves, they are depicted by surrounding them with a dotted bold circle.

In (c), *agent3* sends to *agent2* that X_{10} has no unmarked children or parents in its model, then together with its local knowledge *agent2* can mark node X_{10} as it is a root node. After that *agent2* can also mark X_{12} as it has become a root node.

In (d), *agent3* sends to *agent2* that X_{11} has no unmarked children or parents in its model, then together with its local knowledge *agent2* can mark node X_{11} as it is a leaf node. As a consequence *agent2* can mark node X_{14} , as it has become a leaf node.

In (e), *agent2* sends to *agent1* that X_2 has no unmarked children or parents in its model, then together with its local knowledge *agent1* can mark node X_2 as it is a leaf node. Due to this node X_4 becomes a leaf and can be marked, and finally this makes node X_3 a leaf node and it can be marked.

In (f), *agent2* sends to *agent1* that X_1 has no unmarked children or parents in its model, then together with its local knowledge *agent1* can mark node X_1 as it is a leaf node (and a root node).

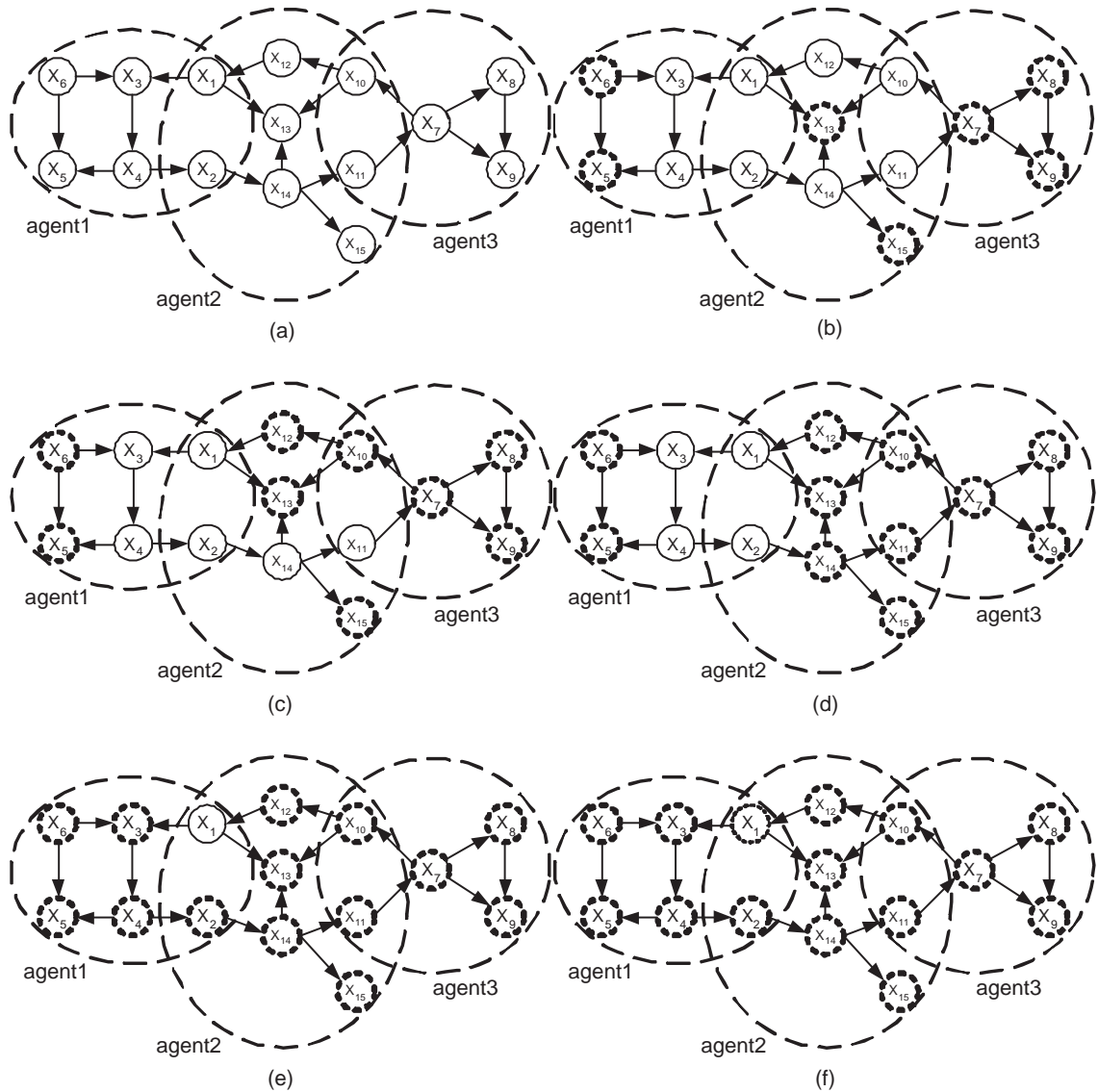


Figure 8.10. (a) A MACM (b)-(f) Different steps in checking whether the MACM in (a) is acyclic.

8.4 Overview of the Chapter

In this chapter we presented our contributions for Multi-Agent Causal Models.

We introduced a generic way to split up a SMCM into its minimal subparts in order to obtain a MACM. We provided an alternative inference algorithm to calculate the causal effect $P(v|do(x_i))$ in a MACM in which the agents are linked in a complex manner.

We introduced an algorithm for learning the structure of MACMs. In a first phase we proved that if we are not concerned with the amount of information being communicated or with privacy issues, it is possible to learn the structure of a MACM. We showed that there

are a lot of non-trivial problems that need to be solved in order to learn a globally consistent network and presented a few starting points to circumvent these.

Conclusion

In this chapter we recapitulate the main results of the research and discuss several possible applications of our results. We end with some directions for future research.

9.1 Results

In this dissertation we have made contributions on modeling causal relationships using three types of models with differing strengths of representational power: causal Bayesian networks (Chapter 4), causal latent models (Chapter 6) and multi-agent causal models (Chapter 8).

9.1.1 Causal Bayesian Networks

