

Institut für Informatik
der Technischen Universität München

Constraint-Based Structural Learning in Bayesian Networks using Finite Data Sets

Harald Steck

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. H. M. Gerndt

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Dr. h.c. W. Brauer
2. Prof. Dr. S. L. Lauritzen, Aalborg University, Dänemark

Die Dissertation wurde am 16. Januar 2001 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08. Mai 2001 angenommen.

Acknowledgments

The years at Siemens Corporate Technology have been a valuable experience. I would like to thank Prof. Bernd Schürmann, head of the Department for Neural Computations, for providing a working environment where academics and industry meet.

I am very grateful to Prof. Wilfried Brauer at the Department of Computer Science, Technical University of Munich, not only for taking over the duty of supervising my thesis work, but also for providing me with valuable comments and suggestions. Apart from that, I would like to thank Prof. Wilfried Brauer for all his support and encouragement. I have appreciated very much that I could always bother him with questions, even when he was traveling.

I am deeply indebted to Prof. Steffen Lauritzen at the Department of Mathematical Sciences, Aalborg University, Denmark, for his spontaneous willingness to be the co-reviewer of my thesis despite all the troubles involved, like, for instance, traveling to Munich because of my defense. Moreover, I am grateful that I could visit Prof. Steffen Lauritzen for two weeks during my thesis work. The enlightening discussions have had a major impact on this thesis.

Moreover, I would like to thank Dr. Volker Tresp for supervising my work at Siemens Corporate Technology. My thesis owes enormously to his influence: he introduced me to the areas of graphical models and machine learning, and over the years I have benefited considerably from his broad knowledge in numerous discussions.

I would also like to thank Dr. Michael Haft and Dr. Reimar Hofmann, also working on graphical models, machine learning and data mining, for valuable debates; Alexander Hall for providing me with very helpful comments; Dr. Peter Mayer for interesting discussions on statistics; Dr. Paul Theo Pilgram for patiently answering all my questions concerning the English language; Marco Pellegrino and Dr. Kai Heesche for helping me out whenever I had a question regarding the local computer cluster at Siemens; Dr. Martin Appl for help and support; and all the members of the Department for Neural Computations at Siemens. Furthermore, I would like to thank Dr. Paul Theo Pilgram, Dr. Kai Heesche and Dr. Henning Lenz for a fantastic atmosphere in the office we shared.

I am also grateful to Siemens AG for financial support of my work and of my journeys to various conferences and workshops, the most distant one taking place at Stanford University, California.

I am indebted to Prof. Iris Pigeot-Kübler, Dr. Angelika Caputo, and Vanessa Didelez for many interesting discussions and for the opportunity to participate in their seminar on graphical models at the Department of Statistics, Ludwig Maximilians University, Munich, in the winter semester 1999 / 2000.

Finally, I would like to thank my parents and my sister, having supported and encouraged me for over twenty years of education and training.

Summary

Bayesian networks have become a popular probabilistic model for coping with uncertainty. Structural learning in Bayesian networks is an NP-hard problem, which suggests the use of heuristic strategies for finding close-to-optimum solutions. The constraint-based approach has proven to be very efficient in many experiments [23, 24, 142]. However, it is only well-understood under certain conditions, typically when infinite data sets are provided.

This thesis focuses on the constraint-based approach for those cases where only a *finite* amount of data is available, as typical in practical applications. The usual assumptions underlying the constraint-based approach are thus dropped, and we look at the constraint-based approach as a particular search strategy towards the optimal Bayesian network structure, with respect to a given scoring function. This point of view is independent of the choice of scoring function: various Bayesian and non-Bayesian scoring functions can be used. We introduce so-called *relative scoring functions* and derive their main properties. They help in understanding the constraint-based approach for finite data.

We propose an extension of the constraint-based approach, which can essentially be divided into two parts. The first one is concerned with the presence of edges, and hence employs an undirected graph, while the second step determines the orientations of the edges. Regarding the first step, we derive the so-called *necessary path condition* from properties of optimal Bayesian network structures. While its use does not notably increase computation time, it entails considerable improvements in the quality of the induced network structures. *Model uncertainty* is an important aspect in learning of Bayesian networks from finite data. In particular, uncertainty regarding the presence of edges can be discovered by this extension: instead of a single network structure, as typical for state-of-the-art constraint-based approaches, it can find several graphs. Of course, this is only possible up to a degree, as the exact learning problem is NP-hard. The induced graphs usually have many common edges and differ only in the presence of a few edges. All the different structures can hence be visualized by means of a *single* graph, thus easing the interpretation [144].

Next, we present an extension of the second part of the algorithm, namely an operator which determines the orientations of the edges, based on a scoring function [143]. This operator proved quite robust when given finite data sets in our experiments, while standard schemes, which rely on the induced conditional independences, are known to be rather unstable [142].

We argue that the Bayesian network structures found by constraint-based approaches generally tend to contain fewer edges than optimal structures. Hence, a post-processing step is necessary if one aims at finding, possibly different, local optima. In case that the assumptions underlying state-of-the-art constraint-based approaches hold, the proposed extensions yield the correct Bayesian network structure. The proposed extensions of the constraint-based algorithm were evaluated on artificial and real-world data.

Our extended constraint-based approach has in parts entered the *Esprit* project called *PRONEL* [120] where it has become the core learning algorithm. A demo version is available from *Hugin Expert A/S* via <http://www.hugin.com/pronel/index.html>.

Contents

1	Introduction	1
1.1	Structural Learning in Bayesian Networks	3
1.2	The Constraint-Based Approach applied to Finite Data Sets	4
1.3	Thesis Overview	6
2	Bayesian Networks	9
2.1	Conditional Independence and Dependence	9
2.2	Bayesian Networks and their Properties	10
2.2.1	Directed Acyclic Graph (DAG)	10
2.2.2	Recursive Factorization of the Probability Distribution	12
2.2.3	Markov Equivalence	13
2.2.4	Perfect Map and Faithfulness	14
2.3	Graphical Models	16
2.4	Some Related Tools for Data Analysis	17
3	Structural Learning	19
3.1	Scoring Functions	19
3.1.1	Decomposability	20
3.1.2	Relative Scoring Functions	20
3.1.3	Score Equivalence	23
3.1.4	Popular Scoring Functions	25
3.2	Model Uncertainty and Model Averaging	33

3.3	Algorithms for Structural Learning	35
3.3.1	Optimizing the Scoring Function	36
3.3.2	Constraint-Based Approach	38
4	Properties of Optimal Structures	43
4.1	Motivation	44
4.2	Properties of Optimal DAGs	47
4.3	Necessary Conditions concerning Skeletons	53
4.3.1	Necessary Path Condition	54
4.3.2	Extended Necessary Path Condition	60
4.3.3	Simplified Necessary Path Conditions	65
4.3.4	Minimal Skeletons	66
4.3.5	Parsimony	68
4.4	Properties of the Perfect Map	70
5	Learning the Presence of Edges	73
5.1	Rules	73
5.2	Simplifying the Rules	75
5.2.1	Removing the Conditions Apparently Fulfilled	75
5.2.2	Condition Graph	80
5.2.3	Reducing the Number of Rules	84
5.3	Ambiguous Regions and Summary Graph	85
5.3.1	Definitions	87
5.3.2	Search within each Ambiguous Region	88
5.4	Computing the Scores	92
5.4.1	Representation of the Data	92
5.4.2	Indirect Evaluation	93
5.4.3	Reducing the Number of Computed Scores	95
5.4.4	Parallel Computing	97

5.4.5	Positive Threshold Values	97
5.5	Efficiency of the Algorithm	98
5.6	Experiments	100
5.6.1	A Simplistic Example	100
5.6.2	The Alarm Network	103
5.7	Discussion	117
5.8	Incorporating Prior Knowledge	118
5.9	Incomplete Data	120
6	Inducing DAGs given Skeletons	123
6.1	Orienting the Edges	123
6.1.1	Scores of Collider Candidates	124
6.1.2	Orienting the Colliders	126
6.1.3	Orienting the Remaining Edges	126
6.1.4	Experiments with the Operator	129
6.1.5	Discussion	132
6.2	Finding Optimal DAGs	134
6.3	Experiments	135
6.3.1	The Alarm Network	136
6.3.2	Real-World Data	138
6.3.3	Summary	150
	Conclusions	151
	Appendix	155
	A Cross Validation	155
	B Data Sets	157
	Bibliography	161

Key to Important Symbols

This list depicts the most important symbols and abbreviations used throughout this thesis, and it gives a pointer to the section where the notation is introduced.

Variables

a, b, v, x_i, \dots	variables	cf. Section 2.1
\mathcal{S}, \dots	set of variables	cf. Section 2.1
$ \mathcal{S} $	number of variables in \mathcal{S} , also called <i>order</i> of the set \mathcal{S}	cf. Section 2.2.2
\mathcal{V}	set of <i>all</i> variables in a domain	cf. Section 2.2
n	number of variables in a domain, i.e. $n = \mathcal{V} $	cf. Section 3.3.2
$\mathcal{I}(a)$	set of all states of variable a	cf. Section 2.1
$\mathcal{I}(\mathcal{S})$	set of all joint states of the variables in \mathcal{S}	cf. Section 2.1
$i, j, \dots \in \mathcal{I}(\mathcal{S})$	a joint state of the variables in \mathcal{S}	cf. Section 2.1
$ \mathcal{I}(\mathcal{S}) $	number of joint states of the variables in \mathcal{S}	cf. Section 3.1.4

Bayesian Networks, Graphs and Parameters

θ	parameters of a Bayesian network	cf. Section 2.2.2
m	directed acyclic graph (DAG), or partially directed acyclic graph (PDAG)	cf. Section 2.2.1 cf. Section 6.1.1
\tilde{m}	skeleton, i.e. undirected graph associated with a DAG	cf. Section 2.2.1
\mathcal{M}	set of all DAGs	cf. Section 3.1.2
$a \sim b$	(undirected) edge between the variables a and b	cf. Section 2.2.1
$a \rightarrow b, b \leftarrow a$	directed edge from variable a to variable b , or edge with a "proposed" direction	cf. Section 2.2.1 cf. Section 6.1.3
$a \Rightarrow b, b \Leftarrow a$	edge with a "fixed" orientation	cf. Section 6.1.3
$a \leftrightarrow b$	edge oriented in "both" directions	cf. Section 6.1.3
$\text{pa}_m(a), \text{pa}(a)$	parents of a variable $a \in \mathcal{V}$ in a DAG m	cf. Section 2.2.1
$\text{an}(a)$	ancestors of a variable $a \in \mathcal{V}$ in a DAG	cf. Section 2.2.1
$\text{ne}(a)$	neighbors of a variable $a \in \mathcal{V}$ in a graph	cf. Section 2.2.1
$a \perp b \mid \mathcal{S}$	d-separation of a and b given \mathcal{S} in a DAG	cf. Section 2.2.1
$a \not\perp b \mid \mathcal{S}$	d-connection of a and b given \mathcal{S} in a DAG	cf. Section 2.2.1

Probabilities and Independences

cf. Section 2.1

$p(a)$	(marginal) probability distribution for variable a
$p(a, b), p(\mathcal{S})$	joint probability distribution for several variables
$p(a \mathcal{S})$	conditional probability distribution for a given \mathcal{S}
$a \perp b$	marginal independence of the variables a and b
$a \not\perp b$	marginal dependence of the variables a and b
$a \perp b \mathcal{S}$	conditional independence of a and b given \mathcal{S}
$a \not\perp b \mathcal{S}$	conditional dependence of a and b given \mathcal{S}
CIDs	conditional independences and dependences

Scoring Functions

cf. Section 3.1

N	sample size, i.e. number of cases in the data set	
$N(\cdot)$	cell counts or frequencies in a contingency table	
N'	equivalent sample size	
F	absolute scoring function (strictly positive)	
f	logarithmic absolute scoring function, i.e. $f = \log F$	
G	relative scoring function (strictly positive)	
$g(\cdot, \cdot, \cdot), g(\cdot, \cdot)$	logarithmic relative scoring function, i.e. $g = \log G$	
$g_{\text{col}}(\cdot, \cdot, \cdot m_{\text{int}})$	relative scoring function concerning collider candidates in the PDAG m_{int}	cf. Section 6.1.1
α	significance level	
γ	threshold value concerning relative scores	
d_{f}	degrees of freedom	
$d(\cdot)$	deviance (difference)	
AIC	Akaike Information Criterion	
BIC	Bayesian Information Criterion	

Rules and other Symbols in Algorithms

$X, Y, [a, b], \dots$	pair of variables, e.g. $X = [a, b] = [b, a]$, it can correspond to an edge, si-path or sc-path	cf. Section 4.3.1
$\mathbf{X}, \mathbf{Y}, \dots$	set of pairs of variables	cf. Section 4.3.1
$SI_i([a, b], \mathbf{X})$	si-path (i enumerates the variants)	cf. Section 4.3.1
$SC_i([a, b], \mathbf{X})$	sc-path (i enumerates the variants)	cf. Section 4.3.1
\mathbf{E}	condition set regarding edges	cf. Section 5.1
\mathbf{C}	condition set regarding sc-paths	cf. Section 5.1
\mathbf{I}	condition set regarding si-paths	cf. Section 5.1
$IR(X, \mathbf{E}, \mathbf{I})$	i-rule concerning the edge X	cf. Section 5.1
$CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$	c-rule concerning the edge X	cf. Section 5.1
\mathfrak{R}	set of rules	cf. Section 5.1
\mathfrak{R}^A	subset of \mathfrak{R} , corresponds to an ambiguous region	cf. Section 5.3.2
\mathfrak{R}_c^A	a copy of \mathfrak{R}^A	cf. Section 5.3.2
$\mathbf{PA}(X)$	parents of X in the condition graph	cf. Section 5.2.2
$\mathbf{AN}(X)$	ancestors of X in the condition graph	cf. Section 5.2.2

1

Introduction

Uncertainty is present in many areas of life, prevailing for various reasons. For instance, in medicine it is sometimes difficult to determine the disease of a person on the basis of the observed symptoms. This is because a disease often entails a particular symptom only with some probability, but not with certainty. Moreover, some symptoms can only be specified loosely, e.g. "high fever" does not exactly begin at a particular temperature. Furthermore, the disease inferred might be based on incomplete knowledge, because a doctor might not know, for instance, about the patient's visit to a certain country, increasing chances of an infection with a particular kind of bacteria. In many areas of science and engineering, when an exact description of a large system is too involved, one often resorts to a tractable model which is a good approximation to the actual system. This is another source of uncertainty. For instance, in artificial intelligence the analysis of texts is often based on a probabilistic treatment, where the text is considered as a "bag of words", i.e. without a syntax. Texts are then characterized by the probabilities that certain words or combinations thereof are present.

Building intelligent systems for reasoning under uncertainty is one of the main challenges in the field of artificial intelligence (AI). Various frameworks for dealing with uncertainty have been used in AI, like fuzzy logic [154] or the Dempster-Schafer theory [42, 136]. Over the past one or two decades, probability theory has gained influence in the AI community, as it is a sound theory for dealing with uncertainty. Probability theory has been applied in statistics in order to induce information from data. This is typically done by means of so-called hypothesis tests, where a hypothesis like "Does smoking have an impact on lung cancer?" is falsified or not. For a long time, statistical analysis has been restricted to only a *small* number of variables. This had two reasons. First, statistical analysis of complex hypotheses involving a *large* number of variables had not been well understood from a theoretical point of view. Second, the analysis of complicated hypotheses was intractable until powerful computers became available.

Bayesian networks have become the standard model for coping with uncertainty in AI. They were developed by the AI community to build probabilistic expert systems for reasoning under uncertainty [116]. One of their main advantages is their sound theoretical basis in the framework of statistics and probability theory. A Bayesian network is a *probabilistic model*

which describes the *multivariate* probability distribution for a set of variables. In particular, it is designed for domains with a *large* number of variables. The basic idea is to display the associations among the variables, namely the (*conditional*) *independences and dependences*, by means of a graph. For this reason, Bayesian networks belong to the class of *graphical models*.

Figure 1.1 shows a simplistic Bayesian network structure, where the edges represent a dependence between the (binary) variables "electricity failure?" and "light failure?" as well as between the former and "computer failure?". The absence of the edge between "light failure?" and "computer failure?" indicates that these two variables are independent conditional on "electricity failure?". This means that "light failure?" is *irrelevant* to "computer failure?", and vice versa, once the state of "electricity failure?" is known. For instance, if one knows that there is no electricity failure, the light-bulb and the computer fail independently of each other, as each of which can break by chance. However, when the state of "electricity failure?" is unknown then "light failure?" and "computer failure?" depend on each other, as they both depend on "electricity failure?" in Figure 1.1. This is also intuitively clear, as an electricity failure entails both light and computer failures. It is hence crucial to distinguish between "direct" and "indirect" associations between variables.

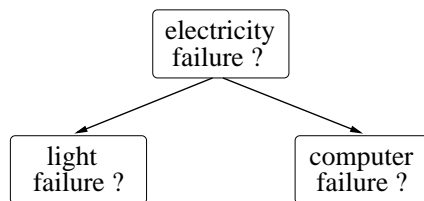


Figure 1.1: A simplistic Bayesian network structure illustrating direct and indirect associations among the three variables.

The graph in Figure 1.1 can also be understood intuitively when interpreted in a causal manner, namely "electricity failure?" can be considered as a common cause of "light failure?" and "computer failure?". This implies that the occurrence of "light failure?" and "computer failure?" can be related when nothing is known about "electricity failure?". Conversely, when the state of the common cause is known, "light failure?" and "computer failure?" can become independent. When Bayesian networks are used in a causal setting, they are also named *causal networks*. Their use as causal networks has been very appealing to the AI community [116, 142], since knowledge on causal relations renders well-understood interventions possible [117], and hence enables one to control the behavior of complex systems. The causal interpretation of a Bayesian network structure is, however, only appropriate under certain conditions. For an overview of this area of current research the reader is referred to [70, 82, 102, 116–118, 142, 148].

Besides the visualization of conditional independences and dependences, a Bayesian network model can also describe the joint probability distribution for a set of variables in a *quantitative* manner, as a Bayesian network comprises also parameters besides its graphical structure. It is beneficial that this description is quite modular so that complex systems can be characterized by combining smaller units. When the values of some variables are known, a Bayesian network can be used for predicting the states of the other variables. This process is called *inference* (for

an overview, see e.g. [21]). The result is not a particular state, but a *probability distribution* over the states of the variables to be predicted, i.e. a Bayesian network describes *probabilistic* associations among the variables in a domain. Since the reader might be familiar with neural networks, let us mention that, for instance in feed-forward neural-networks, the state of an output-variable depends *deterministically* on the inputs, rather than probabilistically.

There are basically two ways for constructing a Bayesian network. First, a system which is well-understood can be modeled by having a Bayesian network built manually by experts. A causal interpretation of the edges and their orientations is often useful in this situation. So-called *object oriented Bayesian networks* were developed to aid the construction of large models [96, 119]. Major fields of application were systems for medical diagnosis and for troubleshooting.

Alternatively, Bayesian networks can be *learned* from given data. This means that a learning algorithm can induce the structure as well as the parameters of a Bayesian network from data. Of course, also a combination of both approaches is possible, where the expert knowledge can serve as prior knowledge incorporated into a learning algorithm. In structural learning in Bayesian networks, all variables in a domain are treated equally, i.e. there is no distinct class-variable. This is typical for *unsupervised* learning. Since an induced Bayesian network structure visualizes the associations among the various variables in a domain, structural learning can provide new insights which help understand the associations among the variables. The extraction of new knowledge from data is called *data mining*. Automatic learning procedures can hence supplement a statistician's analysis of data. Estimating the parameters of a Bayesian network is typically a subproblem of learning its structure. Once a Bayesian network, including its parameters, has been learned, it can be used for quantitative predictions. Collaborative filtering is a recent application where Bayesian networks have proven to yield very accurate results [15]. This is employed, for instance, in online book-stores where Bayesian networks are trained on the data collected from customers, and subsequently used to recommend those books which have a high probability of being interesting to a customer. The World Wide Web will certainly be a main playground for data mining, as it allows to access an enormous amount of data, while the costs of collecting data are tremendously reduced at the same time. Biotechnology and genetics are another field where data mining techniques can help in understanding the processes underlying the data.

1.1 Structural Learning in Bayesian Networks

There have evolved two main approaches to structural learning. Both employ heuristics, as the exact problem is NP-hard [14, 27, 84]. The one approach uses a scoring function which is optimized by means of a heuristics search strategy. The use of Bayesian scoring functions is suggested, besides "classical" ones like the Akaike Information Criterion, because Bayesian network structures are *directed* graphs without cycles so that Equation 3.1.26 can be applied in a natural way (cf. Section 3.1.4). Local search is a very popular, general-purpose search strategy [90]. The main problem of this sort of approach are local maxima of the scoring

function, at which heuristic search-strategies can get stuck.