Causal Bayesian networks (CBNs) are an extension of Bayesian networks in which directed edges represent direct causal relationships. The semantics of the edges in the network allow us to calculate the effect of interventions on some variables in the system being modeled. However, these semantics also render learning the structure of a causal Bayesian network more difficult than just learning the dependence structure as is done for Bayesian networks.

In this dissertation we focused on finding algorithms whose goal is to give an experimentation plan that when executed (a) identifies the entire causal structure, (b) identifies it with the least cost/number of experiments possible.

The motivation for using experiments stems from the fact that in general we can not retrieve the entire causal structure from observational data alone. Causal discovery from observations is limited to retrieving the Markov equivalence class of the underlying correct causal graph.

We discussed two different settings in this dissertation. First, we studied the case where the available observational data is informative enough to retrieve all conditional independence information. In a second algorithm we remove this assumption and allow for uncertain results of (in)dependence.

Learning with perfect observational data

Assuming perfect observational data allows us to use traditional causal structure learning techniques such as PC introduced by (Spirtes et al., 2000a). Since PC has been proved to be asymptotically correct, we can use the correct CPDAG, representing the equivalence class of the correct network, as a starting point. Experiments are then used in a second phase to resolve the undirected substructures in the CPDAG. We presented the MyCaDo (Section 4.1) algorithm which provides an experimentation plan to complete the network.

In this dissertation we used structural experiments to discover causal links. Structural experiments cut off all information from the parents of the intervened variable. In the worst case scenario, which occurs when there is a complete undirected substructure of N variables, $N - 1$ experiments are needed to complete the structure. In the general case we try to minimize the total number of experiments because in practice experiments and measurements have a cost (sometimes they are even impossible to perform).

We use elements from decision theory to create an experimentation plan that is (a) guaranteed to complete the structure if all experiments and measurements are possible and (b) minimizes the total cost of the experiments based on some decision criteria.

Learning with imperfect observational data

When we do not have perfect observational data, it is not safe to assume that the PC algorithm will output the correct CPDAG. Therefore, we allow for some type of uncertainty during the initial learning phase. In order to accomplish this we have proposed an adapted unsure independence test. This test has three possible results; true (independence), false (independence) and unsure (not enough information available).

We proposed an algorithm called UnCaDo (Section 4.2) which in a first phase uses the results found by the adapted independence test to construct an initial unsure skeleton.

The second phase of UnCaDo consists again of performing structural experiments. However this time we start by looking at all unsure edges in the unsure skeleton and use experiments to obtain more data so we can identify the nature of the relationship. When all unsure edges are resolved and hence oriented, we perform the normal orientation rules of the PC algorithm while regarding the resolved unsure edges as undirected edges. After this step we continue as if we were working with perfect data, since we are now guaranteed that the given partially directed dag is correct.

9.1.2 Causal Latent Networks

Causal Bayesian networks lack the representational power to model settings in which there are latent variables, which are unmeasured variables that influence the system. It is realistic

to assume that in many studies not all possible influencing variables are included. Two major causal graphical techniques have been proposed that model latent variables implicitly: semi-Markovian causal models and ancestral graphs.

A complete causal inference algorithm has been constructed for semi-Markovian causal models. However, no structure/parameter learning algorithm for this model had been proposed. Interestingly, no inference algorithm had been proposed for ancestral graphs, but algorithms exist to learn the structure of an ancestral graph up to its Markov equivalence class. This motivated us to combine existing methods from both paradigms to construct an integral modeling approach for causal relationships in a setting with latent variables.

Structure Learning of Causal Latent Networks

Learning the structure of a causal graph when there are latent variables has been mostly confined to ancestral graphs. The FCI algorithm Spirtes et al. (2000a) together with the extra orientation rules proposed in Zhang and Spirtes (2005b) are proved to give the correct CPAG representing the equivalence class of the correct ancestral graph given perfect observational data.

As is the case for CBN, in general, the structure will not be completely directed and experiments are needed to completely orient the structure. However, the semantical interpretation of edges in an ancestral graph, meaning a directed edge represents an ancestral not a direct causal relationship makes this process more difficult. However, by using experiments it is possible to identify which directed edges represent direct causes and which edges are a result of so called inducing paths.

In a SMCM all directed edges represent direct causal relationships. This motivated us to propose a transformation, which changes an ancestral graph into the SMCM representing the same causal relationships. Furthermore, by using experiments we are able to completely orient the causal latent network. We proposed an algorithm called MyCaDo++ (Section 6.1.2) which identifies the set of experiments that need to be performed to gather the right data to perform this transformation.

Inference in Causal Latent Networks

The state of the art causal inference algorithm for causal latent networks has been developed by (Tian, 2002) and works on SMCMs. Causal inference is also possible in AGs but only in those parts common to all graphs in the same Markov equivalence class. Given our transformation, it is possible to perform causal inference indirectly by using the respective SMCM that corresponds to the given AG.

Probabilistic inference in causal latent models has not been studied widely. We propose a different representation called the PR-representation (Section 6.2) of SMCs that allows for probabilistic inference. This PR-representation is formed by replacing confounding (bi-directed) edges, representing latent variables, by a set of directed edges. The PR-representation generated this way is an I-map of the underlying distribution and hence probabilistic inference can be performed therein.

9.1.3 Multi-Agent Causal Models

Multi-Agent Causal Models were proposed to extend existing causal modeling techniques in order to be able to model domains that become evermore complex. Instead of making a centralized model containing all the variables of the system, there are several agents that each model an overlapping part of the domain.

A causal inference algorithm has been proposed for this setting in case the agents are connected in a tree structure. In this dissertation we proposed the first steps of a structure learning algorithm and proposed a simpler, though less powerful, causal inference algorithm, for which agents can be arbitrarily connected.

Multi-Agent Structure Learning

The goal of learning the structure of a MACM is twofold: (a) the individual agents must be able to use the learned local model to resolve local queries, (b) the global model must be able to answer global queries. It is however not so that the global model can just be obtained by stitching the local models together. This is because the local models will contain false dependencies between variables due to the fact that the variables that make them independent in the global model are modeled in another agent.