The other approach is constraint-based, relying on the conditional independences and dependences (CIDs) induced from the data. This method is, however, only well-understood when certain assumptions hold (cf. Section 1.2 for more details). Unfortunately, when the CIDs are inferred from finite data by means of *statistical tests*, as typical in practical applications, this approach can merely be considered as a heuristics (see also Section 1.2). An advantage of this approach is that it does not suffer from local optima, as a scoring function applying to the entire Bayesian network structure is not used. Because this method makes use of a particular property of Bayesian networks, namely the CIDs, it is very efficient in inducing sparse Bayesian network structures, as it generally requires a reasonably small number of independence tests only [23, 142]. This approach can, however, be quite inefficient when the induced graph is dense, since a large number of tests have to be carried out in this case. Regarding domains which require a dense graph, however, Bayesian networks might not be the most appropriate kind of model, since only very limited insight in the associations among the variables is typically gained by interpreting a dense Bayesian network structure. Furthermore, when a Bayesian network model is used for quantitative predictions, e.g. in a probabilistic expert system, inference is typically very time-consuming or even intractable in dense graphs. Thus, one might only be interested in the induction of a Bayesian network in domains where the resulting graph is sparse.

1.2 The Constraint-Based Approach applied to Finite Data Sets

The efficiency of the constraint-based approach when inducing sparse Bayesian network structures is very appealing [23, 24, 142], as other heuristic learning algorithms can be quite time-consuming, particularly in domains with a large number of variables. Unlike other learning algorithms, the constraint-based approach requires two additional assumptions about the probability distribution implied by the data. Let us just mention them here, as they are discussed in detail in the Sections 2.2.4, 3.3.2 and 4.1:

- the probability distribution is perfectly known, i.e. without error, and
- the probability distribution fulfills the so-called *faithfulness assumption*.

If these assumptions hold, the constraint-based approach can be shown to yield the *correct* Bayesian network structure [140, 148], the so-called perfect map. Note that this is not guaranteed for the other approaches aimed at optimizing a scoring function, as they can get stuck at local optima.

In practice, however, these assumptions need not hold. In fact, they can only be expected to hold in the asymptotic limit, i.e. when an infinite amount of data is available. This is because

sampling noise is typically present in *finite* data sets, causing the probability distribution implied by the data to differ from the *true* distribution (from which the data had been sampled). This implies that the constraint-based approach generally yields "almost correct" Bayesian network structures given sufficiently *large* data sets, as confirmed by many experiments reported in the literature [23, 24, 142]. Except for these experiments, not much attention has been paid to the behavior of the constraint-based approach when given *finite* data sets. Finite rather than infinite data sets are, however, typical for practical applications. Given *finite* sample sizes, the two assumptions above are not guaranteed to hold. This is particularly apparent regarding the first assumption, as it is well-known in statistics that an independence test can fail, i.e. a dependence can erroneously be induced instead of an independence, and vice versa. This is called a Type I and a Type II error, respectively. The results of *several* tests are combined by the constraint-based approach in order to construct the Bayesian network structure. Since some of the test results might be incorrect and since the various test results might depend on each other in some unknown manner, the error of the induced Bayesian network structure is not under control. In statistics, this is a well-known problem in the area of multiple testing.

This thesis is concerned with the constraint-based approach applied to *finite* data. Section 4.1 provides a more specific motivation of the approach taken in this thesis, as various terms introduced in the following two chapters are required. Nevertheless, let us mention the basic idea in the following: the above assumptions underlying constraint-based algorithms are dropped. Instead, we prefer the point of view that the constraint-based approach is aimed at finding the optimal Bayesian network structure with respect to a scoring function. The use of a scoring function is similar to the alternative approach to structural learning mentioned in Section 1.1. This point of view is motivated by the fact that the use of a scoring function is well-understood, and the optimal Bayesian network structure is well-defined also in those cases where *finite* data sets are given.

In order to understand the constraint-based approach in the framework of optimizing a scoring function, the notion of *relative scoring functions* is introduced. They are concerned with the differences of the scores rather than with the scores themselves. This allows us to use scoring functions like the Bayesian Information Criterion or the posterior probability in this approach, instead of the χ^2 -test commonly employed by constraint-based algorithms. Our point of view reveals that the network structures induced by the constraint-based approach from finite data sets tend to contain too few edges, compared to the optimal graph. Moreover, the performance of the constraint-based approach can considerably be improved by employing the so-called *necessary path condition* which we derive from properties of optimal Bayesian network structures. Moreover, *model uncertainty* can be discovered when using this extension. Of course, it can only be explored up to some degree, as an exact treatment of model uncertainty is intractable except for domains with a rather small number of variables. Model uncertainty generally prevails when small data sets are given. Because data sets, even when considered as "large", might often be small compared to the number of joint states of the variables in a large domain, model uncertainty may better not be ignored in many applications. Accounting for model uncertainty can improve predictive accuracy, e.g. by model averaging, as well as help avoid incorrect conclusions drawn from the induced structures. Concerning the latter issue, we show that the

multiple solutions induced by means of the necessary path condition can be visualized in a single graph. This is because the various graphs have usually many edges in common and differ only regarding the presence of a few edges. Such a graph can typically be interpreted much more easily than a list of the various solutions.

The necessary path condition can efficiently be implemented in terms of rules. The presence of edges is induced by simplifying the set of rules. This renders a systematic construction of all the multiple solutions possible which can be induced by means of the necessary path condition. Furthermore, this scheme can take advantage of the fact that the various Bayesian network structures typically have many edges in common. For this reason, the computational cost of inducing possibly many graphs is only slightly increased compared to established constraint-based approaches, which induce only a single graph by definition. The constraint-based approach is also appropriate for parallel computation which we have realized in a simple master and slave scheme.

In case that the probability distribution implied by the data fulfills the assumptions required by state-of-the-art constraint-based approaches, the necessary path condition yields the same graph as the other constraint-based approaches do, and it is hence asymptotically correct. Given finite data, however, the necessary path condition is an important extension of the constraint-based approach.

Besides the necessary path condition, concerning the presence of edges, also an operator for inducing the orientations of edges is proposed in this thesis. This operator is aimed at optimizing a scoring function, which renders it quite robust when given finite data in our experiments, in contrast to typical constraint-based schemes relying on the induced independences and dependences.

We also argue that the edges induced to be present by the constraint-based approach are contained in a (locally) optimal Bayesian network structure with a high degree of certainty. This indicates the necessity of a learning step subsequent to the constraint-based approach if one aims at inducing (locally) optimal Bayesian network structures.

1.3 Thesis Overview

In the next chapter, we give a brief overview of Bayesian networks and their properties important to learning. Chapter 3 reviews the main approaches to structural learning in Bayesian networks, namely the constraint-based approach and the one aimed at optimizing a scoring function. A main part of this chapter is spent on introducing so-called relative scoring functions, and on deriving some of their properties. The main benefit is that both kinds of approaches to learning can be based on relative scoring functions.

In Chapter 4, the constraint-based approach is viewed as a particular search strategy aimed at inducing the optimal Bayesian network structure, with respect to a scoring function initially specified. Having discussed this point of view, the main result of this, rather theoretical, chapter

is derived, namely the so-called necessary path condition and its variants. This leads immediately to an extension of the constraint-based approach applied to finite data. Moreover, we discuss properties of the graphs induced by constraint-based approaches from finite data.

The constraint-based approach can typically be split into two parts. The first one is concerned with inducing the presence of edges, thus employing an undirected graph. The second part eventually yields the Bayesian network structure by finding the orientations of the edges. In Chapter 5, we describe the details of the algorithm which induces the *presence* of edges, based on the necessary path condition. The latter is transformed into a set of rules which can then efficiently be simplified. The reduced set of rules can indicate model uncertainty regarding the presence of edge – of course, only to a limited degree. As a consequence, possibly several undirected graphs are obtained. The so-called summary graph with its ambiguous regions is introduced, a single graph visualizing the different structures of the induced undirected graphs. Moreover, we are concerned with the efficiency of such an extended constraint-based algorithm. Various experiments are carried out in order to compare the extended algorithm with other popular learning algorithms. Finally, the incorporation of prior knowledge is described, as well as how missing data can be handled.

Chapter 6 presents a greedy operator aimed at finding the optimal *orientations* with respect to a scoring function. It is not based on induced conditional independences and dependences, like typical schemes in constraint-based algorithms, which are quite unstable when given finite data sets. An advantage of this operator is its robustness when given finite data sets, as confirmed in our experiments. Moreover, a third learning step is added to the constraint-based algorithm in order to induce (locally) optimal network structures. This is necessary because our theoretical considerations yield that the constraint-based approach tends to induce graphs with too few edges compared to optimal Bayesian network structures. The resulting algorithm is compared to established learning algorithms using both artificial and real-world data.

In the Conclusions, the main results of this thesis are summarized and an outlook to future work is given.

2

Bayesian Networks

This chapter gives a brief introduction to Bayesian networks. After defining a Bayesian network model, we focus on properties important to structural learning. An overview of the various aspects of Bayesian networks is provided in [21, 35, 47, 79, 116]. For a first exposure to Bayesian networks, the reader is referred to [89], whereas a comprehensive treatment from a theoretical point of view can be found in [101].

2.1 Conditional Independence and Dependence

A Bayesian network is a probabilistic model based on the notion of *conditional independences and dependences* (CIDs). Let us hence introduce some notation from probability theory.

The probability distribution for a random variable a is denoted as $p(a)$, and the *joint* probability for several variables, e.g. for $\mathcal{S} = \{a, b, c\}$, reads $p(a, b, c)$ or $p(\mathcal{S})$. It describes the probability for each *joint state* or *configuration* of the variables in the set \mathcal{S} . Let the set of all configurations of the variables in a set \mathcal{S} be designated as $\mathcal{I}(\mathcal{S})$, and the configurations as $i, j, k, \dots \in \mathcal{I}(\mathcal{S})$. The above probabilities are also called *marginal* probabilities, as opposed to *conditional* probabilities. The probability of a variable a conditional on a set of variables, e.g. $\mathcal{S} = \{b, c\}$, is designated as $p(a|b, c)$ or $p(a|\mathcal{S})$, and it is defined as $p(a|b, c) = p(a, b, c)/p(b, c) > 0$.

Two random variables a and b are said to be *marginally independent* when their joint probability $p(a, b)$ factors like $p(a, b) = p(a)p(b)$. Such an independence is denoted as $a \perp b$ [39]. It means that the state of a is irrelevant to the state of b , and vice versa. Similarly, two variables a and b are *independent conditional* on some other variables contained in the set \mathcal{S} if the conditional probability factors like $p(a, b|\mathcal{S}) = p(a|\mathcal{S})p(b|\mathcal{S})$ given $p(\mathcal{S}) > 0$. This is equivalent to $p(a|b, \mathcal{S}) = p(a|\mathcal{S})$ or $p(b|a, \mathcal{S}) = p(b|\mathcal{S})$ provided that $p(a, \mathcal{S}), p(b, \mathcal{S}), p(\mathcal{S}) > 0$. This means that b is irrelevant to a if the joint state of the variables in the set \mathcal{S} is known. Let such a conditional independence be denoted as $a \perp b | \mathcal{S}$ [39].

If two variables are not independent then they are said to be *dependent*. Again, one can dis-

tinguish between a marginal dependence of two variables a and b , designated as $a \not\perp b$, and a dependence conditional on a set S , denoted as $a \not\perp b \mid S$.

2.2 Bayesian Networks and their Properties

Throughout this thesis, let the set of *all* the variables in a domain be denoted as \mathcal{V} , and the variables as $a, b, \dots \in \mathcal{V}$. The *conditional independences and dependences* (CIDs) underlying a multivariate probability distribution for the variables in \mathcal{V} are reflected by the *graphical structure* of a Bayesian network, the so-called *directed acyclic graph* (DAG). It is described in the next section. The other component of a Bayesian network is a set of *parameters*, rendering a quantitative description of probability distributions possible (cf. Section 2.2.2).

2.2.1 Directed Acyclic Graph (DAG)

Let us first be concerned with the directed acyclic graph (DAG) m , i.e. the Bayesian network structure. It is related to probability distributions in such a way that the random variables correspond to the nodes or vertices in the DAG. Let us hence use *variables*, *nodes* and *vertices* interchangeably in this thesis. Besides the variables, *directed* edges or arcs are present in a DAG. No undirected edges are allowed. A directed edge which is oriented from variable a to b is denoted as $a \rightarrow b$ or $b \leftarrow a$. When the orientations of the edges are ignored, an undirected graph is obtained, named the *skeleton* of the DAG m . Let the skeleton be designated as \tilde{m} . An undirected edge between two variables $a, b \in \mathcal{V}$ is denoted as $a \sim b$.

A *path* between two variables $a, b \in \mathcal{V}$ is a sequence of edges $a = x_0 \sim x_1 \sim \dots \sim x_r = b$ irrespective of their orientations, and a *directed path* from a to b is a sequence of edges $a = x_0 \rightarrow x_1 \rightarrow \dots \rightarrow x_r = b$ such that $x_{i-1} \rightarrow x_i$ for all $i = 1, \dots, r$. It is crucial that the Bayesian network structure – hence the notion of a directed *acyclic* graph – does *not* contain *directed cycles*, i.e. a directed path which begins and ends at the same variable. This is illustrated in the simplistic DAG shown in Figure 2.1, where no directed cycle occurs although the variables b , d , g and e are involved in a *loop*, i.e. the corresponding skeleton contains a closed path.

The following notation is inspired by a family tree. The *parents* $\text{pa}_m(a)$ of a variable a in the DAG m is the set of variables $v \in \mathcal{V}$ such that there exists a directed edge $v \rightarrow a$. The variable a is called a *child* of v . If the orientations are disregarded, the notion of *neighbors* of a variable $a \in \mathcal{V}$ is useful, denoted as $\text{ne}(a)$. The set $\text{ne}(a)$ contains all the variables adjacent to a . When several ”generations” are considered, a variable $v \in \mathcal{V}$ belongs to the *ancestors* of a , denoted as $\text{an}(a)$, if there exists a directed path from v to a . Conversely, a directed path exists from a variable a to each of its *descendants*. In Figure 2.1, for instance, the variable d has the parents a and b , i.e. $\text{pa}_m(d) = \{a, b\}$, and the children f and g . The neighbors of d comprise a, b, f and g in the skeleton \tilde{m} . Furthermore, the variable a has the descendants c, d, f and g , while the ancestors of g are given by $\text{an}(g) = \{a, b, d, e\}$.

Due to its acyclicity, a DAG entails an *ancestral ordering* on the variables. This is a *total*

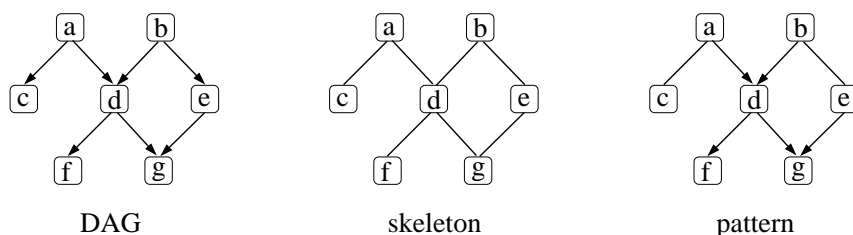


Figure 2.1: A simplistic DAG m , its skeleton \tilde{m} , and the pattern of the equivalence class.

ordering relation \prec , i.e. for all $a, b, c \in \mathcal{V}$ it has to hold that (1) $a \prec b \vee b \prec a$, (2) $a \prec b \wedge b \prec a$ implies $a = b$, (3) $a \not\prec a$, and (4) $a \prec b \wedge b \prec c$ implies $a \prec c$. Typically, a DAG does not determine a unique total ordering. It only defines a *partial* ordering. In Figure 2.1, two valid ancestral orderings – among others – are $a \prec b \prec c \prec d \prec e \prec f \prec g$ or $b \prec a \prec d \prec f \prec e \prec g \prec c$.

There are essentially two alternatives of how to relate the graphical structure of a Bayesian network model to the conditional independences and dependences underlying probability distributions, namely the *Markov assumptions* and the *d-separation criterion*. Regarding the former, three variants have to be distinguished, namely the directed pairwise, directed local and directed global Markov properties (see e.g. [101]). If the probability distribution is strictly positive, all three Markov properties can be shown to be equivalent, as one might desire. The Markov assumption is made in many areas of research in order to approximate problems which are too involved otherwise. The Markov approximation says essentially that the state of a variable depends only on the state taken by the variables in its vicinity, i.e. the latter shield this variable from the influence of the other variables in the domain. For instance, in the analysis of time-series it is often assumed that only the current state of the world has an impact on the state of the world in the next time-step, independent of the past. Another example is pattern recognition, where it is often assumed that only the values of neighboring pixels depend directly on each other.

Let us now focus on the second criterion, i.e. the d-separation criterion [116], as it will be applicable more easily later in this thesis. As desired for reasons of consistency, the d-separation criterion can be shown to be equivalent to the directed global Markov property of Bayesian networks [103], and hence also to the other Markov properties provided that the probability distribution is strictly positive.

Definition 2.1 (D-Separation [116]) *In a DAG, two disjoint sets of variables \mathcal{A} and \mathcal{B} are d-separated by a third set $\mathcal{S} \subseteq \mathcal{V} \setminus (\mathcal{A} \cup \mathcal{B})$, denoted as $\mathcal{A} \perp \mathcal{B} \mid \mathcal{S}$, if and only if along every path between a variable in \mathcal{A} and a variable in \mathcal{B} there is a variable s satisfying one of the following two conditions:*

- *s has converging edges and none of s or its descendants are in \mathcal{S} , or*
- *s does not have converging edges and $s \in \mathcal{S}$.*

A variable $s \in \mathcal{V}$ has converging edges, “ $\rightarrow s \leftarrow$ ”, when the preceding and the successive variable along the path are both parents of s . Not only the presence of edges but also their orientations are thus crucial in order to render variables d-separated. This definition implies that a path between two variables is *blocked* when one of the above two conditions is fulfilled, and *activated* otherwise. Hence, two variables $a, b \in \mathcal{V}$ are d-separated by a set \mathcal{S} , when all paths between them are blocked. It is important to note that $a \perp b \mid \mathcal{S}$ does not imply $a \perp b \mid \mathcal{S}'$ for other $\mathcal{S}' \neq \mathcal{S}$. This holds even if $\mathcal{S}' \supset \mathcal{S}$, since a blocked path can become activated by an additional variable $s' \in \mathcal{S}'$, $s' \notin \mathcal{S}$ due to the first condition in Definition 2.1. This is an important difference to Markov networks (cf. also Section 2.3). In the DAG shown in Figure 2.1, for instance, the variables d and e are d-separated given b , but they are not d-separated given both b and g . If two disjoint sets \mathcal{A} and \mathcal{B} are not d-separated by a set \mathcal{S} then they are also said to be *d-connected*, designated as $\mathcal{A} \not\perp \mathcal{B} \mid \mathcal{S}$.

Let us now relate the structure of a Bayesian network to the set of probability distributions it describes by means of the d-separation criterion. Namely, a d-separation $a \perp b \mid \mathcal{S}$ read off the DAG m entails the conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$ in the probability distributions described. In other words, all the probability distributions exhibit the conditional independences implied by the DAG m , but might differ from each other due to different values of the parameters θ chosen in the Bayesian network. Particular choices of the parameters can entail probability distributions which imply additional independences *not* represented in the DAG. However, it can be shown that *almost all* probability distributions described by Bayesian networks (in a measure-theoretic sense) imply a conditional independence if and only if the DAG represents the corresponding d-separation [111] (cf. also the Sections 2.2.4 and 4.1).

2.2.2 Recursive Factorization of the Probability Distribution

The d-separation criterion and the Markov properties are equivalent to yet another characteristic of Bayesian networks, namely to the decomposability of its probability distribution when the latter is strictly positive. The proof of the equivalence is essentially based on [74]. A Bayesian network model with the DAG m describes a probability distribution for a set of variables \mathcal{V} which factorizes recursively like

$$p(\mathcal{V}) = \prod_{v \in \mathcal{V}} p(v \mid \text{pa}_m(v)). \quad (2.2.1)$$

The *multivariate* probability distribution for \mathcal{V} hence decomposes into *univariate* probability distributions, rendering the Bayesian network model to be quite modular. The set of parameters θ of a Bayesian network model is the set of conditional probabilities $p(v \mid \text{pa}_m(v))$, where $\text{pa}_m(v)$ denotes the parents of a variable v in the DAG m . This sort of factorization has two consequences regarding learning Bayesian networks. First, since each of the conditional probabilities typically involves only a small number of variables, i.e. $|\text{pa}_m(v)| \ll |\mathcal{V}|$ for all $v \in \mathcal{V}$, the parameters of a Bayesian network can be estimated from finite data with increased reliability. Second, the parameters of a Bayesian network, since they are conditional probabilities, can directly be calculated from the probability distribution implied by the data. In contrast, param-

parameter estimation in other graphical models generally requires iterative procedures, e.g. iterative proportional scaling in Markov networks [37] (cf. also Section 2.3).