In order to learn a correct global model, agents must communicate information on certain variables with each other. More specifically some local variables have to be disclosed to other agents.

In this dissertation we have developed an algorithm called M-A CaDo (Section 8.2) that transforms the results of a single agent learning algorithm performed on each agents' local variables into a valid multi-agent causal model.

Multi-Agent Causal Inference

The basic algorithm for multi-agent causal inference has been proposed by Sam Maes in his dissertation Maes (2005). Therein, he describes the complete inference task which consists of three different phases: identifiability, negotiation and identification.

A causal effect $P(y|do(x))$ is identifiable from a causal model if it can be calculated from that model. The identifiability phase consists of checking two properties: (a) would the causal effect be identifiable if the global model was modeled by a single agent, (b) is the causal effect identifiable given the structural decomposition of the model into several agents.

In the case that the causal effect is not identifiable because of the decomposition into agents, a negotiation phase is needed in which agents exchange information with each other in order to make the causal effect identifiable. The proposed negotiation algorithm is performed in such a way that the agents try to find a fair trade-off for the amount of variables that they have to disclose.

The actual calculation of the causal effect is called identification. A single agent causal inference algorithm has been extended for calculating effects from a multi-agent causal model when the local models of the agents are linked in a chain.

In this dissertation, see Section 8.1, we adapted and simplified the causal inference algorithm in that it is only possible to calculate the effect of performing an experiment on all other variables in the model. However, agents can be arbitrarily linked and they do not need to disclose variables to each other.

The algorithm described in this dissertation is based on actually decomposing a semi-Markovian causal model into its basic blocks, the c-components. Each agent will host the variables of a c-component and the parents of these variables. This assures that calculations for the identification of causal effects can be performed locally. Instead of exchanging variables, the agents disclose local joint probabilities for a given instantiation and for a given experiment to each other.

9.2 Possible Applications

In this section we discuss some possible applications of the research proposed in this dissertation.

There are several applications for our techniques in *bio-informatics*. Gene regulatory networks are networks that show how the expression levels of genes are related to each other Bower and Bolouri (2001). For instance, over expression of one gene can induce over expression of another gene or diminish the expression of again another gene. Learning the network of interacting genes is a difficult task since typically there are many more genes than there are samples. Furthermore, the expression level of genes is influenced by a number of unmeasured variables.

The techniques proposed in this dissertation can help to (a) model these complex networks Yu et al. (2004), (b) learn them partially from observational data Spirtes et al. (2000b); Badae (2003), (c) give directed targets to perform experiments on to complete the network Didelez and Sheehan (2007), (d) incorporate unmeasured variables in the model and (e) predict the effect of some experiment without having to actually perform it.

Furthermore, if certain conditions are met, i.e. overlapping data sets and no unmeasured confounders between two studies, then results from different research groups on subsets of genes can be combined by using MACM.

In general, the techniques described in this dissertation are particularly useful for *modeling complex domains* since they can incorporate unknown variables in the form of latent causal modeling by SMCMs and can handle problem domains are separated into several modular parts or that are inherently distributed. This versatility is the reason why causal models are used in many domains such as physics, social sciences and biomedical sciences Russo and Williamson (2007).

Causal inference can be useful as a component in a *decision support* system. A decision support system is an automated system that assists humans in making complex decisions. For instance, if a company wants to enforce a new policy, it is possible to model the influenced variables into a causal network and then calculate the expected effect of this policy without having to enforce it. This way a company can decide based on the model which policy is the most interesting.

9.3 Directions for Future Research

In this section we discuss some possible pathways for future research.

Learning Causal Latent Networks with Minimal Cost

We have discussed learning the structure of a CBN with a minimal amount of experiments and minimal overall cost (Section 4.1). We would like to extend this approach to causal latent networks. The plan would be to use the orientation rules proposed in Zhang and Spirtes (2005b) as an intermediate phase between experiments. This will possibly allow the discovery of an additional set of directions and hence remove the need for some experiments.

Learning Causal Latent Networks from Imperfect Data

In settings with latent variables it is also realistic to assume that the provided data is not perfect and that hence faults can infiltrate in the results of FCI. We would like to extend the

approach used for CBN in this case (Section 4.2) to causal latent networks. This way we can provide a practical useful experiment scheme to discover a SMCM.

Dynamic Models

As was discussed in the introduction of this dissertation, we did not take time into account for any of our techniques. It is however an interesting research area to study the influence of the explicit inclusion of time in some of our algorithms. For instance, we want to look whether our results can be extended to dynamic Bayesian networks or to other continuous time models.

References

- Ali, Ayesha R., Thomas Richardson, Peter Spirtes, and J. Zhang. 2005. Orientation rules for constructing markov equivalence classes of maximal ancestral graphs. Technical Report 476, Dept. of Statistics, University of Washington.
- Badea, L. 2003. Inferring large gene networks from microarray data: a constraint-based approach. In Proceedings of the Workshop on Learning Graphical Models for Computational Genomics.
- Bower, J.M., and H. Bolouri, ed. 2001. Computational modeling of genetic and biochemical networks. MIT Press.
- Chen, R., K. Sivakumar, and H. Kargupta. 2003. Learning bayesian network structure from distributed data. In Proceedings of the 3rd SIAM International Data Mining Conference, 284–288.
- Chen, R., K. Sivakumar, and H. Kargupta. 2004. Collective mining of bayesian networks from distributed heterogeneous data. Knowledge and Information Systems 6:164–187.
- Cheng, J., D.A. Bell, and W. Liu. 1997. Learning belief networks from data: An information theory based approach. In Proceedings of the sixth ACM International Conference on Information and Knowledge Management CIKM, 325–331.
- Chickering, D.M. 2002a. Learning equivalence classes of bayesian-network structures. The Journal of Machine Learning Research 2:445–498.

- Chickering, D.M. 2002b. Optimal structure identification with greedy search. Journal of Machine Learning Research 3:507–554.
- Cooper, G., and E. Herskovits. 1992. A bayesian method for the induction of probabilistic networks from data. Machine Learning 9:309–347.
- Cooper, G.F., and C. Yoo. 1999. Causal discovery from a mixture of experimental and observational data. In Proceedings of Uncertainty in Artificial Intelligence, 116–125.
- Crestani, F., and C. J. Van Rijsbergen. 1994. Probability kinematics in information retrieval: a case study. Dept. of CS, Univ. of .
- Didelez, V., and N.A. Sheehan. 2007. Mendelian randomisation as an instrumental variable approach to causal inference. Statistical Methods in Medical Research 16:309–330.
- Dubois, D. 2006. Possibility theory and statistical reasoning. Comput. Stat. Data Anal. .
- Eaton, D., and K. Murphy. 2007. Exact bayesian structure learning from uncertain interventions. AI & Statistics 107–114.
- Eberhardt, F., C. Glymour, and R. Scheines. 2005a. N-1 experiments suffice to determine the causal relations among n variables. Technical report, Carnegie Mellon University.
- Eberhardt, F., C. Glymour, and R. Scheines. 2005b. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. In Uncertainty in Artificial Intelligence Conference (UAI), 178–184.
- Eberhardt, F., and R. Scheines. 2006. Interventions and causal inference. In Philosophy of Science Assoc. 20th Biennial Mtg, 981–995.
- Friedman, N., and D. Koller. 2000. Being bayesian about network structure. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000), 201–210.
- Haavelmo, T. 1995. The statistical implications of a system of simultaneous equations. The Foundations of Econometric Analysis 477–490.