2.2.3 Markov Equivalence

A DAG determines a unique set of probability distributions which exhibit conditional independences according to the d-separation criterion. The opposite situation is encountered in structural learning, where the question arises whether the conditional independences and dependencies implied by a probability distribution determine a unique DAG. In general, the answer is negative. For instance, the DAGs m_0 , m_1 and m_2 in Figure 2.2 display the same d-separations and d-connections. Hence, they describe the same set of probability distributions. This can also be seen from the factorization of the probability distribution (cf. Equation 2.2.1), as it is equivalent to the d-separation criterion given strictly positive distributions. The following three factorizations are entailed by the DAGs m_0 , m_1 and m_2 in Figure 2.2:

$$p(a, b, c) = \underbrace{p(a|c)p(c|b)p(b)}_{m_0} = \underbrace{p(a|c)p(b|c)p(c)}_{m_1} = \underbrace{p(c|a)p(b|c)p(a)}_{m_2} \quad (2.2.2)$$

The identity of the different factorizations is apparent, as the second product follows from the first one due to the general law $p(c|b)p(b) = p(b, c) = p(b|c)p(c)$, and similarly the third one from the second one, assuming $p(\cdot) > 0$. Although the DAG m_3 contains the same edges as the previous three DAGs, it represents different d-separations and d-connections. This is because the DAG m_3 has a *collider* at variable c . A collider, or v-structure, is an ordered triple of variables $a, c, b \in \mathcal{V}$ such that the edge between a and b is absent and $a \rightarrow c \leftarrow b$.

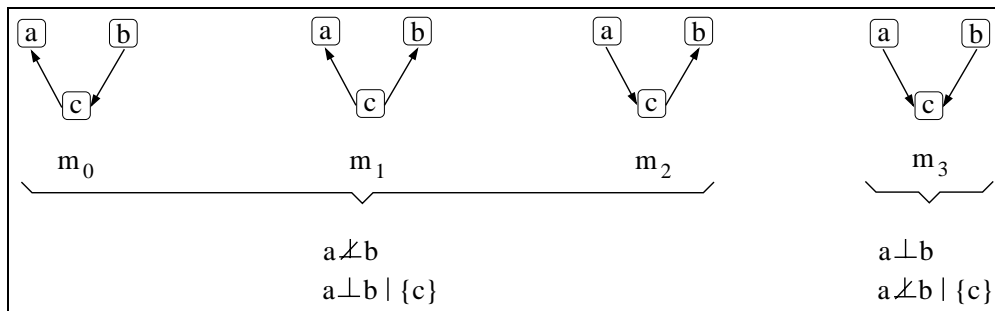


Figure 2.2: The DAGs m_0 , m_1 and m_2 belong to the same equivalence class, whereas m_3 is contained in a different one.

It is obvious that the d-separations and d-connections establish a reflexive, symmetric and transitive relation among DAGs, which is called *Markov equivalence* relation. This allows to partition the space of DAGs into *equivalence classes*, where each equivalence class contains all the DAGs which are Markov equivalent to each other. It can be shown that two DAGs are equivalent if and only if they have the same skeleton and the same colliders [148]. This means that all the edges not involved in a collider can be oriented arbitrarily as long as an additional collider does not occur. The edges can hence be divided into ones with a reversible

orientation and others with an irreversible one. Edges with an irreversible orientation are also called *compelled edges* [26]. An equivalence class is often visualized by means of a so-called *pattern* [148], where edges with a reversible orientation are displayed without directions so that only irreversible orientations are shown. This is illustrated in Figure 2.1.

The existence of equivalence classes has three consequences regarding structural learning in Bayesian networks. First and most important, only the equivalence class rather than a particular DAG can generally be induced from the probability distribution implied by the data. Only if statistical learning algorithms are supplemented with additional knowledge, e.g. about the orientations of edges, a particular DAG can be determined. Second, the undirected edges in the induced pattern of the equivalence class clearly cannot be interpreted as an association between a cause and a consequence if no additional knowledge is available which implies certain orientations. Third, structural learning in the search space of DAGs suffers from some disadvantages, which can be overcome in the search space of equivalence classes [28, 107]. Learning algorithms operating in the space of equivalence classes are, however, computationally very involved [28, 107]. The constraint-based approach is an efficient way to directly induce the pattern of an equivalence class, which was used extensively in [142] (cf. also Section 3.3.2).

If one likes to explore the various DAGs contained in the same equivalence class, the notion of a *covered edge* is useful [26]. A directed edge between two variables $a, b \in \mathcal{V}$ is said to be covered in a DAG m if it holds that $\text{pa}_m(a) \setminus \{b\} = \text{pa}_m(b) \setminus \{a\}$, i.e. when disregarding the edge between a and b they have the same parents. The orientation of a covered edge can be reversed in order to obtain another equivalent DAG [26]. This has important consequences for the properties of scoring functions derived in Section 3.1.3. Moreover, this notion will also be used in the discussion of the experiments in the Sections 5.6 and 6.3.

2.2.4 Perfect Map and Faithfulness

Structural learning is aimed at inducing DAGs which describe a given probability distribution implied by the data. This section thus addresses the question whether a Bayesian network structure is capable of describing all the conditional independences and dependences (CIDs) implied by an arbitrary data set. For instance, assume that a probability distribution for the three variables a , b and c implies the conditional independences $a \perp\!\!\!\perp c \mid \{b\}$ and $b \perp\!\!\!\perp c \mid \{a\}$, while the other associations are dependences. Obviously, the CIDs cannot all be represented in a single DAG (cf. also Figure 2.3). This suggests to represent either all the (conditional) independences or all the (conditional) dependences in a DAG. This leads to the notion of I-maps and D-maps (see e.g. [116]).

Definition 2.2 (Independence Map (I-Map)) *A DAG is an independence map of a probability distribution for a set of variables \mathcal{V} if and only if $a \perp b \mid \mathcal{S} \Rightarrow a \perp\!\!\!\perp b \mid \mathcal{S}$, or equivalently $a \not\perp\!\!\!\perp b \mid \mathcal{S} \Rightarrow a \not\perp b \mid \mathcal{S}$ (for all $a, b \in \mathcal{V}$, $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$).*

This means that every d-separation displayed in an I-map entails a (conditional) independence in the probability distribution. However, the probability distribution might imply additional

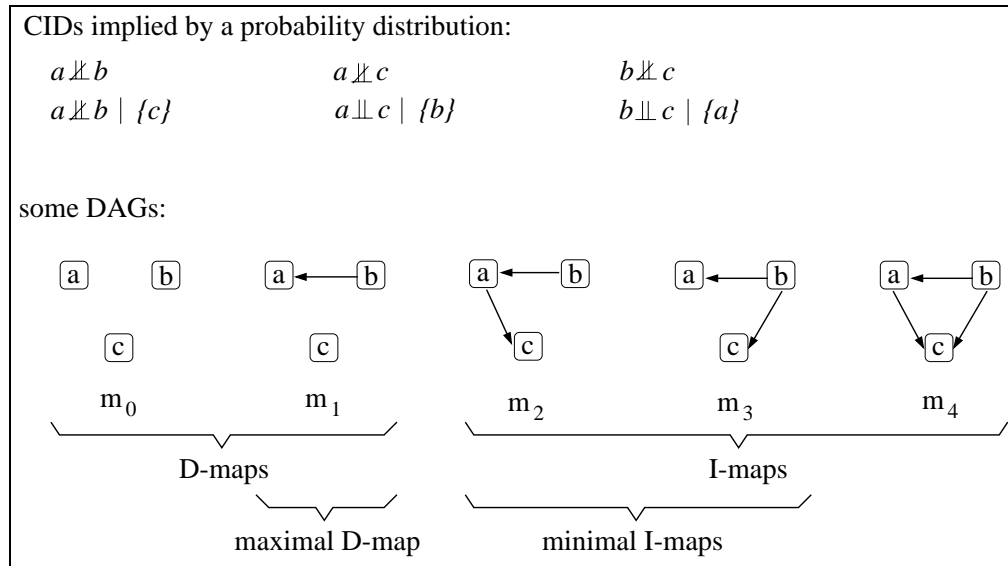


Figure 2.3: A DAG might not be capable of representing all the CIDs implied by an arbitrary probability distribution.

independences not represented in an I-map. Apparently, the complete graph, i.e. the one where all edges are present, is a trivial I-map of any probability distribution. More interesting are *minimal I-maps*, which are minimal in the sense that no edge can be removed without destroying the property of being an I-map. In general, there can exist *several* minimal I-maps which are not Markov equivalent. This is illustrated in Figure 2.3, depicting two minimal I-maps containing different edges.

Definition 2.3 (Dependence Map (D-map)) A DAG is a dependence map of a probability distribution for a set of variables \mathcal{V} if and only if $a \not\perp b \mid S \Rightarrow a \not\perp b \mid \mathcal{S}$, or equivalently $a \perp b \mid \mathcal{S} \Rightarrow a \perp b \mid S$ (for all $a, b \in \mathcal{V}$, $S \subseteq \mathcal{V} \setminus \{a, b\}$).

A D-map hence represents all the conditional independences underlying the probability distribution, but additional independences can possibly be read off a D-map. Equivalently, all dependences implied by a D-map are present in the probability distribution. Thus, the empty graph is a trivial D-map, as it does not represent any dependences. A DAG is a *maximal D-map* if no edge can be included into the graph without losing the property of being a D-map. Like I-maps, there can generally be several D-maps which are not equivalent to each other.

Since D-maps represent fewer dependences, they typically contain a smaller number of edges than I-maps. In the special case that a probability distribution is such that a DAG can simultaneously be a minimal I-map and a maximal D-map of the distribution, this DAG is called a *perfect map*, and the probability distribution is called *faithful*.

Definition 2.4 (Perfect Map and Faithfulness) *A DAG is a perfect map of a probability distribution for a set of variables \mathcal{V} if and only if $a \perp\!\!\!\perp b \mid \mathcal{S} \Leftrightarrow a \perp b \mid \mathcal{S}$ (for all $a, b \in \mathcal{V}$, $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$). A probability distribution is faithful if and only if it has a perfect map.*

If a probability distribution is faithful then the perfect map is *uniquely* determined (up to Markov equivalence, of course), unlike I-maps and D-maps. The constraint-based approach to structural learning in Bayesian networks typically requires the assumption that the probability distribution implied by the data be faithful. This is discussed in the Sections 3.3.2 and 4.1, as this assumption may not hold for probability distributions implied by *finite data*.

2.3 Graphical Models

Graphical models can be traced back to the beginning of the twentieth century, where they were first used in statistical physics [68]. Independently, they have evolved in genetics and statistics [7, 153]. In statistics, so-called *log-linear models* have long been a popular approach to modeling multivariate probability distributions for discrete variables. For a comprehensive account, the reader is referred to [11]. *Graphical models* can be considered as a special case of so-called *hierarchical log-linear models*. Important papers which laid the modern foundations of graphical models include [36, 104, 105, 149–151]. The class of *graphical models* comprises not only Bayesian networks but also *Markov networks* and *chain graphs*. While the latter two models are widely used in statistics, Bayesian networks have attracted a lot of attention in the AI community over the past one or two decades. In contrast to Bayesian networks, Markov networks contain solely undirected edges.¹ Since chain graphs allow for both directed and undirected edges, Bayesian networks and Markov networks might be viewed as special cases of chain graphs.

The common characteristic of the various graphical models is the graphical visualization of conditional independences and dependences (CIDs) by means of Markov properties. Regarding a Markov network, a conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$ corresponds to a separation of a and b in the undirected graph such that every path between a and b contains a variable in \mathcal{S} . This entails again a factorization of the joint probability distribution, similar to Bayesian networks. However, it cannot be expressed in terms of (conditional) probabilities in general. This renders parameter estimation in Markov networks quite involved, as those factors of the probability distribution, the so-called clique potentials, cannot be calculated directly from the data, but usually require some iterative schemes like iterative proportional scaling [37].

Figure 2.4 depicts the classes of probability distributions – in terms of CIDs – which can be captured by Bayesian networks and Markov networks [116]. It is obvious that certain CIDs can only be represented by DAGs while others can only be reflected by the undirected graph of a Markov network. An example for each of which is depicted in Figure 2.5. Moreover,

¹Although the undirected graph of a Markov network looks like the skeleton of a DAG, they may not be confused.

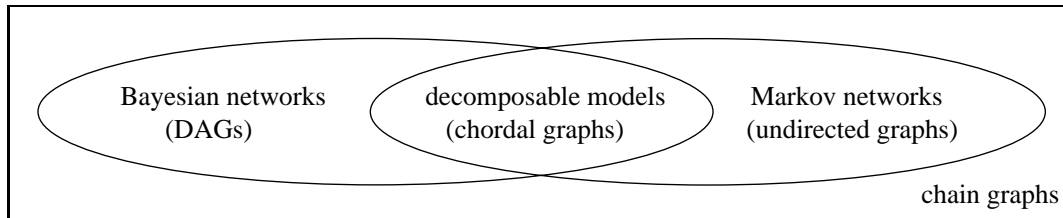


Figure 2.4: Correspondence among graphical models (their graphs are put into brackets).

there is also an overlap between both models, indicating that certain probability distributions can be described by both a Bayesian network and a Markov network [149]. Many structural learning algorithms focus on inducing these so-called *decomposable models*, as they have some properties which ease their induction from data. The structure of decomposable models is typically visualized by an (undirected) chordal graph. An undirected graph is called *chordal*, or *triangulated*, if every closed path (loop) of length four or more has at least one chord, i.e. an edge between two variables along the loop which is not contained in the loop. This yields that the undirected graph in Figure 2.5 is not chordal. Equivalently to an (undirected) chordal graph, the structure can also be represented by a DAG without colliders.² This indicates that the probability distribution described by a decomposable model factorizes into a product of conditional probabilities – which is similar to Bayesian networks – while its graph can be displayed using undirected edges – corresponding to a Markov network.



Figure 2.5: The CIDs represented by the DAG (left) cannot all be displayed in the undirected graph of a Markov network. Conversely, there is no DAG which can represent all the CIDs shown by the Markov network structure (right).

Finally, let us note that the probability distributions described by graphical models belong to the exponential family (see e.g. [6, 64–66]). In detail, graphical models with hidden variables, i.e. variables which are unobserved, belong to the so-called stratified exponential family [64–66]. In the absence of hidden variables, Bayesian networks are curved exponential families, while Markov networks are linear exponential families [64–66]. Some consequences regarding structural learning are mentioned in Section 5.9.

2.4 Some Related Tools for Data Analysis

There are various approaches to data analysis. Let us depict two of them which are closely related to Bayesian networks. Association rules are a popular technique in data mining. Asso-

²Note that converging arrows $a \rightarrow b \leftarrow c$ are not forbidden if also the edge between a and c is present in the DAG.

ciation rules are aimed at measuring the strength of statistical associations between variables. In the domain depicted in Figure 1.1, the variables "light failure?" and "computer failure?" are only indirectly related to each other. Since association rules cannot account for such *conditional* independences, they yield, a possibly strong, statistical association between "light failure?" and "computer failure?". One can imagine that, in domains with many variables, a few direct associations entail a large number of indirect associations. In order to gain insight into such a domain it is thus desirable to distinguish between direct and indirect associations, as the former entail the latter. For this reason, association rules might provide less insight into the interrelations of the various variables in a domain than Bayesian networks, or graphical models in general, do.

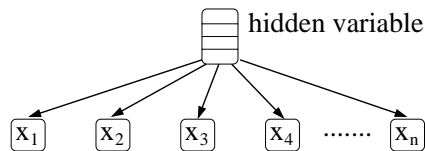


Figure 2.6: A naïve Bayesian network can be used for clustering.

Let us also mention clustering, another popular approach to data analysis, which is very closely related to a particular kind of Bayesian networks, namely the so-called naïve Bayesian network. Clustering is aimed at grouping similar configurations contained in the data set into the same cluster. The data can be considered as a set of cases, each of which representing a configuration. The similarity of configurations can be measured by a metric defined in the space of configurations. The latter is typically called feature space in the context of clustering. The feature space is hence partitioned into disjoint subspaces, where each subspace is called a cluster or a scenario. Typically, each configuration belongs to a certain cluster, and the data set can hence be understood in terms of (a few) scenarios, capturing the main effects underlying the data. If the data set is such that the scenarios are not well-separated from each other, one might prefer *soft* clustering, where a case is fractionally assigned to multiple clusters. Besides other methods, this can be achieved by means of a naïve Bayesian network. It contains a hidden, discrete variable whose states correspond to the different clusters or scenarios. Moreover, the different variables (or features) of the domain, $\mathcal{V} = \{x_i : i = 1, \dots, n\}$, are assumed to be independent conditional on the hidden variable. This is sketched in Figure 2.6. The fraction with which a configuration belongs to a particular cluster is measured in terms of probability. For this reason, this approach to clustering has a sound theoretical basis. When a configuration is entered as evidence into the naïve Bayesian network, carrying out inference yields the probabilities with which this configuration belongs to the various clusters, i.e. states of the hidden variable. Since all the information is contained in the parameter values (rather than in the structure) of the Bayesian network, visualization of the various clusters or scenarios can be difficult. Various approaches are applicable for learning the parameters in the presence of a hidden variable (cf. also Section 5.9).

3

Structural Learning

Structural learning, or model selection, is concerned with determining a Bayesian network which describes the probability distribution implied by the data to some degree. This degree is usually measured by means of a so-called *scoring function*. The latter maps the, possibly high-dimensional, space of Bayesian networks to a one-dimensional one, typically to the real numbers. The properties of Bayesian networks outlined in the previous chapter suggest certain features of scoring functions. Throughout the remainder of this thesis, it is important to keep in mind the notion of *relative* scoring functions introduced in this chapter. Additionally, interrelations among the different relative scores are derived in the following. As will become clear, the main consequences are that certain combinations of induced conditional independences cannot coincide, and that a speed-up of the computations can be achieved.

Many learning algorithms try to find the (global) optimum of the scoring function. Unfortunately, the task of actually finding the optimal Bayesian network is an NP-hard problem [14, 27, 84]. For this reason, one has to resort to a heuristic *search strategy* which can efficiently determine a Bayesian network close to optimum. In Section 3.3, an overview of popular learning algorithms is given. As reviewed in Section 3.2, it can be beneficial to allow for model uncertainty, instead of learning a single Bayesian network.

3.1 Scoring Functions

Scoring functions assign a score to a Bayesian network structure in the light of data. A data set is a collection of cases, and a case contains an instantiation, i.e. a value, for each variable in the domain. It is typically assumed that the data set comprises *independently and identically distributed* cases, i.e. all cases are drawn from the same probability distribution, and this is done for each case independently of the others. Let a scoring function be denoted as F and the score of a DAG m as $F(m)$. It is assumed that F is strictly positive. One can hence use the *logarithmic* scoring function $f(\cdot) = \log F(\cdot)$, which helps avoid numerical problems. For brevity, also $f(m)$ is simply called scoring function (without "logarithmic") when this is clear

from the context.

3.1.1 Decomposability

A main property of the probability distribution described by a Bayesian network is its recursive factorization, or decomposability (cf. Equation 2.2.1). Hence, one might desire that also the scoring function F shows this property,

$$F(m) = \prod_{v \in \mathcal{V}} F_v(v|\text{pa}(v)), \quad (3.1.1)$$

where each term $F_v(v|\text{pa}(v))$ involves only a variable $v \in \mathcal{V}$ and its parents $\text{pa}(v)$. Here, we have explicitly distinguished among the different functions $F_v, v \in \mathcal{V}$. This is, however, often skipped for concise notation in the remainder of this thesis. The decomposability of the scoring function F holds for all the commonly used scoring functions (cf. also Section 3.1.4) in the case of complete data and in the absence of hidden variables. The issue of incomplete data is addressed in Section 5.9. The logarithmic scoring function reads analogously

$$f(m) = \sum_{v \in \mathcal{V}} f_v(v|\text{pa}(v)). \quad (3.1.2)$$

3.1.2 Relative Scoring Functions

The scoring function F assigns a score $F(m)$ to the *entire* graph m . Hence, the evaluation of the score $F(m)$ can be computationally tedious, in particular in domains with a large number of variables $v \in \mathcal{V}$. Popular search strategies explore the search space of all DAGs $m \in \mathcal{M}$ by proceeding from one intermediate graph m_i to some other m_{i+1} , where successive graphs are typically very similar to each other. The score $F(m_{i+1})$ can hence be computed efficiently by reusing many terms $F_v(v|\text{pa}(v))$ already evaluated for the score $F(m_i)$ (cf. Equation 3.1.1). Alternatively, a *relative scoring function* may be employed, as described in the following.

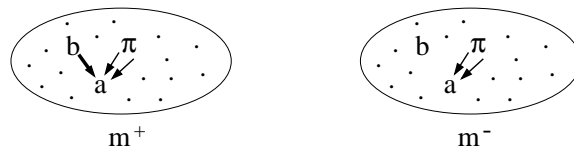


Figure 3.1: The DAGs m^+ and m^- are distinct regarding the presence of exactly one edge, namely $a \leftarrow b$. $\pi = \text{pa}_{m^-}(a)$ depicts the parents of a in the DAG m^- , and the dots symbolize the remaining network structure assumed to be identical in both DAGs.

Instead of an *absolute* scoring function F concerning an *entire* graph, a *relative* scoring function is concerned with the *difference* in the scores of two graphs. In particular, let us consider two DAGs m^+ and m^- which differ in exactly one edge, namely let the edge $a \leftarrow b$ be present

in m^+ while it is absent in m^- ($a, b \in \mathcal{V}$). This as illustrated in Figure 3.1. The relative scoring function, designated as G , is given by the following ratio involving the absolute scoring function F ,

$$\begin{aligned}
 G(m^+, m^-) &= \frac{F(m^+)}{F(m^-)} \\
 &= \frac{F_a(a|\text{pa}_{m^+}(a))}{F_a(a|\text{pa}_{m^-}(a))} \prod_{v \in \mathcal{V} \setminus \{a\}} \frac{F_v(v|\text{pa}_{m^+}(v))}{F_v(v|\text{pa}_{m^-}(v))} \\
 &= \frac{F_a(a|\text{pa}_{m^-}(a) \cup \{b\})}{F_a(a|\text{pa}_{m^-}(a))} \\
 &= \frac{F_a(a|\text{pa}_{m^+}(a))}{F_a(a|\text{pa}_{m^+}(a) \setminus \{b\})} \\
 &= G(a, b, \text{pa}_{m^\pm}(a) \setminus \{b\}),
 \end{aligned} \tag{3.1.3}$$

where we again assume that $F(m) > 0$ for all DAGs $m \in \mathcal{M}$. The second line in Equation 3.1.3 shows that the decomposability of F entails a considerable simplification of G . Because the DAGs m^+ and m^- differ only in the parents of the variable a , i.e. $\text{pa}_{m^+}(a) \setminus \{b\} = \text{pa}_{m^-}(a)$, all the other ratios $F_v(v|\text{pa}_{m^+}(v))/F_v(v|\text{pa}_{m^-}(v))$, $v \in \mathcal{V} \setminus \{a\}$, are equal to one. Consequently, the relative score $G(m^+, m^-)$ depends only on the variables a, b and $\text{pa}_{m^\pm}(a) \setminus \{b\}$, where m^\pm is a short-hand notation for the DAGs m^+ and m^- . Note the importance of the parents of the variable a to which the edge $a \leftarrow b$ points, while the parents of the other variable are irrelevant. It is crucial that G is independent of all the other variables and hence of the Bayesian network structure involving the remaining variables. In other words, the relative score $G(a, b, \text{pa}_{m^\pm}(a) \setminus \{b\})$ is determined once the parents of the variable a are known. Instead of considering the *entire* graph, relative scoring functions are hence concerned with single *edges* given the parents of the corresponding variable.

If the relative score $G(a, b, \text{pa}_{m^\pm}(a) \setminus \{b\})$ is larger than one then the absolute score $F(m^+)$ is larger than $F(m^-)$. Regarding the edge $a \leftarrow b$, this means that its presence is favored more than its absence given the parents $\text{pa}_{m^\pm}(a) \setminus \{b\}$. Similarly, if the relative score $G(a, b, \text{pa}_{m^\pm}(a) \setminus \{b\}) < 1$ then the absence of the edge $a \leftarrow b$ is preferred given the parents $\text{pa}_{m^\pm}(a) \setminus \{b\}$. The logarithm of the relative scoring function is given by

$$\begin{aligned}
 g(m^+, m^-) &= \log G(a, b, \text{pa}_{m^\pm}(a)) \\
 &= f_a(a|\text{pa}_{m^\pm}(a) \cup \{b\}) - f_a(a|\text{pa}_{m^\pm}(a)) \\
 &= g(a, b, \text{pa}_{m^\pm}(a) \setminus \{b\}).
 \end{aligned} \tag{3.1.4}$$

For brevity, we often use $g(a, b) := g(a, b, \emptyset)$ in the remainder of this thesis.

Let us now be concerned with the difference in the scores of two DAGs m_1 and m_2 which are distinct from each other with respect to *several* edges. The difference in the scores, $\Delta(m_2, m_1) := f(m_2) - f(m_1)$, can be expressed in terms of the relative scoring function

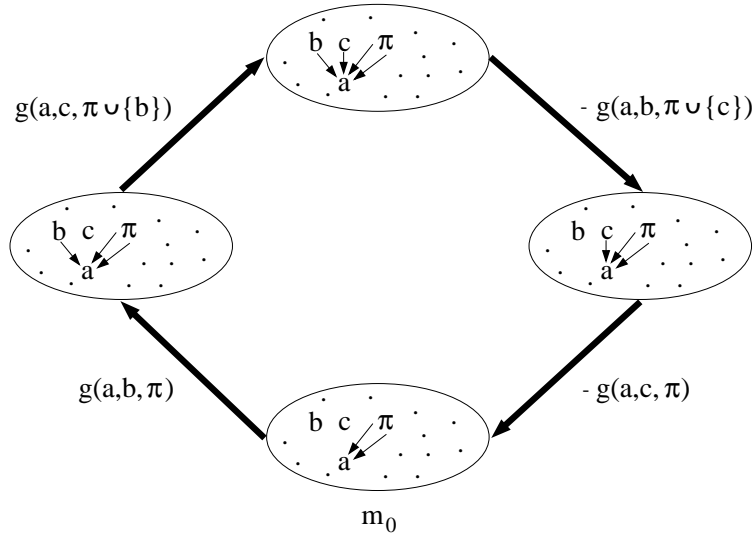


Figure 3.2: A cycle in the space of DAGs. π depicts the parents of a in m_0 , and the dots symbolize the remaining network structure assumed to be identical in these DAGs.

g as follows. The DAG m_1 can be viewed as the beginning of a sequence of DAGs where successive DAGs differ from each other by the presence or absence of exactly one edge, and the DAG m_2 is at the end of that sequence. The sum over the relative scores regarding successive DAGs results in the overall difference $\Delta(m_2, m_1)$. It is obvious that such a sequence of DAGs exists. Moreover, any sequence of DAGs between m_1 and m_2 can be used for calculating this difference. This is because the absolute scores $f(m_1)$ and $f(m_2)$ depend only on the graphs m_1 and m_2 . In physics, this is a typical property of so-called *potentials*.

In the following, let us focus on a particular sequence of DAGs, namely the one which begins and ends at the same DAG (cf. Figure 3.2). This sequence is hence a cycle. Let us begin to proceed along this cycle at the DAG m_0 . First, two edges are subsequently included and then removed in the reversed order. Since f is a potential, the sum of the involved relative scores along the cycle has to vanish,¹

$$0 = g(a, b, \pi) + g(a, c, \pi \cup \{b\}) - g(a, b, \pi \cup \{c\}) - g(a, c, \pi),$$

or equivalently

$$g(a, b, \pi \cup \{c\}) - g(a, c, \pi \cup \{b\}) = g(a, b, \pi) - g(a, c, \pi), \quad (3.1.5)$$

where $\pi = \text{pa}_{m_0}(a)$. All four relative scores in Equation 3.1.5 have the same variable in their first argument, namely a . In contrast, the variable in the second argument can change. Since this relation among relative scores has to hold for all $a, b, c \in \mathcal{V}$ and for all $\pi \subseteq \mathcal{V} \setminus \{a, b, c\}$,

¹In physics, a well-known consequence is the fact that perpetual motion cannot exist.

it follows immediately for all $S \subseteq \mathcal{V}$ which contain at least three variables that

$$g(a, b, S \setminus \{a, b\}) - g(a, c, S \setminus \{a, c\}) = g(a, b, S \setminus \{a, b, c\}) - g(a, c, S \setminus \{a, b, c\}). \quad (3.1.6)$$

This equation shows that relative scores cannot all be independent of each other, since three relative scores determine the value of the fourth one in Equation 3.1.6. Hence, only certain combinations of relative scores can coincide, namely the ones in accordance with the constraints implied by Equation 3.1.6. This means, however, that the other combinations of values are forbidden. For instance, given that the edge $a \leftarrow b$ is favored to be absent due to $g(a, b, \emptyset) < 0$ while the presence of the edge $a \leftarrow c$ is supported by $g(a, c, \emptyset) > 0$, the relative scores $g(a, c, \{b\}) < 0$ and $g(a, b, \{c\}) > 0$ cannot be found simultaneously. When negative values of the relative scoring function are interpreted as induced (conditional) independences, Equation 3.1.6 thus forbids the simultaneous occurrence of various combinations of induced (conditional) independences.

Moreover, regarding the third argument of the relative scores in Equation 3.1.6, the sets on the left hand side comprise one additional variable compared to one on the right hand side. This can be exploited to speed up the computations of scores, as discussed in Section 5.4.2.

In this section, we have defined the relative scoring function g by means of the absolute scoring function f . Conversely, f can also be expressed in terms of g . This is, however, not of practical use concerning efficient computation. Starting out from the empty graph, a sequence of DAGs can be obtained by successively including edges (in some arbitrary order) until one arrives at the DAG m of concern. Each time an edge $v \leftarrow w$ is included, the relative scoring function is evaluated on the basis of the current DAG m_{int} . This eventually results in

$$f(m) = f(m_{\text{empty}}) + \sum_{v \in \mathcal{V}} \sum_{w \in \text{pa}_m(v)} g(v, w, \text{pa}_{m_{int}}(v)), \quad (3.1.7)$$

where the sums are only carried out over the variables $v \in \mathcal{V}$ with $\text{pa}_m(v) \neq \emptyset$. The score $f(m_{\text{empty}})$ of the empty graph is often a meaningless constant, which might hence be set to zero for simplicity. The only exception occurs when a (posterior) probability is used as a scoring function (cf. Section 3.1.4). In this case, $f(m_{\text{empty}}) \neq 0$ accounts for the normalization of the (posterior) probability distribution. Up to a constant, absolute and relative scoring functions can thus be used as equivalent alternatives.

3.1.3 Score Equivalence

An important property of Bayesian network structures is Markov equivalence, as outlined in Section 2.2.3. Consequently, one can only aim at determining the equivalence class rather than a particular DAG on the basis of the probability distribution implied by the data. Only in a causal discovery setting or in the case of prior knowledge about the orientations of some edges, it might be reasonable to distinguish among equivalent DAGs. In many cases, however, there is no reason for favoring a particular DAG more than another equivalent one. Consequently, it is

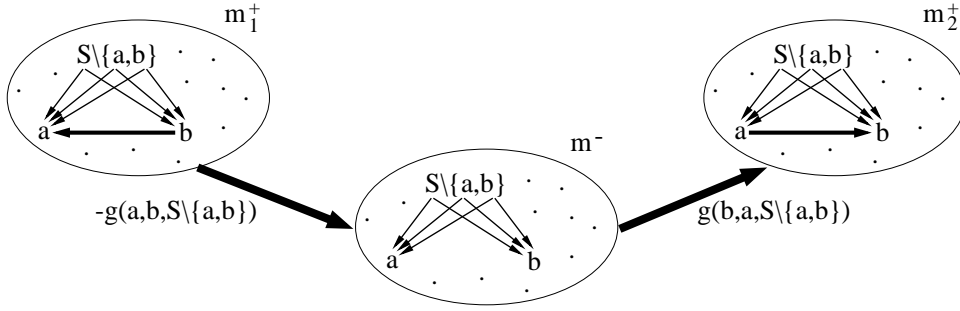


Figure 3.3: The equivalent DAGs m_1^+ and m_2^+ are assumed to be identical except for the orientation of the edge between a and b , which is covered because $\text{pa}(a) \setminus \{b\} = \text{pa}(b) \setminus \{a\} = S \setminus \{a, b\}$.

often desirable that an (absolute) scoring function assigns the same score to equivalent DAGs. This is called *score equivalence*. This section is concerned with the properties of absolute and relative score-equivalent scoring-functions.

In Figure 3.3, the DAGs m_1^+ and m_2^+ are equivalent, since they differ only in the orientation of the edge between a and b , which is a *covered* edge (cf. Section 2.2.3). Hence, the scores $f(m_1^+)$ and $f(m_2^+)$ have to be identical, and it follows immediately for a *score-equivalent* relative scoring-function g that

$$g(a, b, S \setminus \{a, b\}) = g(b, a, S \setminus \{a, b\}) \quad (3.1.8)$$

for all $S \subseteq \mathcal{V}$ which contain at least two variables, and for all $a, b \in S$. This implies that there are more interdependencies among score-equivalent relative scores than there are among non-score-equivalent relative scoring-functions. Whereas the first variable in each of the scores has to be the same in Equation 3.1.6, Equation 3.1.8 allows the first and the second variables to be swapped. Regarding score-equivalent scoring-functions, there is hence also a relation between the relative scores $g(a, b, S \setminus \{a, b\})$ and $g(c, d, S \setminus \{c, d\})$ where *all* the $a, b, c, d \in S \subseteq \mathcal{V}$ can be pairwise different. For all $S \subseteq \mathcal{V}$ which comprise at least four variables this is given by

$$\begin{aligned} & g(a, b, S \setminus \{a, b\}) - g(c, d, S \setminus \{c, d\}) \\ &= g(a, b, S \setminus \{a, b, c\}) - g(a, c, S \setminus \{a, b, c\}) \\ & \quad + g(a, c, S \setminus \{a, c, d\}) - g(c, d, S \setminus \{a, c, d\}) \\ &= g(a, b, S \setminus \{a, b, c\}) - g(b, c, S \setminus \{a, b, c\}) \\ & \quad + g(b, c, S \setminus \{b, c, d\}) - g(c, d, S \setminus \{b, c, d\}) \\ &= g(a, b, S \setminus \{a, b, d\}) - g(a, d, S \setminus \{a, b, d\}) \\ & \quad + g(a, d, S \setminus \{a, c, d\}) - g(c, d, S \setminus \{a, c, d\}) \\ &= g(a, b, S \setminus \{a, b, d\}) - g(b, d, S \setminus \{a, b, d\}) \\ & \quad + g(b, d, S \setminus \{b, c, d\}) - g(c, d, S \setminus \{b, c, d\}). \end{aligned} \quad (3.1.9)$$

Like in Equation 3.1.6, the third arguments of the relative scores in the first line contain one

variable more than the ones in the remaining lines in Equation 3.1.9. This can be exploited in order to achieve computations of score-equivalent scoring-functions which are more efficient than the ones of non-score-equivalent scoring-functions (cf. Section 5.4.2), particularly when the scores are evaluated in ascending order of the set \mathcal{S} .

Concerning absolute scoring functions f which are decomposable and score-equivalent, it is apparent from the equivalence of the DAGs m_1^\dagger and m_2^\dagger in Figure 3.3 that

$$f_a(a|\mathcal{S} \setminus \{a\}) + f_b(b|\mathcal{S} \setminus \{a, b\}) = f_a(a|\mathcal{S} \setminus \{a, b\}) + f_b(b|\mathcal{S} \setminus \{b\}) \quad (3.1.10)$$

or equivalently

$$f_a(a|\mathcal{S} \setminus \{a\}) - f_b(b|\mathcal{S} \setminus \{b\}) = f_a(a|\mathcal{S} \setminus \{a, b\}) - f_b(b|\mathcal{S} \setminus \{a, b\}) \quad (3.1.11)$$

for all $\mathcal{S} \subseteq \mathcal{V}$ with at least two variables and for all $a, b \in \mathcal{S}$. Like in the case of relative scoring functions, score-equivalence entails additional interrelations among absolute scoring functions. Moreover, the terms on the left hand side in Equation 3.1.11 contain one variable more than the ones on the right hand side. This can be exploited for efficient computations of absolute scores in a similar manner as for relative scores (cf. Section 5.4.2).

3.1.4 Popular Scoring Functions

The most popular Bayesian networks contain solely discrete random variables with a multinomial distribution. Alternatively, domains may contain solely continuous variables with a multivariate Gaussian distribution (e.g. [63]) or a combination of discrete and continuous variables. In the so-called conditional Gaussian distribution [80, 105], the discrete variables have a multinomial distribution and the continuous ones a multivariate Gaussian distribution conditional on the discrete variables. Note that a Gaussian distribution implies a linear association among the variables. Continuous variables with non-linear associations can be modeled by various approaches [60, 85, 86, 113]. Since the main purpose of this work is the development of a new search strategy rather than the extension of the scoring function to some particular distribution, we focus on discrete variables with a multinomial distribution for simplicity when becoming more specific in the following. Nevertheless, many parts of this section also hold in general.

Maximum Likelihood

A common measure of goodness of fit of a Bayesian network model with the DAG m is the maximum (log-)likelihood $l(\hat{\theta}_m) = \log L(\hat{\theta}_m) := \log p(D|m, \hat{\theta}_m)$, where $\hat{\theta}_m$ is the maximum likelihood estimate of the parameters of the model, and D represents the data. Since the likelihood decomposes as $L(\hat{\theta}_m) = \prod_{v \in \mathcal{V}} L_v(\hat{\theta}_v | \text{pa}_m(v))$ given complete data and in the absence of hidden variables, each "local" likelihood L_v can be maximized separately. The parameters of a Bayesian network are the conditional probabilities of a variable given its parents in the

DAG (cf. Section 2.2.2). In the domain of discrete variables with a multinomial distribution, the maximum-likelihood estimate of the parameters is given by

$$\hat{\theta}_{v|\text{pa}(v)}(i, k) = \frac{N_{v,\text{pa}(v)}(i, k)}{N_{\text{pa}(v)}(k)}, \quad (3.1.12)$$

which one might have guessed right away, as it is the maximum-likelihood estimate of the conditional probability that the variable v is in state $i \in \mathcal{I}(v)$ given that its parents $\text{pa}(v)$ are in the (joint) state $k \in \mathcal{I}(\text{pa}(v))$. The frequencies or (cell) counts $N(\cdot)$ of the various configurations are sufficient statistics in this domain. The maximum (log-)likelihood of a Bayesian network with the DAG m and the parameters $\hat{\theta}_m$ reads

$$l(\hat{\theta}_m) = \sum_{v \in \mathcal{V}} \sum_{\substack{i \in \mathcal{I}(v), \\ k \in \mathcal{I}(\text{pa}(v))}} N_{v,\text{pa}(v)}(i, k) \log \frac{N_{v,\text{pa}(v)}(i, k)}{N_{\text{pa}(v)}(k)}, \quad (3.1.13)$$

where the first sum ranges over all the variables $v \in \mathcal{V}$ in the domain, and the second one over the various configurations $\mathcal{I}(\cdot)$. Note that all graphical models do not allow maximum-likelihood estimates to be calculated in closed form from the cell counts. In fact, this is a particular property of Bayesian networks. For instance, Markov networks require iterative schemes, like iterative proportional scaling [41], in order to calculate maximum-likelihood estimates approximately. A concise description of maximum-likelihood estimation in graphical models with discrete variables is provided in [100].

Regarding two nested models² with the DAGs m^+ and m^- it can be shown that the inequality $l(\hat{\theta}_{m^+}) \geq l(\hat{\theta}_{m^-})$ holds (cf. e.g. [55]). It is thus not advisable to use the maximum likelihood by itself for structural learning, since the maximum likelihood of the complete graph is larger than the one of any other graph. Some practical problems related to a saturated model³ are the intractability of inference in the model and the possibly large demand for computer memory in order to store all the parameters. Although a saturated model usually performs very well on the training data, i.e. the data which was provided for learning, its predictions are in general very poor on new data, i.e. data which was not used for learning. This is an important issue, called *over-fitting*. It was noticed in classification problems long ago [17]. Ever since researchers have paid attention to avoid over-fitting in order to achieve improved predictions. A popular way of approximately assessing the quality of the predictions of a model in the light of new data is *cross validation* [145]. It is briefly described in Appendix A, and we applied this technique in our experiments with real-world data in Section 6.3.2. Prequential validation [40] is a more recent approach addressing the problem of over-fitting, too. However, it has been used less commonly so far.

Maximum likelihood is, however, not useless in structural learning as long as it is combined with a term penalizing model complexity, since the optimal model with respect to such a scoring function is not necessarily the saturated one. Hence, this helps avoid over-fitting. A trade-off between model complexity and the goodness of fit of a model is thus essential for scoring

²Two Bayesian networks are said to be nested if it holds for their DAGs m^+ and m^- that $\text{pa}_{m^+}(v) \supseteq \text{pa}_{m^-}(v)$ for all $v \in \mathcal{V}$, i.e. the one is a submodel of the other.