- Heckerman, D. 1995. A tutorial on learning with bayesian networks. Technical report, Microsoft Research.
- Heckerman, D., D. Geiger, and M. Chickering. 1995. Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning 20:197–243.
- Ide, J.S., F.G. Cozman, and F.T. Ramos. 2004. Generating random bayesian networks with constraints on induced width. In European Conference on Artificial Intelligence (ECAI), 323–327.
- J-P. Pellet, A. Elisseeff. 2008. Using markov blankets for causal structure learning. JMLR 1295–1342.
- Jensen, F.V. 1996. Bayesian updating in causal probabilistic networks by local computations. Computational Statistics Quarterly 4:269–282.
- Kindermann, R., and J.L. Snell. 1980. Markov random fields and their applications. American Mathematical Society.
- Koivisto, M. 2006. Advances in exact bayesian structure discovery in bayesian networks. In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006), 241–248.
- Koivisto, M., and K. Sood. 2004. Exact bayesian structure discovery in bayesian networks. Journal of Machine Learning 5:549–573.
- Lam, W., and F. Bacchus. 1994. Learning bayesian belief networks: An approach based on the mdl principle. Computational Intelligence 10:269–293.
- Lewis, D. 1981. Probability of conditionals and conditionals probabilities, 129–147. The University of Western Ontario Series in Philosophy of Science. D.Reidel Publishing Company.
- Maes, S. 2005. Multi-agent causal models: Inference and learning. Doctoral Dissertation, Vrije Universiteit Brussel.

- Maes, S., S. Meganck, and P. Leray. 2007a. An integral approach to causal inference with latent variables. In Causality and Probability in the Sciences (Texts in Philosophy), 17–41. College Publications.
- Maes, S., S. Meganck, and B. Manderick. 2004. Multi-agent identification of causal effects. In Proceedings of the 2004 European Workshop on Multi-Agent Systems (EUMAS), ed. Chiara Ghidini, Paolo Giorgini, and Wiebe van der Hoek, 409–417.
- Maes, S., S. Meganck, and B. Manderick. 2005a. Causal inference in multi-agent causal models. In Proceedings of Modèles Graphiques Probabilistes pour la Modélisation des Connaissances, Atelier of EGC 05, 53–62.
- Maes, S., S. Meganck, and B. Manderick. 2005b. Identification in chain multi-agent causal models. Proceedings of the Special Track on Uncertain Reasoning of FLAIRS 2005.
- Maes, S., S. Meganck, and B. Manderick. 2007b. Inference in multi-agent causal models. Int. J. Approx. Reasoning 46:274–299.
- Maes, S., J. Reumers, and B. Manderick. 2003. Identifiability of causal effects in a multi-agent causal model. In Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT), 605–608.
- Meek, C. 1995. Causal inference and causal explanation with background knowledge. In Proceedings of 11th Conference on Uncertainty in Artificial Intelligence, 403–418.
- Meganck, S., P. Leray, and B. Manderick. 2006a. Learning causal bayesian networks from observations and experiments: A decision theoretic approach. In Proceedings of Modelling Decisions in Artificial Intelligence, MDAI 2006, 58–69.
- Meganck, S., P. Leray, and B. Manderick. 2007. Causal graphical models with latent variables: Learning and inference. In Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2007, 5–16.
- Meganck, S., P. Leray, and B. Manderick. 2008a. Separating semi-markovian causal models. Technical report, Vrije Universiteit Brussel.

- Meganck, S., P. Leray, and B. Manderick. 2008b. Uncado: Unsure causal discovery. In Proceedings of 4emes journees francophones de reseaux bayesiens JFRB, 5–16.
- Meganck, S., S. Maes, P. Leray, and B. Manderick. 2005a. Distributed learning of multi-agent causal models. In Proceedings of 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05), 285–288.
- Meganck, S., S. Maes, P. Leray, and B. Manderick. 2005b. A learning algorithm for multi-agent causal models. In The Third European Workshop on Multi-Agent Systems (EUMAS 2005). Electronical publication.
- Meganck, S., S. Maes, P. Leray, and B. Manderick. 2006b. Learning semi-markovian causal models using experiments. In Proceedings of The third European Workshop on Probabilistic Graphical Models , PGM 06., 195–206.
- Murphy, K.P. 2001. Active learning of causal bayes net structure. Technical report, Department of Computer Science, UC Berkeley.
- Nadkarni, S., and P.P. Shenoy. 2001. A bayesian network approach to making inferences in causal maps. European Journal of Operational Research 128:479–498.
- Neapolitan, R.E. 2003. Learning bayesian networks. Prentice Hall.
- Pearl, J. 1988. Probabilistic reasoning in intelligent systems. Morgan Kaufmann.
- Pearl, J. 2000. Causality: Models, reasoning and inference. MIT Press.
- Pena, J.M., R. Nilsson, J. BJORKEGREN, and Jesper Tegner. 2007. Towards scalable and data efficient learning of markov boundaries. IJAR 211–232.
- Rabiner, L.R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In IEEE 77(2), 257–286.
- Richardson, T., and P. Spirtes. 2002. Ancestral graph markov models. Annals of Statistics 30:962–1030.