³A Bayesian network model is called saturated when its DAG is complete, i.e. no edge is missing.

functions. When the goodness of fit is measured in terms of maximum likelihood, very popular scoring functions are the *Akaike Information Criterion (AIC)* [1,2] and the *Bayesian Information Criterion (BIC)*, also called (*Jeffreys-*)*Schwarz Criterion* [132]:

$$AIC : f_{AIC}(m) = l(\hat{\theta}_m) - |\hat{\theta}_m| \quad (3.1.14)$$

$$BIC : f_{BIC}(m) = l(\hat{\theta}_m) - \frac{1}{2}|\hat{\theta}_m| \log N \quad (3.1.15)$$

In both criteria,⁴ the term penalizing model complexity is measured in terms of the number of *independent* parameters which is given by

$$|\hat{\theta}_m| = \sum_{v \in \mathcal{V}} (|\mathcal{I}(v)| - 1) \cdot |\mathcal{I}(\text{pa}(v))|, \quad (3.1.16)$$

where $|\mathcal{I}(v)|$ denotes the number of states of the discrete variable $v \in \mathcal{V}$, and

$$|\mathcal{I}(S)| = \prod_{s \in S} |\mathcal{I}(s)| \quad (3.1.17)$$

is the number of (joint) states of a set of variables $S \subseteq \mathcal{V}$.⁵ Note that the number of independent parameters is $(|\mathcal{I}(v)| - 1)$ rather than $|\mathcal{I}(v)|$ conditional on each configuration of the parents, since the parameters in the Bayesian network model are normalized, as they are (conditional) probabilities. The main difference between the two penalty terms in the *AIC* and *BIC* is that the latter depends also on the number of cases N in the data set. Thus, the *BIC* generally favors less complex models than the *AIC* does (when $\log N > 2$). In fact, it was shown that the *AIC* tends to yield too complex models, even in the asymptotic limit [95, 137]. The value of the constant factor in the penalty term of the *AIC* is more or less arbitrary, as it is a consequence of *AIC*'s optimality with respect to the Kullback-Leibler divergence in the asymptotic limit. Other distance measures lead to different values [106]. In contrast, the *BIC* is asymptotically equivalent to the log-likelihood in prequential validation [40]. This is, however, only proven to hold for Bayesian networks without hidden variables [64–66, 78] (cf. also Section 5.9). Another popular measure is the *Minimum Description Length (MDL)* [125, 127, 128], which stems from information theory. Although it has a completely different origin than the *BIC*, both measures are essentially identical. Variants of the *MDL* metric can, for instance, be found in [13, 75, 99, 126].

In the case of complete data and in the absence of hidden variables, both the maximum log-likelihood and the term penalizing model complexity decompose into terms such that each of which involves only a variable and its parents. Hence, *AIC* as well as *BIC* can be used as relative scoring functions g (cf. Section 3.1.2):

$$AIC : g_{AIC}(a, b, S) = d(a \perp b | S) - d_f, \quad (3.1.18)$$

$$BIC : g_{BIC}(a, b, S) = d(a \perp b | S) - \frac{1}{2}d_f \log N, \quad (3.1.19)$$

⁴Regarding both *AIC* and *BIC*, we have omitted a factor of 2 which is often present.

⁵In case that $S = \emptyset$: $|\mathcal{I}(\emptyset)| = 1$.

where $a, b \in \mathcal{V}$ and $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$. As discussed in Section 3.1.2, these relative scoring functions are independent of the other variables $\mathcal{V} \setminus (\mathcal{S} \cup \{a, b\})$ in the domain. In the Equations 3.1.18 and 3.1.19, the logarithm of the likelihood ratio is written in terms of the deviance difference,⁶

$$\begin{aligned} d(a \perp\!\!\!\perp b \mid \mathcal{S}) &:= l(\hat{\theta}_{m+}) - l(\hat{\theta}_{m-}) \\ &= \log \frac{L(\hat{\theta}_{m+})}{L(\hat{\theta}_{m-})} = \log \frac{L_a(\hat{\theta}_{a \mid \mathcal{S} \cup \{b\}})}{L_a(\hat{\theta}_{a \mid \mathcal{S}})} \\ &= \sum_{\substack{i \in \mathcal{I}(a), \\ j \in \mathcal{I}(b), \\ k \in \mathcal{I}(\mathcal{S})}} N_{a,b,\mathcal{S}}(i, j, k) \log \frac{N_{a,b,\mathcal{S}}(i, j, k) N_{\mathcal{S}}(k)}{N_{a,\mathcal{S}}(i, k) N_{b,\mathcal{S}}(j, k)}. \end{aligned} \quad (3.1.20)$$

The deviance difference depends only on a, b and \mathcal{S} , and so do the degrees of freedom obtained from Equation 3.1.16,

$$d_f := |\hat{\theta}_{m+}| - |\hat{\theta}_{m-}| = (|\mathcal{I}(a)| - 1) \cdot (|\mathcal{I}(b)| - 1) \cdot |\mathcal{I}(\mathcal{S})|, \quad (3.1.21)$$

where $|\mathcal{I}(\mathcal{S})|$ is given in Equation 3.1.17. Instead of Equation 3.1.21, one might also use *adjusted* degrees of freedom [5, 11], in particular when the contingency table contains a large number of zero cell-counts.

Many constraint-based approaches to structural learning employ the χ^2 -independence-test (cf. Section 3.3.2). Let us note that it can essentially be rewritten such that it resembles a relative scoring function, namely like

$$g_{\chi^2}(a, b, \mathcal{S}) = d(a \perp\!\!\!\perp b \mid \mathcal{S}) - \frac{1}{2} \chi_{1-\alpha}^2(d_f), \quad (3.1.22)$$

where the quantile $\chi_{1-\alpha}^2(d_f)$ plays the role of the term penalizing model complexity. The quantile depends on the significance level α , e.g. 5%, as well as on the degrees of freedom d_f . Since it is a non-linear function of the latter, there cannot exist an absolute scoring function, though. Being aware of this fact, we nevertheless apply this "relative scoring function" and compare it to other relative scoring functions in our experiments (cf. the Sections 5.6.2 and 6.3.2). A positive sign of $g_{\chi^2}(a, b, \mathcal{S})$ corresponds to rejecting the conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$, while the latter may be accepted for negative values.

Marginal Likelihood and Posterior Probability

Bayesian statistics is an alternative to the classical (frequentist) approach.⁷ In many respects, the Bayesian approach can be considered as diametrically opposite to the frequentist one. A

⁶Again, the factor of 2 is omitted.

⁷The expressions *Bayesian statistics* and *Bayesian networks* may not be confused. While the former specifies the kind of statistical approach (Bayesian versus frequentist), taken to structural learning here, the latter is the name of the model under consideration here (as opposed to ,e.g., Markov networks, chain graphs, neural networks, etc.).

comprehensive account of Bayesian statistics is provided in [10]. The Bayesian approach treats the parameters of a Bayesian network model as random variables with some distribution, like the other random variables $v \in \mathcal{V}$ in the domain. The *marginal likelihood* of a DAG m in the light of the data D is given by

$$p(D | m) = \int d\theta_m p(D | \theta_m, m) p(\theta_m | m), \quad (3.1.23)$$

where $p(\theta_m | m)$ is the *a priori* probability for the parameters θ_m . Since the marginal likelihood can depend on the particular choice of the prior $p(\theta_m | m)$, a careful analysis of data usually requires a *sensitivity analysis* where the influence of the prior on the final result is examined.

Unlike *AIC* and *BIC*, described in the previous section, the marginal likelihood in Equation 3.1.23 does not exhibit a term explicitly penalizing model complexity. Nevertheless, such a penalty term is inherent in $p(D | m)$, as can be understood in various ways. First, integrating out the parameters in Equation 3.1.23 penalizes more complex models, i.e. models with a larger number of independent parameters, because these models can a priori describe a larger range of distributions. Second, the data D can be treated in a sequential manner, i.e. one case after the other is taken into account. Let the entire data set D comprise the cases C_i ($i = 1, \dots, N$), and let D_i denote the cases C_1, \dots, C_{i-1} . When the marginal likelihood is rewritten by the chain rule of probability theory as

$$p(D | m) = \prod_{i=1}^N p(C_i | D_i, m), \quad (3.1.24)$$

it becomes apparent that the prediction of the case C_i is based on the previous cases C_1, \dots, C_{i-1} [40]. This mechanism is kind of similar to cross validation (cf. also Appendix A). Third, the penalty regarding model complexity becomes explicitly visible (to some approximate degree) when the logarithm of the marginal likelihood is asymptotically approximated by the *BIC* (cf. Equations 3.1.28 and 3.1.15).

The posterior probability of a DAG m is related to its marginal likelihood by means of Bayes' theorem,

$$p(m | D) = \frac{p(m)}{p(D)} p(D | m), \quad (3.1.25)$$

where $p(m)$ is the *a priori* probability for the DAG m and $p(D)$ is the probability for the data D . It is apparent that the Bayesian approach allows the incorporation of prior knowledge regarding the parameters as well as concerning the Bayesian network structure. According to Equation 3.1.25, the learning task can be viewed as updating one's prior belief on the basis of the data, as the latter has an impact on the marginal likelihood, i.e.

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}. \quad (3.1.26)$$

The proportionality is due to the probability for the data $p(D)$ being unknown in general.

Instead of the posterior probability, one might use $p(m, D)$, like in [81], since it involves only $p(m)$ and $p(D | m)$. An alternative to this absolute scoring function is a relative scoring function regarding two DAGs m^+ and m^- . A popular choice is the posterior ratio,

$$\frac{p(m^+ | D)}{p(m^- | D)} = \frac{p(D | m^+) p(m^+)}{p(D | m^-) p(m^-)}, \quad (3.1.27)$$

which is the product of the prior ratio $p(m^+)/p(m^-)$ and the Bayes factor $B(m^+, m^-) = p(D | m^+)/p(D | m^-)$ [88]. A particular choice of the DAGs m^+ and m^- is depicted in Figure 3.1. Before being concerned with this, let us first consider the marginal likelihood in more detail.

Marginal Likelihood

From a practical point of view, the main problem of the Bayesian approach is the calculation of the marginal likelihood, because it involves the evaluation of the integral in Equation 3.1.23 which is very complicated in general. This problem can be tackled by methods like importance sampling [25, 115, 122], Markov chain Monte Carlo, in particular the Metropolis-Hastings algorithm [77, 112] and the Gibbs sampler [67], or by variational approximations (for an overview, see e.g. [93] and the references therein). In the following, the asymptotic approximation by means of the Bayesian Information Criterion (*BIC*) is described, as well as the use of conjugate priors, the latter rendering an exact analytical evaluation of the integral in Equation 3.1.23 possible.

The *BIC* appears in the asymptotic approximation to the *log marginal likelihood* by Laplace's method [132],

$$\log p(D | m) \approx f_{BIC}(m). \quad (3.1.28)$$

This is a valid approximation for Bayesian networks *without* hidden variables, as their probability distribution belongs to the curved exponential family [78]. In contrast, Bayesian networks with hidden variables describe a larger class of distributions, the so-called stratified exponential family, where the *BIC* may not be a valid approximation to the log marginal likelihood [64–66]. Only terms which are relevant in the asymptotic limit are retained in the *BIC*. When it is applied to finite data, this approximation may be improved by including terms with finite values, like $1/2|\hat{\theta}_m| \log(2\pi)$, as examined in [46]. Additional terms are present in Kashyap's criterion [92]. An overview of the various approximations to the marginal likelihood can be found in [30, 31, 93, 106, 133].

An advantage of this approximation is that it can easily be evaluated, like the maximum log likelihood. Moreover, the *BIC* avoids the explicit introduction of priors. However, the *BIC* is only a rough approximation to the log marginal likelihood. This is apparent from the fact that $\exp(g_{BIC})/B \not\rightarrow 1$ occurs for at least some priors in the asymptotic limit (B is the Bayes factor, and g_{BIC} is given in Equation 3.1.19). Nevertheless, $(\log B - g_{BIC})/\log B \rightarrow 0$ holds in the asymptotic limit (see e.g. [94]). The relative error is generally of the order of $\mathcal{O}(1)$. For some particular choices of the prior, e.g. the *unit information prior*, this error is only of the

order $\mathcal{O}(N^{-1/2})$, where N is the sample size [93, 94, 123]. The *BIC* is hence conveniently used in many learning algorithms, e.g. [22].

A Bayesian scoring function based on conjugate priors was first applied to structural learning in Bayesian networks in [33, 34], which was later extended in [79]. The idea behind conjugate priors is to choose the prior distribution for the parameters such that the prior belongs to the same family as the posterior distribution. Equation 3.1.26 illustrates that the posterior can then easily be used as a prior when learning from subsequent cases. Such a choice does not necessarily exactly reflect one's prior knowledge, but it allows the evaluation of the integral in Equation 3.1.23 in closed form. In order to calculate the marginal likelihood of a DAG analytically, a few assumptions are necessary, like complete data, (global and local) parameter independence [139] and parameter modularity [81]. With these assumptions, Bayesian networks are particularly suitable for developing a Bayesian scoring function like the marginal likelihood or the posterior probability of a DAG. Regarding discrete variables with a multinomial distribution, the conjugate prior belongs to the Dirichlet distribution. Other distributions for the variables require, of course, different prior distributions for the parameters (see e.g. [10]). For instance, the marginal likelihood of Bayesian networks containing continuous variables with a Gaussian distribution is calculated in [63]. Based on these assumptions, the marginal likelihood of a DAG m with discrete variables $v \in \mathcal{V}$ can be written as

$$p(D | m) = \prod_{v \in \mathcal{V}} \frac{\Omega(\{v\} \cup \text{pa}_m(v))}{\Omega(\text{pa}_m(v))} \quad (3.1.29)$$

where D is the data and $\text{pa}_m(v)$ are the parents of the variable $v \in \mathcal{V}$ in the DAG m , and

$$\Omega(S) = \prod_{j \in \mathcal{I}(S)} \frac{\Gamma(N_S(j) + N'_S)}{\Gamma(N'_S)}. \quad (3.1.30)$$

The function $\Omega(\cdot)$ maps the cell counts $N_S(j)$ of the various configurations $j \in \mathcal{I}(S)$ of a subset $S \subseteq \mathcal{V}$ to the real number $\Omega(S)$ by means of the Gamma function $\Gamma(\cdot)$. Although this notation differs slightly from the original one in [34, 81], it is equivalent to it. This notation is used to illustrate the different factors more clearly. The parameters N'_S arise from the Dirichlet prior. When they are chosen as $N'_S = N'_S(j) = N'p(j)$ with a constant N' then the marginal likelihood in Equation 3.1.29 results in a score-equivalent scoring-function [81]. Conversely, it can be shown that the assumption of score equivalence entails the prior distribution to be Dirichlet with $N'_S = N'p(j)$ [81].

The prior probabilities $p(j)$ of the various configurations $j \in \mathcal{I}(S)$ can, for instance, be determined by a Bayesian network specified a priori [81]. Often, such prior knowledge might not be available so that one might prefer the uninformative assignment

$$N'_S = \frac{N'}{|\mathcal{I}(S)|}, \quad (3.1.31)$$

where $|\mathcal{I}(S)|$ denotes the number of all joint states of the variables in S [19]. A similar assignment was chosen in [34]. However, the latter does not obey the normalization condition

$N' = \sum_{j \in \mathcal{I}(\mathcal{S})} N'_j = |\mathcal{I}(\mathcal{S})| N'_j$ for all $\mathcal{S} \subseteq \mathcal{V}$, as required by a score-equivalent scoring-function.

Posterior Probability

Let us now be concerned with the particular DAGs m^+ and m^- shown in Figure 3.1. Since the marginal likelihood in Equation 3.1.29 is clearly decomposable, the relative scoring function based on the posterior ratios (cf. Equation 3.1.27) reads

$$\begin{aligned} G_{\text{post}}(m^+, m^-) &= \frac{p(m^+ | D)}{p(m^- | D)} = \frac{p(m^+)}{p(m^-)} \frac{p(D | m^+)}{p(D | m^-)} \\ &= \frac{p(m^+)}{p(m^-)} \frac{\Omega(\mathcal{S} \cup \{a, b\})}{\Omega(\mathcal{S} \cup \{a\})} \frac{\Omega(\mathcal{S})}{\Omega(\mathcal{S} \cup \{b\})} \\ &= G_{\text{post}}(a, b, \mathcal{S}), \end{aligned} \quad (3.1.32)$$

where the function $\Omega(\cdot)$ is given by Equation 3.1.30. Again, many terms have canceled out, and G depends only on $a, b, \in \mathcal{V}$ and $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$, as typical for relative scoring functions (cf. Section 3.1.2). Various ways for assigning priors to the different network structures are discussed in [81]. A score-equivalent scoring-function is obtained when the same prior probability is assigned to equivalent DAGs, which is called *prior equivalence*. A simple choice, which also leads to decomposable priors, is the assignment

$$p(m) = e^{\beta |\mathcal{E}(m)| + \eta}, \quad (3.1.33)$$

where $|\mathcal{E}(m)|$ denotes the number of edges in the DAG m , and η is the normalization constant. The factor β can be chosen to penalize or favor DAGs a priori according to their number of edges. The assignment $\beta = 0$ leads to a uniform prior for the network structures. It is apparent that Equation 3.1.33 fulfills prior equivalence, since this prior is independent of the orientations of edges.

According to the Equations 3.1.32 and 3.1.33, the logarithmic relative score reads

$$\begin{aligned} g_{\text{post}}(a, b, \mathcal{S}) &= \log G_{\text{post}}(a, b, \mathcal{S}) \\ &= \beta + \log \frac{\Omega(\mathcal{S} \cup \{a, b\})}{\Omega(\mathcal{S} \cup \{a\})} \frac{\Omega(\mathcal{S})}{\Omega(\mathcal{S} \cup \{b\})} \end{aligned} \quad (3.1.34)$$

for the variables $a, b \in \mathcal{V}$ and the subset $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$. When the relative score $g(a, b, \mathcal{S})$ is positive then the presence of the edge $a \leftarrow b$ is favored given \mathcal{S} , whereas its absence is supported by $g(a, b, \mathcal{S}) < 0$ (cf. Section 3.1.2).⁸ In greedy algorithms, the relative score $g(a, b, \mathcal{S})$ is often compared to a threshold value γ which might differ from zero. A positive value of γ is often chosen in order to induce an edge $a \leftarrow b$ to be present given the set \mathcal{S} only if

⁸For simplicity, we assume that $g(a, b, \mathcal{S}) \neq 0$, i.e. the presence and the absence of an edge are not equally likely.

it is so to some notable degree (cf. also Table 5.5). A positive value of γ can also be viewed as an additional prior which penalizes DAGs with an increased number of edges, like the prior in Equation 3.1.33. Similar statements apply to negative threshold values. Thus, Equation 3.1.34 can alternatively be written like $\beta_{\text{new}} := \beta - \gamma$, implying the use of a vanishing threshold value $\gamma_{\text{new}} = 0$.

In the Bayesian approach, the parameters of a Bayesian network are usually estimated by averaging over the posterior distribution for the parameters, which are treated as random variables. In the approach taken above [34, 81], the estimate of the parameter concerning the variable $v \in \mathcal{V}$ in state $i \in \mathcal{I}(v)$ and the parents $\text{pa}(v)$ with the configuration $k \in \mathcal{I}(\text{pa}(v))$ is given by

$$\bar{\theta}_{v|\text{pa}(v)}(i, k) = \frac{N_{v,\text{pa}(v)}(i, k) + N'_{v,\text{pa}(v)}}{N_{\text{pa}(v)}(k) + N'_{\text{pa}(v)}}, \quad (3.1.35)$$

where $N'_{\text{pa}(v)}$ and $N'_{v,\text{pa}(v)} = N'_{\{v\} \cup \text{pa}(v)}$ are given by Equation 3.1.31. This expression is very similar to Equation 3.1.12 concerning maximum likelihood estimation. The only difference is that $N'_{\text{pa}(v)}$ and $N'_{v,\text{pa}(v)}$ act like pseudo observations in Equation 3.1.35. This fact entails a clear interpretation of these parameters which origin from the Dirichlet prior: the constant N (cf. Equation 3.1.31) is thus called *equivalent sample size* [81].

3.2 Model Uncertainty and Model Averaging

In general, there may not exist a single *true* model which describes the probability distribution implied by the data, while all the other models are useless. When given *finite* data, sampling noise or lack of enough data can entail that *several* Bayesian networks might describe the implied probability distribution about equally well. This is called *model uncertainty*. In order to allow for model uncertainty in structural learning, the learning algorithm has thus to be capable of finding possibly several DAGs rather than a single one. Accounting for model uncertainty has several benefits. First, when the various induced Bayesian network structures are considered, misleading interpretations can be avoided by taking into account the confidence in a certain structure. Second, when the induced Bayesian networks are used for predictions, the predictive accuracy can be improved by using an *average* of the predictions of *several* models, instead of relying on the prediction of a single one. When a single model is used for predictions then uncertainty might be underestimated [46].

There are various approaches to model averaging. The Bayesian approach is theoretically well-understood, and the posterior probability of a model can be interpreted as the probability of this model being the *true* one on the basis of the given data D . When accounting for model uncertainty, the prediction of a quantity of interest x , e.g. the value of some variables or the presence of a certain structure in the DAGs, is given by

$$p(x|D) = \sum_{m \in \mathcal{M}} p(x|m, D)p(m|D), \quad (3.2.1)$$

where the average is taken over *all* models. Apparently, this is a weighted average, and the posterior probabilities $p(m|D)$ play the role of the weights. Regarding the logarithmic scoring-rule [71], the following inequality shows the benefits of model averaging [108]:

$$\langle \log p(x|D) \rangle_{p(x|D)} = \langle \log \sum_{m \in \mathcal{M}} p(x|m, D) p(m|D) \rangle_{p(x|D)} \quad (3.2.2)$$

$$\geq \langle \log (p(x|m_o, D)) \rangle_{p(x|D)}, \quad (3.2.3)$$

where $\langle \cdot \rangle_{p(\cdot)}$ denotes the average with respect to $p(\cdot)$. This inequality indicates that averaging over all models improves the predictive accuracy compared to using a single $m_b \in \mathcal{M}$.

The number of all DAGs, however, is much too large in many applications (cf. Equation 3.3.1). Hence, one has to resort to averaging over a feasible number of models, typically about 10 or 20. Although the above inequality is not guaranteed to hold in this case, model averaging based on a small number of models appeared to improve the predictive accuracy in many experiments, e.g. [4, 18, 108]. Various strategies for selecting a tractable number of models have been proposed. For instance, one might choose only the high-scoring models, i.e. the ones whose posterior probability is larger than a certain threshold value. It is also popular to apply Occam's razor, which disregards those complex models which contain a submodel with a larger score. The latter two schemes can also be combined, which results in the so-called Occam's window [108]. Furthermore, *coherence rules* [62] are often employed in order to render learning algorithms more efficient when allowing for model uncertainty [48, 49, 108]. In the Bayesian approach, it is also popular to explore the posterior probability distribution for the model space by Markov chain Monte Carlo simulation [69, 72, 109, 110]. A recent variant is Markov chain Monte Carlo simulation applied to the ancestral ordering on the variables [59].

Although Bayesian model averaging theoretically leads to optimal predictive accuracy, it did not lead to a considerable improvement when combining several Bayesian networks in the experiment reported in [81]. In [44], it is argued that Bayesian model averaging does not overcome the problem of over-fitting, as the posterior probability depends exponentially on the cell counts implied by the data. The posterior probability can thus depend very sensitively on the sampling noise present in finite data. For this reason, a single model might be assigned a large posterior probability by chance, and the Bayesian model-average is dominated by the single model.

Model uncertainty can also be explored by the bootstrap [50], which was applied to learning Bayesian networks in [57, 58]. In this approach, a certain number l of replicate samples is generated by drawing N cases with replacement from a given data set of size N . From each of the l samples, a model is learned. The differences among the l models reflect model uncertainty. Although this scheme is simple in its nature, it is typically very time-consuming in practice, as the number l of bootstrap samples has usually to be chosen quite large in order to obtain reliable results. The l models can also be combined in order to improve predictive accuracy. In the scheme called bagging [16], the l models are combined with uniform weights. Similar schemes are stacking [152] and boosting [53]. Although these schemes are more or less heuristics, they led to less over-fitting than Bayesian model averaging in the experiments reported in [44].

3.3 Algorithms for Structural Learning

In structural learning in Bayesian networks, all variables are typically treated equally, i.e. there is no distinct variable such as the class variable, as opposed to the attribute variables in supervised learning. In the following, we are hence concerned with *unsupervised* learning, as the aim of structural learning in Bayesian networks is to approximate the *joint* probability distribution for all the variables in the domain, as implied by the data. All the scoring functions depicted in Section 3.1 are applicable. A scoring function used in structural learning in Bayesian networks is typically score-equivalent. Since it is impossible to distinguish among Markov-equivalent DAGs in this case (cf. Section 3.1.3), one can only aim at inducing the *equivalence class* rather than a particular DAG. Since learning algorithms applied to the search space of equivalence classes are, however, computationally rather inefficient [28, 107], one is typically concerned with the search space of DAGs.

Structural learning in Bayesian networks is computationally very involved. One reason is the extremely large search space of DAGs $m \in \mathcal{M}$. The number u of DAGs containing n variables can be calculated recursively according to [129]

$$u(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} u(n-i), \quad (3.3.1)$$

where $u(0) = 1$. Since an explicit expression for $u(n)$ is unknown, one can get an idea of the behavior of $u(\cdot)$ from the inequality

$$2^{\binom{n}{2}} < u(n) < 3^{\binom{n}{2}}. \quad (3.3.2)$$

The (loose) lower bound follows immediately from the fact that a DAG with n variables can contain at most $\binom{n}{2}$ edges, and each edge can either be present or absent when disregarding orientations. There are thus at least two alternatives for each edge. The upper bound follows in a similar fashion, as an edge can be absent, oriented in the one direction or in the other. This amounts to at most three alternatives for each edge. Since the acyclicity of DAGs imposes an additional restriction on the directions of edges, this can only be a loose upper bound, though. It is hence obvious that the number of DAGs rises super-exponentially with the number of variables (and exponentially with the number of edges). Moreover, it could be shown that finding the optimal DAG with respect to a scoring function is an NP-hard problem [14, 27, 84].

Due to the complexity of structural learning in Bayesian networks one has to resort to approximate learning algorithms which can find close-to-optimum solutions in a reasonable amount of time. During the last decade, two main approaches to structural learning in Bayesian networks have evolved. The one aims at maximizing a scoring function by means of a heuristic search strategy, while the constraint-based approach recovers the structure of a Bayesian network on the basis of the conditional independences and dependences induced from the data. Besides these two main approaches, there are also various other algorithms. Some of them are mentioned in the Sections 3.2 and 5.9. Recent reviews on structural learning in Bayesian networks are given in [20, 35, 131].

At first glance, it seems that the two main approaches might induce different DAGs from the same data. However, it can be shown that, when the data has been sampled from a faithful probability distribution and when the used scoring function obeys certain properties, both approaches determine the same DAG in the asymptotic limit, namely the perfect map [14]. Given *finite* data, however, it is not guaranteed that both approaches yield the same result.

3.3.1 Optimizing the Scoring Function

K2 Algorithm

The K2 algorithm [33, 34] has evolved from the Kutató algorithm [83]. It is the first algorithm employing a Bayesian scoring function in structural learning in Bayesian networks (cf. also Section 3.1.4). The aim of the K2 algorithm is to determine the DAG with the largest posterior probability, i.e. the optimum with respect to this scoring function.

Besides a data set and an upper bound on the number of parents of a variable in the induced DAG, the K2 algorithm requires also a correct (total) ordering, i.e. an ancestral ordering, on the variables as input. Since a DAG imposes only a partial ordering on the variables $v \in \mathcal{V}$, there typically exist many total orderings (cf. also Section 2.2.1). When given as input, a total ordering entails several advantages. First, the task of optimizing the scoring function is greatly simplified. This is because the optimal DAG with respect to the decomposable (absolute) scoring function $f(\cdot) = \sum_{v \in \mathcal{V}} f_v(v|\text{pa}(v))$ (cf. Equation 3.1.2) can be induced by optimizing each term $f_v(v|\text{pa}(v))$ *independently* of the others. The optimal DAG is then determined by the optimal parents $\text{pa}(v)$ of each variable $v \in \mathcal{V}$. Second, the total ordering ensures automatically that no directed cycles can occur in the induced DAG. Third, a *correct* ordering on the variables can also be expected to be beneficial regarding the quality of the induced DAGs, as it is additional knowledge given to the learning algorithm. This was indeed found in the experiments reported in [33, 34], where the induced DAG was very close to the original one, although the K2 algorithm uses a very simple search strategy to optimize the scoring function. Namely, starting out from the empty graph, that variable x is included into the current parent set $\text{pa}(v)$ of a variable v which is an ancestor of v according to the specified ordering and which increase the absolute score by most. In terms of the relative scoring function, the edge $v \leftarrow x$ with the largest score $g(v, x, \text{pa}(v)) > \gamma$ is included into the DAG (γ is a threshold value, cf. Section 5.4.5). Due to the greedy nature of this search strategy, the K2 algorithm tends to include slightly more edges than optimal, as found in experiments [33, 34].

This search strategy is called *forward inclusion*. In contrast, the scheme named *backward elimination* starts out with the complete graph, and successively removes edges. The latter approach is, however, not applicable to Bayesian networks, as the complete graph entails such a large model complexity that the learning process becomes intractable.

The results of the K2 algorithm can depend sensitively on the ordering initially specified [81]. Since such an ordering is usually unknown in real-world applications, the K2 algorithm can usually be applied only in experiments with artificial data where a correct ordering is known.

When a correct ordering is given to the K2 algorithm, it can find better DAGs than other more sophisticated search strategies like greedy local search or local search combined with simulated annealing, regarding structural differences between the induced DAG and the original Bayesian network structure [81].

Local Search

Local search is a general-purpose search strategy [90]. Starting out from a DAG initially specified, this scheme proceeds through the search space of DAGs \mathcal{M} in a sequence of *neighboring* DAGs $m \in \mathcal{M}$. There are two apparent definitions of a neighborhood of a DAG. First, a DAG m belongs to the neighborhood of a DAG m_o if m and m_o are identical except for the presence or absence of a single edge. An increased neighborhood of a DAG m_o contains all those DAGs which differ from m_o concerning the presence, absence or orientation of a single edge. According to either definition, there exists obviously a sequence of neighboring DAGs between any two DAGs $m_1, m_2 \in \mathcal{M}$. This ensures that local search can reach any DAG within the search space. In practice, the latter definition of a neighborhood appeared to yield better results, in particular when employed in a greedy scheme.

Given the current DAG m_o in the search process, there are many neighboring DAGs which can become the next intermediate graph m'_o . The difference in their scores, $\Delta := f(m'_o) - f(m_o)$, can be expressed in terms of the relative scoring function. If an edge $a \leftarrow b$ is included in the transition from m_o to m'_o then $\Delta = g(a, b, \text{pa}(a) \setminus \{b\})$. Conversely, the removal of an edge $a \leftarrow b$ entails $\Delta = -g(a, b, \text{pa}(a) \setminus \{b\})$, and the reversal of an orientation (from $a \leftarrow b$ to $a \rightarrow b$) results in $\Delta = g(b, a, \text{pa}(b) \setminus \{a\}) - g(a, b, \text{pa}(a) \setminus \{b\})$. Of course, it has to be ensured that no directed cycles occur, i.e. that the next graph is indeed a DAG. In greedy hill climbing, that DAG m'_o becomes the next intermediate graph which entails the largest improvement $\Delta > \gamma$ compared to the current DAG m_o (γ is a threshold value, cf. Section 5.4.5).

The main problem of local search is that it can get stuck at local optima of the (absolute) scoring function. A DAG m_o is a local optimum when all its neighboring DAGs m are assigned a smaller score, i.e. $f(m) - f(m_o) \leq 0$. There are various approaches to overcome this problem, e.g. [97, 108]. Let us focus on simulated annealing [112], as it was applied to structural learning in Bayesian networks in [81]. As implied by its name, the idea stems from cooling down a physical system, and the aim is to reach the global minimum of the *potential* governing this system. In our case, the negative of the scoring function, $-f(\cdot)$, plays the role of such a potential. An ensemble of classical particles which is subject to this potential has a Boltzmann distribution $p(\cdot) \propto \exp(f(\cdot)/T)$, where T denotes the temperature of the system. It is apparent, that the number of particles at a small values of $-f(\cdot)$ is larger than at high values of $-f(\cdot)$. Moreover, as expected from common sense, the number of particles at small values of the potential $-f(\cdot)$ increases as the temperature T drops.

The procedure of simulated annealing starts out with a quite large temperature T_o which is then gradually decreased down to a final temperature $T_f \approx 0$. At an intermediate temperature

T ($T_o > T > T_f$), this method picks randomly a neighboring DAG m'_b of the current graph m_o . This is called a proposed change. The transition to m'_b is made with the probability $p(\Delta) = \min\{1, \exp(\Delta/T)\}$. Hence, a change improving the score is always carried out, whereas a decrease in the score is accepted or rejected with a finite probability. This procedure is repeated several times at a given temperature T . In the experiments in Section 6.3.2, we chose to stay at the same temperature until 600 transition to neighboring DAGs were proposed or 200 transitions were accepted, whatever occurred first. After that, the temperature T was decreased to T' according to $T' = \omega \cdot T$, $\omega < 1$, and the above steps were repeated at the new temperature. It can be shown that, when a sufficiently small cooling rate $\omega < 1$, i.e. $\omega \approx 1$, is chosen, simulated annealing eventually arrives with probability one at the global minimum of the potential, $-f(\cdot)$, and hence at the global optimum of the scoring function $f(\cdot)$. Despite this theoretical result simulated annealing is, however, not guaranteed to find the global optimum in practical applications, since computation time is limited. This procedure is indeed quite time-consuming. Nevertheless, it usually finds DAGs with a higher score than greedy hill-climbing. Note that the latter can be viewed as special case of simulated annealing where the temperature is chosen to be zero.

3.3.2 Constraint-Based Approach

The constraint-based approach was originally used in a causal discovery setting, e.g. [140–142]. In contrast to local search, the constraint-based approach makes use of a particular property of graphical models, namely the conditional independences and dependences (CIDs) represented by the graph. For this approach to work, it is essential that the CIDs can be induced without any error from the probability distribution implied by the data. Moreover, it is assumed that this probability distribution is faithful (cf. Section 2.2.4). If these assumptions hold, the constraint-based approach has been proven to recover the correct DAG, namely the perfect map (up to equivalence, of course) [140, 148]. The constraint-based approach does not suffer from getting stuck at local optima, unlike the search strategies aimed at optimizing a scoring function. For the same reason, equivalent DAGs are not a particular problem for constraint-based algorithms.

This approach appeared to be very efficient when recovering *sparse* graphs, since only a reasonably small number of tests had to be carried out in order to find the relevant independences [142]. In the worst case, i.e. when the induced graph is very dense, the complexity of this approach is, of course, exponential in the number of variables (cf. [142] and the Sections 5.5 and 5.4.2). Domains which require the induced DAG to be dense, however, might better not be modeled by Bayesian networks, as neither learning nor inference are tractable, and the interpretation of dense Bayesian network structures is difficult, too. This was discussed in Section 1.1. Since statistical tests carried out on *finite* data sets can entail Type I and Type II errors, it is clear that the assumptions underlying the constraint based approach cannot be expected to hold in this case. For this reason, when the constraint-based approach combines the results of the various independence tests in order to construct the DAG, the overall error, i.e. the one concerning the entire graph, is not under control. The reason is that the various independence

tests may not be independent of each other. This is a major problem in the area of multiple testing. The remaining chapters are concerned with this issue, and propose extensions to the constraint-based approach when applied to finite data sets.

SGS Algorithm

input: data D regarding the set of variables \mathcal{V} .

output: the pattern of the induced equivalence class.

- (1) compute the CIDs given the data D ,
- (2) induce the presence of the edges on the basis of the CIDs:
 - (i) begin with the complete skeleton (undirected graph),
 - (ii) for each pair of variables a and b , if there exists a set $S \subseteq \mathcal{V} \setminus \{a, b\}$ such that $a \perp\!\!\!\perp b \mid S$ then remove the edge $a \sim b$ from the current skeleton.
- (3) induce the orientations of the edges given the current graph and the CIDs:
 - (i) find the colliders in the current graph:

for each triple of vertices a , b and c such that the edges $a \sim b$ and $b \sim c$ are present while the edge between a and c is absent ($a \not\sim c$), orient $a \sim b \sim c$ as $a \rightarrow b \leftarrow c$ iff $a \not\perp\!\!\!\perp b \mid S$ for all subsets $S \subseteq \mathcal{V} \setminus \{a, c\}$ with $b \in S$,
 - (ii) orient the remaining edges in the current graph:

repeat until no more edges can be oriented:

 - if $a \rightarrow b \sim c$ and $a \not\sim c$ then orient $b \sim c$ as $b \rightarrow c$,
 - if $a \sim b$ and there exists a directed path $a \rightarrow \dots \rightarrow b$ then orient $a \sim b$ as $a \rightarrow b$.

Algorithm 3.1: The SGS algorithm [141] clearly illustrates the basic scheme underlying all constraint-based approaches, as it is the simplest constraint-based algorithm.

Let us now review popular constraint-based algorithms. When the above assumptions hold, the constraint-based approach can be split up into successive steps. The first one is concerned with determining the CIDs from the given data, while the second one constructs the DAG on the basis of those CIDs. This is most apparent in the SGS algorithm [141], see Algorithm 3.1. Having induced the CIDs from the data, e.g. by means of a relative scoring function and a threshold value, the perfect map is constructed in two steps. The first step is concerned with the *presence* of edges, and utilizes the fact that a conditional independence $a \perp\!\!\!\perp b \mid S$ in the probability distribution entails the d-separation $a \perp\!\!\!\perp b \mid S$ in the perfect map (cf. Section 2.2.4). Since this step disregards the orientations, an *undirected* graph, i.e. the so-called skeleton of a

DAG, is used. Having recovered the skeleton, the *orientations* of the edges are determined in the final step, which is again subdivided. First, the colliders are found, which determines the equivalence class of the perfect map (cf. Section 2.2.3). Figure 2.2 illustrates that a collider can be distinguished from non-colliders by means of CIDs, since conditioning on a collision node entails a *d-connection*, whereas conditioning on a non-collision-node can entail a *d-separation* (cf. also Section 2.2.1). After that, the remaining edges are oriented with the aim to avoid additional colliders. In order to avoid directed cycles in this step, the last edge before a cycle is completed is simply oriented in the contrary direction. The output of the SGS algorithm is a pattern of the equivalence class (cf. Section 2.2.3 and Figure 2.1).

It is apparent that the SGS algorithm is very inefficient in domains with a large number of variables $n = |\mathcal{V}|$, as the number of different subsets $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ grows exponentially with n for each pair of variables a and b . The PC algorithm [142] is an algorithm which can reduce this complexity tremendously. It is one of the most popular constraint-based approaches, and it is available in the TETRAD software-package [147]. The increased efficiency of the PC algorithm is achieved by carrying out the steps (1) and (2) of the SGS algorithm simultaneously: if a conditional independence is induced then the corresponding edge is immediately removed from the current skeleton. The current skeleton is then used for determining the next conditional independences to be tested. This is done by restricting the subset $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ such that it is a subset of the neighbors of a or b in the current graph, i.e. $\mathcal{S} \subseteq \text{ne}(a) \setminus \{b\}$ or $\mathcal{S} \subseteq \text{ne}(b) \setminus \{a\}$. The possible choices in the subsets \mathcal{S} are thus greatly reduced, and so is the number of independence tests which have to be carried out. It can be shown that the PC algorithm induces the same DAG as the SGS algorithm provided that the assumptions essential for the constraint-based approach hold [140]. However, if these assumptions are violated then the results might differ. A disadvantage of the PC algorithm is in this case that its result can depend on the sequence in which it proceeds through the variables in \mathcal{V} , as the current structure of the graph has an impact on the subsets \mathcal{S} to be considered next.

In the PC algorithm, the independence tests are carried out in ascending order of the set \mathcal{S} . This increases its reliability, since tests of lower orders are generally more reliable than tests of higher orders. Moreover, the number of possible subsets $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ ($a, b \in \mathcal{V}$) is considerably smaller at small orders than at high orders of \mathcal{S} , and hence leads to an additional speed-up in the computations.

The various constraint-based approaches mainly differ regarding the heuristics applied to reduce the number of independence tests carried out. Furthermore, various kinds of statistical independence-tests can be employed (see e.g. [47, 151]). An early version of the PC algorithm restricted the variables of the subset \mathcal{S} to be not only among the neighbors of a or b but also to lie on undirected paths between the two variables [140]. Not much attention has been paid to this scheme, as it was found to be intractable in practical applications, except for domains with a very small number of variables. Furthermore, it was regarded to be less robust than the PC algorithm in practice, because the Type I and Type II errors of independence tests increased chances that an edge which was "erroneously" removed at an early stage of the learning process entailed other edges to be "erroneously" present [142].