- Richardson, T., and P. Spirtes. 2003. Causal inference via ancestral graph models, chapter 3. Oxford Statistical Science Series: Highly Structured Stochastic Systems. Oxford University Press.
- Robinson, R. W. 1977. Counting unlabeled acyclic digraphs. Combinatorial Mathematics 622:28–43.
- Rosenbaum, P.R., and D.B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. Biometrika 70:41–55.
- Russell, S.J., and P. Norvig, ed. 1995. Artificial intelligence: A modern approach. Prentice Hall.
- Russo, F., and J. Williamson, ed. 2007. Causality and probability in the sciences. College Publications.
- Salmon, W. C. 1998. Causality and explanation. Oxford University Press.
- Schwartz, G. 1978. Estimating the dimension of a model. Annals of statistics 6(2):461–464.
- Shipley, B. 2000. Cause and correlation in biology. Cambridge University Press.
- Shpitser, I., and J. Pearl. 2008. Dormant independence. In Twenty-Third Conference on Artificial Intelligence, 1081–1087.
- Spirtes, P., C. Glymour, and R. Scheines. 2000a. Causation, prediction and search. MIT Press.
- Spirtes, P., C. Glymour, R. Scheines, S. Kauffman, V. Aimale, and F. Wimberly. 2000b. Constructing bayesian network models of gene expression networks from microarray data. In Proceedings of the Atlantic Symposium on Computational Biology.
- Spirtes, P., C. Meek, and T. Richardson. 1999. An algorithm for causal inference in the presence of latent variables and selection bias. In Computation, causation, and discovery, 211–252. Menlo Park, CA: AAAI Press.

- Tian, J. 2002. Studies in causal reasoning and learning. Ph.d. thesis, UCLA Computer Science Department.
- Tian, J. 2005. Generating markov equivalent maximal ancestral graphs by single edge replacement. In Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI), 591–598.
- Tian, J., and J. Pearl. 2002a. A general identification condition for causal effects. In Proceedings of the National Conference on Artificial Intelligence (AAAI), 567–573.
- Tian, J., and J. Pearl. 2002b. On the identification of causal effects. Technical Report (R-290-L), UCLA C.S. Lab.
- Tian, J., and J. Pearl. 2002c. On the testable implications of causal models with hidden variables. In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI), 519–527.
- Tong, S., and D. Koller. 2001. Active learning for structure in bayesian networks. In Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), 863–869.
- Verma, T., and Judea Pearl. 1990. Equivalence and synthesis of causal models. In Proceedings of the 6th Conference in Uncertainty in Artificial Intelligence, 220–227.
- Wald, A. 1945. Sequential tests of statistical hypotheses. The Annals of Mathematical Statistics 16:117186.
- Xiang, Y. 2002. Probabilistic reasoning in multiagent systems: A graphical models approach. Cambridge University Press.
- Yu, J., V. Smith, P. Wang, A. Hartemink, and E. Jarvis. 2004. Advances to bayesian network inference for generating causal networks from observational biological data. Bioinformatics .
- Zhang, J., and P. Spirtes. 2005a. A characterization of markov equivalence classes for ancestral graphical models. Technical Report 168, Dept. of Philosophy, Carnegie-Mellon University.

Zhang, J., and P. Spirtes. 2005b. A transformational characterization of markov equivalence for directed acyclic graphs with latent variables. In Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI), 667–674.

Zhang, Jiji. 2006. Causal inference and reasoning in causally insufficient systems. Doctoral Dissertation, Carnegie Mellon University.