An algorithm somewhat similar to the PC algorithm was proposed independently in [61]. The tree algorithm of Chow and Liu [32] is the basis of a very efficient constraint-based algorithm [23, 24], and a so-called branch-and-bound technique employing the minimum description length principle was developed in [146].

Also hybrid algorithms were proposed, combining the two main approaches to structural learning in Bayesian networks. For instance, the PC algorithm can be employed to induced an ancestral ordering on the variables which is subsequently used by the K2 algorithm when inducing a Bayesian network structure [138].

4

Properties of Optimal Structures

In the previous chapters, Bayesian networks and various approaches to structural learning were introduced. We pointed out that the search space grows super-exponentially with the number of variables (cf. Section 3.3), and that finding the optimal DAG with respect to a scoring function is an NP-hard problem [14, 27, 84]. The tractability of structural learning algorithms has thus to be of major concern in practice. The constraint-based approach was reported to be very efficient in many experiments [23, 24, 142]. It is well-understood only when certain assumptions are fulfilled. In many applications, however, these assumptions can only be expected to hold in the asymptotic limit, i.e. when an infinite amount of data is available. Nevertheless, the experimental results obtained from *finite* data sets are very promising [23, 24, 142].

Based on the various scoring functions outlined in Section 3.1, in this chapter the constraint-based approach is viewed as a particular search strategy aimed at optimizing such a scoring function. This point of view reveals important properties of the graphs induced by the constraint-based approach. In particular, this chapter is concerned with the *presence or absence* of edges in optimal DAGs and their skeletons, while the *orientations* of edges are considered in Chapter 6. The theoretical considerations lead to an extension of the constraint-based approach which can considerably improve the results obtained from finite data sets. This extension is based on the so-called necessary path condition (cf. Section 4.3.1, in particular Proposition 4.12). It applies to various kinds of "paths", among which four variants are presented:

- the si-1-path, involving the sc-1-path (cf. the Definitions 4.11 and 4.9),
- the si-2-path, involving the sc-2-path (cf. the Definitions 4.14 and 4.13),
- the si-3-path (cf. Definition 4.15), and
- the si-4-path (cf. Definition 4.16).

These variants differ in their strictness, e.g. the si-2-path requires the most edges and paths to be present, while the si-4-path accounts for the fewest paths among the alternatives presented. This chapter lays the foundations of the structural learning algorithm (and its variants) described in the next chapter.

4.1 Motivation

In Section 3.3.2, it was discussed that constraint-based algorithms can essentially be divided into two steps, namely where

- (i) the conditional independences and dependences (CIDs) are induced from the probability distribution implied by the data, and where
- (ii) the DAG (perfect map) is determined on the basis of the induced CIDs.

In the following, we are concerned with the two assumptions important to the constraint-based approach to work. First, the CIDs have to be perfectly known, i.e. without any error. This ensures that the true CIDs are induced in step (i). Second, the probability distribution has to be *faithful*, i.e. all the (true) CIDs can be represented in a *single* DAG, the perfect map. Hence, if both assumptions hold then the *induced* CIDs can be represented in the perfect map. This simplifies the learning task greatly, since an *independence* of two variables a and b conditional on *some* set S entails the absence of the edge between a and b in the perfect map. The absence of the edge between a and b thus depends on the set S only, and it is independent of the remaining network structure. This means that the absence of an edge is only affected by a certain *vicinity* of that edge in the perfect map, i.e. the variables in S (cf. Figure 4.1). Since the remaining variables outside such a vicinity can be disregarded, the learning task has hence become *local*. The presence or absence of each edge can thus be determined on the basis of its vicinity, *independently* of the other variables and edges in the graph. For this reason, the assumption of faithfulness renders the constraint-based approach to be computationally efficient, as confirmed in many experiments [23, 24, 142].

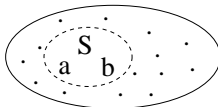


Figure 4.1: The edge absent between a and b , and its vicinity comprising only the variables in S . The dots symbolize the remaining network structure.

However, if the assumption of faithfulness does not hold, the presence of an edge (and its orientation, of course) can depend on the exact structure of the *entire* DAG. This *non-locality* causes structural learning to be very involved in general.

Regarding faithfulness, an interesting result was derived in [111]. Roughly speaking, it says that almost all probability distributions described by Bayesian networks are faithful. This is meant in a measure-theoretic sense, namely that, in the space of all probability distributions which can be described by a Bayesian network, the subspace of unfaithful probability distributions has a vanishing Lebesgue measure. This result suggests that the assumption of faithfulness might not impose a severe restriction on the constraint-based approach. However, note that this claim refers only to probability distributions which can actually be described by Bayesian networks.

Having mentioned this result concerning faithfulness, let us now consider the other assumption, saying that the CIDs are perfectly known, i.e. that they can be induced without any error. In typical applications, one can expect this assumption to hold only *in the asymptotic limit* when the data has been sampled from a faithful probability distribution. Given *finite* data, however, this assumption may not be reasonable. The reason is that, given a finite sample, CIDs have to be induced by means of some statistical tests. In (classical) statistics, it is well known that statistical tests entail Type I and Type II errors. An alternative point of view is as follows. A conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$ is typically derived by means of the relative score $g(a, b, \mathcal{S}) < 0$ (cf. Section 3.1). In classical statistics as well as in Bayesian statistics, such relative scoring functions contain a term penalizing model complexity. This is essential in order to induce the results with some *significance* or to avoid *over-fitting*, whatever point of view one may prefer. Hence, it can occur that a "weak" dependence between two variables in the probability distribution is disregarded by such a decision mechanism solely due to the penalty term in the scoring function dominating over the "weak" dependence. As a result, a conditional independence is induced. This shows that, even when the CIDs are derived from a faithful probability distribution, the number of *induced* independences can be larger than the number of *true* independences. As a consequence, there might not exist a perfect map of the *induced* CIDs, even if there is a perfect map of the *true* CIDs. This may occur already when the probability distribution implied by a finite data set is *nearly* unfaithful (for instance, measured by the Kullback-Leibler divergence). It is intuitively clear that, in the space of all probability distributions which can be described by Bayesian networks, the subspace of nearly unfaithful probability distributions cannot be expected to have a vanishing Lebesgue measure. Hence, chances can be quite high that there is no perfect map corresponding to the CIDs induced from a finite sample. Consequently, there does not exist a perfect map of the induced CIDs in general, and learning from finite samples becomes a non-local problem. The locality entailed by the faithfulness assumption can hence only be considered as an approximation to the exact learning task, as the latter is indeed non-local.

A typical example we encountered quite often in our experiments (cf. Section 5.6) is depicted in Figure 4.2. Due to the rather small sample size, the relative scores of increased orders – involving an increased penalty for model complexity – tend to become negative, and hence imply conditional independences. The DAG m_0 is induced by the constraint-based approach assuming the existence of a perfect map. As it removes each edge for which a conditional independence was found, only the edge between a and b is left in the graph. It is apparent that the DAG m_0 is not the perfect map of the induced CIDs, as the graph implies the marginal independences $a \perp\!\!\!\perp c$ and $b \perp\!\!\!\perp c$, although these variables are induced to be marginally dependent, i.e. $a \not\perp\!\!\!\perp c$ and $b \not\perp\!\!\!\perp c$. The induced DAG m_0 is a D-map of the induced CIDs in this example,¹ since every induced conditional independence entails the absence of the corresponding edge.

If there does not exist a perfect map, a reasonable aim in structural learning is to find those DAGs which are *optimal* with respect to the used (absolute) scoring function f . Note that the latter can also be expressed in terms of the relative scoring function g (cf. Equation 3.1.7). It is

¹This does not hold in general. For instance, when a DAG is induced from the CIDs represented by the Markov network in Figure 2.5 the acyclicity of the DAG entails an additional conditional dependence.

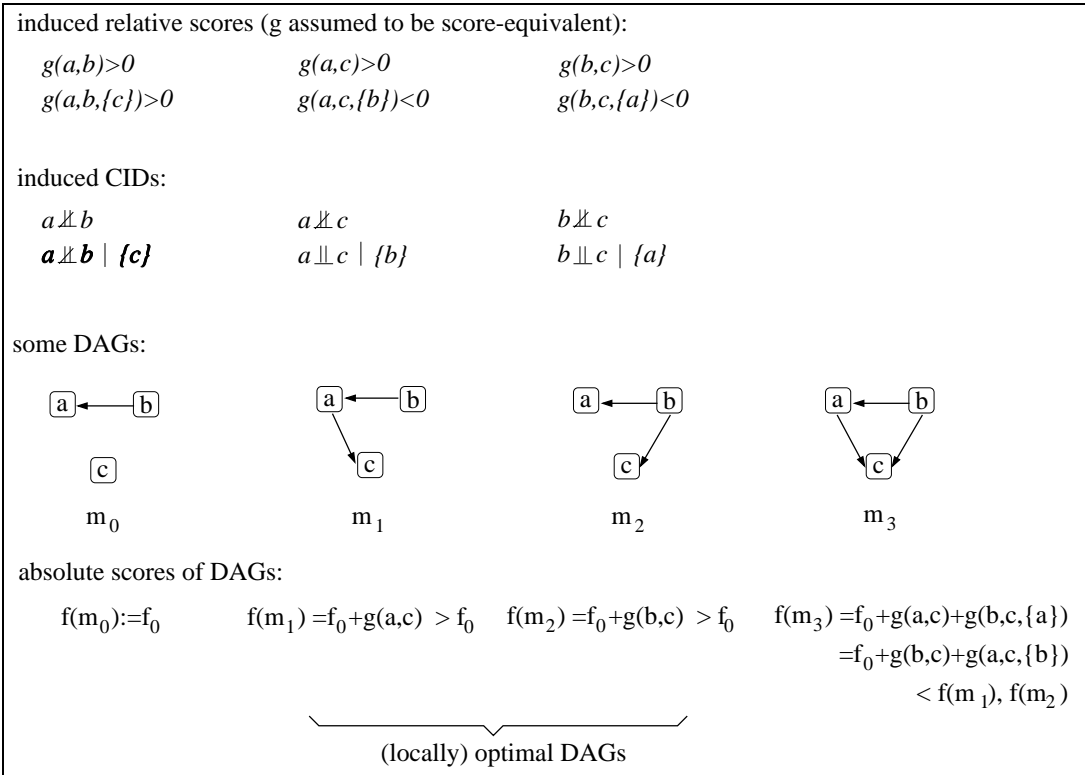


Figure 4.2: In this example, the induced conditional independences and dependences (CIDs) cannot be represented by a perfect map.

apparent that the highest scores are assigned to the DAGs m_1 and m_2 in Figure 4.2, which are hence local optima. Compared to the DAG m_0 , each of the local optima apparently contains an *additional* edge. Moreover, there are *two* local optima rather than a *single* perfect map, indicating some model uncertainty regarding the structure of Bayesian networks. Furthermore, the DAGs m_1 and m_2 are minimal *I*-maps rather than D-maps, i.e. *all* the induced (conditional) *dependences* are represented in the graph, whereas some of the found independences cannot be read off these DAGs (cf. also Section 2.2.4).

The coincidence of *optimal* DAGs and minimal I-maps is not completely unexpected in this example. Roughly speaking, it can be shown that minimal I-maps m are given higher scores $f(m)$ than non-minimal-I-maps in the asymptotic limit if no perfect map exists [14]. This indicates that, if no perfect map exists, the DAGs induced from finite samples by established constraint-based approaches are usually not very close to optimal DAGs, as maximal D-maps typically contain fewer edges than minimal I-maps. Only in the case where the maximal D-map coincides with the minimal I-map the constraint-based approach induces the optimal DAG, which coincides with the perfect map.

This motivates to extend the constraint-based approach to *finite* data with the following aims:

- drop the assumption of faithfulness,
- aim at inducing (locally) *optimal* DAGs, i.e. with respect to a scoring function,
- employ relative scoring functions (other than the one in the χ^2 -test),
- induce additional edges compared to state-of-the-art constraint-based approaches,
- model uncertainty: allow for possibly *several* graphs to be induced, instead of a single DAG assumed to be the perfect map.

The concepts of relative and absolute scoring functions were introduced in Section 3.1. We noted that the χ^2 -test – which is commonly applied by constraint-based approaches – can be denoted in a similar fashion as relative scoring functions. However, there does not exist a corresponding absolute scoring function. For this reason, we prefer relative scoring functions derived from an absolute one, like for instance the Akaike Information Criterion, the Bayesian Information Criterion or the posterior probability (cf. Section 3.1). While scoring functions are related to step (i) of the constraint-based approach, step (ii) has to be concerned with the other four issues stated above. The second step is typically further subdivided, namely into two parts, where the first one is concerned with determining the presence of edges (in skeletons), while the second one finds the orientations of the edges. In this thesis, we mainly focus on the first part dealing with skeletons (cf. the Chapters 4 and 5). The remainder of this chapter develops the notion of the necessary path condition and focuses on the properties of the graphs induced by the constraint-based approach.

4.2 Properties of Optimal DAGs

In this section, we derive a property of *optimal* Bayesian network structures concerned with the *presence* of edges and paths. Note that the assumption of faithfulness is not used in the following. The main result of this section is Proposition 4.8. The starting point of its derivation is the following fact about the presence of an edge in a DAG being a (local) optimum with respect to a scoring function.

Proposition 4.1 *Let g be a relative scoring function, and let m be a (locally) optimal Bayesian network structure. A (directed) edge $a \leftarrow b$, $a, b \in \mathcal{V}$, is present in m if and only if*

- (i) $g(a, b, \text{pa}_m(a) \setminus \{b\}) > 0$,² and
- (ii) m contains no directed path $a \rightarrow x_1 \rightarrow \dots \rightarrow x_{r-1} \rightarrow b$, $x_i \in \mathcal{V} \setminus \{a, b\}$, from a to b , and
- (iii) $g(a, b, \text{pa}_m(a) \setminus \{b\}) > g(b, a, \text{pa}_m(b) \setminus \{a\})$ or a directed path from b to a is present.

²For simplicity, we assume that $\forall a, b \in \mathcal{V}, \forall \mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\} : g(a, b, \mathcal{S}) \neq 0$, as the scoring function would favor neither the presence nor the absence of the edge $a \leftarrow b$ in the case $g(a, b, \mathcal{S}) = 0$. For simplicity, we do not consider this source of uncertainty regarding the presence of edges.

Again, \mathcal{V} denotes the set of variables in a domain, and $\text{pa}_m(a)$ are the parents of a variable $a \in \mathcal{V}$ in the DAG m .

Proof. The neighboring DAGs of a local optimum differ from the latter concerning a single directed edge being either added, removed or reversed. In (i), it is stated that the presence of the edge $a \leftarrow b$ is favored more than its absence. Statement (ii) ensures that the graph does not contain a directed cycle when the edge $a \leftarrow b$ is present, as required in a DAG. Statement (iii) is concerned with the optimality of the orientation of the edge, as it ensures that the edge with the contrary orientation, i.e. the edge $a \rightarrow b$, is favored less than the orientation $a \leftarrow b$. This is given when a larger score is assigned to the latter, or when the edge cannot be oriented like $a \rightarrow b$, since a directed cycle has to be avoided. \square

In the following, we are mainly interested in the *presence* of an edge as opposed to its *absence*, while its orientation is of minor interest. Let us hence denote an edge between two variables $a, b \in \mathcal{V}$ by $a \sim b$ when we are concerned with its *presence* independent of its orientation in the DAG. An edge $a \sim b$ is called present in the DAG if either the edge $a \rightarrow b$ or $a \leftarrow b$ is present. Conversely, when we say that the edge $a \sim b$ is absent from the DAG, it means that neither one of the edges $a \rightarrow b$ and $a \leftarrow b$ is present. Naturally, this notation is symmetrical with respect to a and b , and it is identical with the one concerning skeletons.

As a special case of Proposition 4.1, let us first consider edges whose inclusion into *any* DAG m leads to an improvement in the (absolute) scoring function f . It is clear that the presence of such edges is a necessary requirement for a DAG to be a local optimum. Let us hence call them *certainly-present* edges. Of course, there might also be edges whose elimination from any DAG m leads to an increase in the score $f(m)$. As such edges have to be absent from a local optimum, and we call them *certainly-absent* edges. Proposition 4.1 yields immediately:

Proposition 4.2 (Certainly-Present or Certainly-Absent Edges) *An edge $a \sim b$ is certainly present in an optimal DAG m or in its skeleton \tilde{m} if $\forall S \subseteq \mathcal{V} \setminus \{a, b\} : g(a, b, S) > 0 \wedge g(b, a, S) > 0$. Conversely, an edge $a \sim b$ is certainly absent in an optimal DAG m if $\forall S \subseteq \mathcal{V} \setminus \{a, b\} : g(a, b, S) < 0 \wedge g(b, a, S) < 0$.³*

Proof. We only note that the acyclicity of a DAG does not prevent a certainly-present edge $a \sim b$ from being indeed present in any DAG. This is because the presence of $a \leftarrow b$ or $a \rightarrow b$ could only be forbidden when both the directed paths $a \rightarrow \dots \rightarrow b$ and $b \rightarrow \dots \rightarrow a$ were simultaneously present. This is, however, forbidden by the acyclicity of DAGs. \square

Certainly-present edges are typically the only ones found by those constraint-based approaches which take into account *all* relative scores, e.g. the SGS algorithm [142]. In practice, however, it is usually infeasible to compute all the scores, and the graphs are determined on the basis of a limited number of scores, e.g. like in the PC algorithm [142]. Since the number of computed scores is (heuristically) reduced, additional edges can be present in the induced graphs.

³When the scoring function is score-equivalent, as it is often the case, then $g(a, b, S) = g(b, a, S)$ (cf. Section 3.1.3), leading to a great simplification.

Besides certainly-present or certainly-absent edges, there are generally also other sorts of edges in an optimal DAG. In fact, these additional edges can play a crucial role in optimal DAGs. For such an edge $a \sim b$ ($a, b \in \mathcal{V}$) holds that

- $g(a, b, \mathcal{S}) < 0$ or $g(b, a, \mathcal{S}) < 0$ given *some* sets $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$, and
- $g(a, b, \mathcal{S}') > 0$ or $g(b, a, \mathcal{S}') > 0$ given *some other* sets $\mathcal{S}' \neq \mathcal{S}$, $\mathcal{S}' \subseteq \mathcal{V} \setminus \{a, b\}$.

Because some scores are positive while others are negative, it is not clear at first glance if such an edge is present or absent in an optimal DAG. Let us hence focus on this kind of edges in the remainder of this section. Concerning their presence, Proposition 4.1 yields immediately:

Proposition 4.3 *Let g be a relative scoring function, and let m be a (locally) optimal DAG. The edge $a \sim b$, i.e. $a \leftarrow b$ or $a \rightarrow b$, is present in m if and only if*

- (i) $g(a, b, \text{pa}_m(a) \setminus \{b\}) > 0$, and
- (ii) *there is no directed path from a to b in m , and*
- (iii) $g(a, b, \text{pa}_m(a) \setminus \{b\}) > g(b, a, \text{pa}_m(b) \setminus \{a\})$ *or a directed path from b to a is present,*
*or*⁴
- (iv) $g(b, a, \text{pa}_m(b) \setminus \{a\}) > 0$, and
- (v) *there is no directed path from b to a in m , and*
- (vi) $g(a, b, \text{pa}_m(a) \setminus \{b\}) < g(b, a, \text{pa}_m(b) \setminus \{a\})$ *or a directed path from a to b is present.*

This proposition only restates Proposition 4.1 for *both* the orientations $a \leftarrow b$ and $b \leftarrow a$. Since constraint-based algorithms focus on the absence of edges rather than on their presence, let us derive a proposition concerning the absence of edges. This can be achieved by negating Proposition 4.3, leading to the following

Proposition 4.4 *Let g be a relative scoring function, and let m be a (locally) optimal Bayesian network structure. The edge $a \sim b$, $a, b \in \mathcal{V}$, is absent in m if and only if*

- (i) $g(a, b, \text{pa}_m(a) \setminus \{b\}) < 0$ *and* $g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$,⁵ *or*
- (ii) $g(a, b, \text{pa}_m(a) \setminus \{b\}) < 0$ *and m contains a directed path $b \rightarrow x_1 \rightarrow \dots \rightarrow x_{r-1} \rightarrow a$, $x_i \in \mathcal{V} \setminus \{a, b\}$, from b to a , or*
- (iii) $g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$ *and m contains a directed path $a \rightarrow x_1 \rightarrow \dots \rightarrow x_{r-1} \rightarrow b$, $x_i \in \mathcal{V} \setminus \{a, b\}$, from a to b .*

⁴This notation means $((i) \wedge (ii) \wedge (iii)) \vee ((iv) \wedge (v) \wedge (vi))$.

⁵While the identities $\text{pa}_m(a) \setminus \{b\} = \text{pa}_m(a)$ and $\text{pa}_m(b) \setminus \{a\} = \text{pa}_m(b)$ hold in a DAG m without the edge $a \sim b$, the inequalities $\text{pa}_m(a) \setminus \{b\} \neq \text{pa}_m(a)$ and $\text{pa}_m(b) \setminus \{a\} \neq \text{pa}_m(b)$ are important in Proposition 4.3. The longer expressions are thus used for clarity.

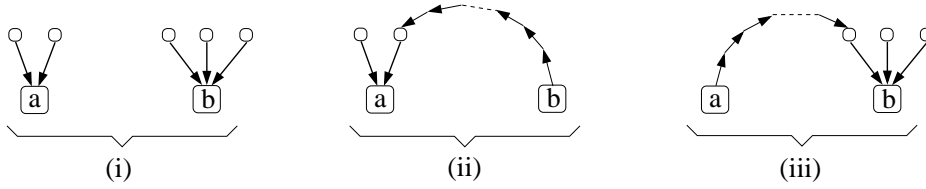


Figure 4.3: If the edge $a \sim b$ is absent then one of the three structures (i), (ii) or (iii) has to be present in an optimal DAG according to Proposition 4.4.

Proof. Let the indices without a prime refer to the ones in Proposition 4.3, and the ones with a prime to this proposition. Moreover, let us denote the negation of a statement (i) as $\overline{(i)}$, and so on. The negation of Proposition 4.3 can then be simplified according to the laws of De Morgan,

$$\begin{aligned}
 & \overline{((i) \wedge (ii) \wedge (iii)) \vee ((iv) \wedge (v) \wedge (vi))} \\
 = & \overline{(i)} \vee \overline{(ii)} \vee \overline{(iii)} \wedge (\overline{(iv)} \vee \overline{(v)} \vee \overline{(vi)}) \\
 = & \underbrace{\overline{(i)} \wedge \overline{(iv)}}_{(i)'} \vee \underbrace{\overline{(ii)} \wedge \overline{(v)}}_{(ii)'} \vee \underbrace{\overline{(i)} \wedge \overline{(vi)}}_{(A)} \vee \underbrace{\overline{(ii)} \wedge \overline{(iv)}}_{(iii)'} \vee \underbrace{\overline{(ii)} \wedge \overline{(v)}}_{(B)} \\
 & \vee \underbrace{\overline{(ii)} \wedge \overline{(vi)}}_{(C)} \vee \underbrace{\overline{(iii)} \wedge \overline{(iv)}}_{(\tilde{A})} \vee \underbrace{\overline{(iii)} \wedge \overline{(v)}}_{(\tilde{C})} \vee \underbrace{\overline{(iii)} \wedge \overline{(vi)}}_{(D)} \\
 = & (i)' \vee (ii)' \vee (iii)'.
 \end{aligned}$$

The various simplification steps are in detail:

- Expression (A) can be omitted, as it is stricter than $(i)'$, and hence $(A) \vee (i)' = (i)'$. The reason is that, in (A), $g(a, b, \text{pa}_m(a) \setminus \{b\}) < 0 \wedge g(a, b, \text{pa}_m(a) \setminus \{b\}) > g(b, a, \text{pa}_m(b) \setminus \{a\})$ implies $g(a, b, \text{pa}_m(a) \setminus \{b\}) < 0 \wedge g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$. The latter is equivalent to statement $(i)'$.
- For the same reason, also expression (\tilde{A}) can be dropped, as it is stricter than $(i)'$, too.
- Statement (B) can be eliminated, since it cannot be fulfilled, as a directed cycle $a \rightarrow \dots \rightarrow b \rightarrow \dots \rightarrow a$ cannot occur in a DAG.
- The statements (C) and (\tilde{C}) are always false, since a path cannot be present and absent at the same time. Hence, also (C) and (\tilde{C}) can be omitted.
- Also statement (D) can be dropped, because it forbids the existence of a directed path, and hence also of an edge, in either direction between a and b .

□

The conditions (i), (ii) and (iii) in Proposition 4.4 are illustrated in Figure 4.3. The statements (ii) and (iii) are identical, except for the two variables a and b being swapped. In contrast,

statement (i) does not require a (directed) path being present, but it requires a negative score regarding *both* the variables a and b .

As a short remark, let us reconsider Proposition 4.3, because it can be rewritten in a less complicated fashion due to the simplifications which led to Proposition 4.4. The following proposition is equivalent to Proposition 4.3:

Proposition 4.5 *Let g be a relative scoring function, and let m be a (locally) optimal DAG. The edge $a \sim b$, i.e. the edge $a \leftarrow b$ or the edge $a \rightarrow b$, $a, b \in \mathcal{V}$, is present in m if and only if*

$$(i) \quad g(a, b, \text{pa}_m(a) \setminus \{b\}) > 0, \text{ and}$$

$$(ii) \quad m \text{ does not contain a directed path } a \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1} \rightarrow b, \quad x_i \in \mathcal{V} \setminus \{a, b\}, \text{ from } a \text{ to } b,$$

or

$$(iii) \quad g(b, a, \text{pa}_m(b) \setminus \{a\}) > 0, \text{ and}$$

$$(iv) \quad m \text{ does not contain a directed path } b \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1} \rightarrow a, \quad x_i \in \mathcal{V} \setminus \{a, b\}, \text{ from } b \text{ to } a.$$

Proof. The negation of this proposition is identical to Proposition 4.4, because

$$\begin{aligned} & \overline{((i) \wedge (ii)) \vee ((iii) \wedge (iv))} \\ = & \overline{((i) \vee (ii)) \wedge ((iii) \vee (iv))} \\ = & \underbrace{\overline{((i) \wedge (iii))}}_{(i)'} \vee \underbrace{\overline{((ii) \wedge (iv))}}_{(ii)'} \vee \underbrace{\overline{((ii) \wedge (iii))}}_{(iii)'} \vee \underbrace{\overline{((ii) \wedge (iv))}}_{(iv)'} \\ = & (i)' \vee (ii)' \vee (iii)', \end{aligned}$$

where the indices (i) , (ii) , (iii) , (iv) refer to this proposition, while the ones with a prime correspond to Proposition 4.4. Again, the negation of (i) is denoted as $\overline{(i)}$, and so on. Note that statement $(iv)'$ can never be fulfilled in a DAG since there cannot be a directed cycle $a \rightarrow \cdots \rightarrow b \rightarrow \cdots \rightarrow a$ involving any two variables $a, b \in \mathcal{V}$. Since the negations of both the Propositions 4.3 and 4.5 are identical, namely to Proposition 4.4, also the propositions themselves have to be equivalent. \square

Although the Propositions 4.3 and 4.5 are equivalent, the latter might be understood more easily. The reason is that, in Proposition 4.5, the neighborhood of a DAG comprises only those graphs which differ in the presence or absence of an edge, whereas DAGs differing in the orientation of an edge are not among these neighbors. This is in contrast to Proposition 4.3, where neighboring DAGs can differ in the presence, absence or orientation of an edge. It is interesting that the definition of the vicinity of a DAG has no impact on the presence or absence of an edge $a \sim b$ when its orientation is disregarded.

Recursive Complete Path in a DAG

In condition (ii) of Proposition 4.4, it is clear that the variable x_{r-1} along the directed path from b to a has to be among the parents of a . This has two consequences. First, the set $\text{pa}_m(a) \setminus \{b\}$ cannot be empty. Second, since the edges between a and each of its parents have to be present by the definition of a DAG, one can replace the required path $b \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1} \rightarrow a$ by the shorter one $b \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1}$, where $x_{r-1} \in \text{pa}_m(a) \setminus \{b\}$. The same holds analogously for the path in (iii) of Proposition 4.4. This suggests the following recursive definition of a "path":

Definition 4.6 (Recursive Complete Path in a DAG) *Let g be a relative scoring function. We define a recursive complete path, or c-path, from a to b ($a, b \in \mathcal{V}$) in a DAG m to have the following properties:*

- (i) *the edge $a \rightarrow b$ is present, or*
- (ii) *the edge $a \rightarrow b$ is absent, $g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$ with $\text{pa}_m(b) \setminus \{a\} \neq \emptyset$, and there is an $x \in \text{pa}_m(b) \setminus \{a\}$ such that a c-path is present from a to x .*

Let the length of a c-path be measured in terms of the number of edges along it.

It is obvious from this definition that a c-path is qualitatively different from a "usual" path in a DAG, as the c-path is not only concerned with the presence of edges, but also with relative scores. Hence, a "usual" path is not necessarily related to a c-path in a DAG. In an *optimal* DAG, however, the following coincidence is given:

Proposition 4.7 *In an optimal DAG m , a directed path $a \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1} \rightarrow b$ ($a, b, x_i \in \mathcal{V}$) is present if and only if there is a c-path from a to b .*

Proof. The fact $g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$ implies that, for all $y \in \text{pa}_m(b) \setminus \{a\}$, the edge $y \rightarrow b$ has to be present by the definition of a DAG. Hence, if there is a c-path from a to b in an optimal DAG then there is a directed path from a to b . Conversely, if there is a directed path from a to b in an optimal DAG, then there exists a shortest directed path $a = x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{r-1} \rightarrow x_r = b$. For all x_i, x_j ($j = 2, \dots, r; i \leq j - 2$) the edge $x_i \rightarrow x_j$ has thus to be absent. Since the DAG is optimal and does not contain cycles, it has to hold that $g(x_j, x_i, \text{pa}_m(x_j) \setminus \{x_i\}) < 0$. Moreover, $\text{pa}_m(x_j) \setminus \{x_i\} \neq \emptyset$ since $x_{j-1} \in \text{pa}_m(x_j) \setminus \{x_i\}$. Now, the existence of the c-path from a to b follows immediately by complete induction on its length. This completes the proof. \square

The advantage of having defined a c-path is that Proposition 4.4 can be restated in a simpler fashion:

Proposition 4.8 *Let g be a relative scoring function, and let m be a (locally) optimal DAG. The edge $a \sim b$ ($a, b \in \mathcal{V}$) is absent in m if and only if*

- (i) $g(a, b, \text{pa}_m(a) \setminus \{b\}) < 0$ and $g(b, a, \text{pa}_m(b) \setminus \{a\}) < 0$, or
- (ii) there is a c -path of length $l \geq 2$ from b to a , or
- (iii) there is a c -path of length $l \geq 2$ from a to b .

Proposition 4.8 can be used for making the decision whether a given DAG is a (local) optimum or not. The so-called necessary path condition, concerning skeletons, is derived from this proposition in the following section.

4.3 Necessary Conditions concerning Skeletons

As structural learning in Bayesian networks is very involved (cf. also Section 3.3), there is demand for approximations such that DAGs close to optimum can be induced efficiently from given data. We thus divide the problem of inducing optimal DAGs into two parts, as typically done in the constraint-based approach. In the first step, we largely ignore the directions of edges, and we hence utilize *skeletons*. Skeletons are undirected graphs obtained by dropping the directions of the edges in a DAG (cf. Section 2.2.1). For brevity, we call the skeleton of a (locally) optimal DAG a (locally) optimal skeleton. The induction of optimal skeletons is naturally concerned with learning about the *presence or absence* of edges. The directions of edges are mainly ignored, which leads to a considerable simplification of the learning task. The second step is then concerned with inducing the directions of edges on the basis of the skeletons induced in the first step. Details on the latter part are provided in Chapter 6.

Of course, it is an approximation to divide the learning task into these two steps. In general, the presence of edges can depend on the orientations, and vice versa. Hence, the presence of an edge (and also its orientation, of course) is subject to the exact structure of the *entire* DAG in general. One source of this non-locality is the *orientation* of edges, as the required acyclicity can entail some constraints on the presence as well as on the orientations, concerning even such edges which are "far away" from each other in the DAG. As a consequence, when the orientations are ignored in the first step of the approximation, one focuses essentially on the *vicinity* of an edge within the graph. This vicinity contains not only variables, like the vicinity due to the faithfulness assumption, but also *edges* and *paths*, as will become clear shortly. The exact non-local learning task is hence again approximated by a *local* one. Like before, this local approach becomes asymptotically correct if the data was sampled from a faithful probability distribution [142]. Due to the extended vicinity, however, this local approximation can be expected to work quite well given *finite* data sets, if the latter are not too small.

Due to this local approximation, the learning task splits up into several independent ones, where each of which is concerned with an edge and its vicinity. The size of an edge's vicinity, i.e.

the number of edges involved, may vary. The edges in the vicinity are, of course, not known at the beginning of the learning process and have to be induced as well. As we will see in the remainder of this chapter, the vicinity of an absent edge comprises *edges* as well as some sort of *paths* required to be present in an optimal graph. This is a consequence of Proposition 4.8, as will become clear later.

A benefit of dividing up the actually non-local learning task into independent local ones is that a considerable speed-up of the computations can be expected. In fact, the required computations are not much more time-consuming than the ones entailed by the PC algorithm, which has proven to be very efficient in many applications [142]. This is typical for virtually all constraint-based approaches, as outlined in Section 3.3.2.

Proposition 4.8, a necessary and sufficient condition for the absence of an edge in an optimal DAG, is not immediately applicable when the skeletons are induced in the first step of the approximation. As a matter of fact, when dealing with skeletons, one has to resort to *necessary* conditions. This means that, if an edge is absent in a skeleton, the latter can only be a (local) optimum when certain edges and paths are present. However, the contrary need not hold, i.e. the skeleton need not be a (local) optimum even if for each edge the required edges and paths are present. Nevertheless, such necessary conditions can help to efficiently find those skeletons which are *candidates* for being an optimum. The properties of these skeletons are further discussed in Section 4.3.4. Let us now be concerned with the vicinity of an edge in detail.

4.3.1 Necessary Path Condition

In this section, we derive a necessary condition applying to optimal skeletons. It is based on Definition 4.6 and Proposition 4.8 concerning DAGs. Let us first define a *recursive complete path in a skeleton*, or *sc-path*, which is present whenever there is a c-path in the corresponding DAG. An sc-path (in a skeleton) is hence a necessary condition for the existence of a c-path (in a locally optimal DAG). A proper definition of an sc-path in a skeleton, unfortunately, is more involved than the definition of a c-path. In particular, two facts have to be considered.

First, the neighbors of a node $a \in \mathcal{V}$ cannot be divided into its parents and its children in a skeleton, as the edges are not oriented. Thus, one has to resort to the fact that the parents $\text{pa}_m(a)$ of a node $a \in \mathcal{V}$ in a DAG m are a subset $\mathcal{S} \subseteq \text{ne}_{\tilde{m}}(a)$ of the neighbors of a in the skeleton \tilde{m} . The variables in such a subset \mathcal{S} can hence serve as candidates for being the parents of a in the corresponding DAG. Furthermore, let us explicitly state that an edge has to be present between a node and each parent-candidate $s \in \mathcal{S}$ in the following propositions and definitions.

Second, a DAG provides an ancestral ordering on the variables in \mathcal{V} , ensuring the c-paths to be well-defined in Definition 4.6: when an edge $a \rightarrow b$ is absent then a c-path is required between a and an $x \in \text{pa}_m(b) \setminus \{a\}$, i.e. the variables along this c-path are ancestors of b . In each recursion step according to Definition 4.6, the number of ancestors which can be along the current c-path decreases, and eventually the recursion terminates. In contrast, a skeleton does

not provide such an ancestral ordering. As a consequence, one has to explicitly ensure that an sc-path in a skeleton is well-defined. Like the number of ancestors in a DAG decreases at each recursion step, the number of edges allowed to be along the current sc-path at each recursion step has to be reduced explicitly. The simplest solution to this problem is as follows: when an edge $a \sim b$ is absent and when there is a subset $S \subseteq \text{ne}(b)$ such that $g(b, a, S) < 0$ then the sc-path required between a and an $s \in S$ is not allowed to include an sc-path between a and b . This is a simple consequence of the ancestral ordering in DAGs. Due to this restriction, the number of allowed sc-paths decreases at each recursion step, ensuring that the recursion will terminate. After these considerations, we can now give a proper definition of a recursive complete path in a skeleton, or sc-path. Since we will present different variants of sc-paths in the following, we enumerate them as sc-1-path, sc-2-path, and so on.

Definition 4.9 (Recursive Complete Path in a Skeleton: SC-1-Path) *Let g be a relative scoring function. Let $[a, b]$ be a pair of variables ($a, b \in \mathcal{V}$),⁶ and let \mathbf{X} be a set of pairs of variables such that $[a, b] \notin \mathbf{X}$. We define a recursive complete path, or sc-1-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to be a tuple $SC_1([a, b], \mathbf{X})$ with the following properties:*

- (i) *the edge $a \sim b$ is present, or*
- (ii) *the edge $a \sim b$ is absent and $\exists S \subseteq \mathcal{V} \setminus \{a, b\}, S \neq \emptyset$, such that*
 - (a) *$g(a, b, S) < 0$, and*
 - (b) *$\forall s \in S$ the edge $s \sim a$ is present, and*
 - (c) *there is an $s' \in S$ such that $[b, s'] \notin \mathbf{X}$ and there exists an sc-1-path $SC_1([b, s'], \mathbf{X} \cup \{[a, b]\})$ between b and s' ,**or*⁷
 - (d) *$g(b, a, S) < 0$, and*
 - (e) *$\forall s \in S$ the edge $s \sim b$ is present, and*
 - (f) *there is an $s' \in S$ such that $[a, s'] \notin \mathbf{X}$ and there exists an sc-1-path $SC_1([a, s'], \mathbf{X} \cup \{[a, b]\})$ between a and s' .*

For brevity, the set \mathbf{X} may not explicitly be stated later. Then it is understood that $\mathbf{X} = \emptyset$. Let the length of an sc-1-path be measured in terms of the number of its edges.

In this definition, the set \mathbf{X} can be interpreted to contain those sc-paths which are not allowed to be part of the sc-1-path $SC_1([a, b], \mathbf{X})$ between a and b . As discussed above, such a set \mathbf{X} is necessary due to the absence of orientations in a skeleton, rendering the definition of an sc-path more complex than the one of a c-path in a DAG. Let us illustrate that the given sc-1-path is well-defined by considering parts of the example depicted in Figure 4.4: in the DAG m_\emptyset ,

⁶A pair of variables $a, b \in \mathcal{V}$ is symmetrical regarding a and b , i.e. $[a, b] = [b, a]$, like a set with two elements.

⁷This notation means explicitly: $(i) \vee \{[(ii, a) \wedge (ii, b) \wedge (ii, c)] \vee [(ii, d) \wedge (ii, e) \wedge (ii, f)]\}$.

there is obviously no c -path present between a and c (cf. Definition 4.6), because all the edges between c and the other variables are absent. Let us now focus on the corresponding skeleton \tilde{m}_0 . After one recursion step according to Definition 4.9, an sc -1-path is present between a and c , i.e. $SC_1([a, c], \emptyset)$ if an sc -1-path $SC_1([c, b], \{[a, c]\})$ exists between b and c . This is a consequence of the absence of the edge $a \sim c$, of the presence of the edge $a \sim b$ and of the score $g(a, c, \{b\}) < 0$. Concerning the sc -1-path $SC_1([c, b], \{[a, c]\})$, another recursion step may be taken according to Definition 4.9, since the edge $b \sim c$ is absent: due to the score $g(b, c, \{a\}) < 0$ and the presence of the edge $a \sim b$, an sc -path between b and c can only be present if there is an sc -1-path between a and c . Since $\mathbf{X} = [a, c]$ in the sc -1-path $SC_1([c, b], \{[a, c]\})$, however, it is forbidden that this sc -1-path between b and c contains an sc -1-path between a and c . Hence, the recursion terminates with the result that no sc -1-path exists between a and c . Note that, without employing the set \mathbf{X} in the above definition, the recursion would not have terminated in this example. The set \mathbf{X} is hence necessary for a well-defined si -path, as it accounts for the ancestral ordering on the variables to some, limited, degree.

One might also consider other definitions of an sc -path, e.g. a stricter one absorbing more properties implied by the ancestral ordering in a DAG. However, this might increase the computational effort. As previously mentioned, the given definition is quite simple, but nevertheless ensures that the sc -path is well-defined. Once an sc -path is defined, a necessary condition regarding skeletons can immediately be obtained from Proposition 4.8:

Proposition 4.10 (Necessary Path Condition) *Let g be a relative scoring function, and let \tilde{m} be a (locally) optimal skeleton. For all pairs of variables $a, b \in \mathcal{V}$, it has to hold that if the edge $a \sim b$ is absent then*

- (i) $\exists S_1, S_2 \subseteq \mathcal{V} \setminus \{a, b\}$ such that
 - (a) $g(a, b, S_1) < 0$ and $g(b, a, S_2) < 0$, and
 - (b) $\forall x \in S_1$ the edge $a \sim x$ is present, and
 - (c) $\forall y \in S_2$ the edge $b \sim y$ is present,

or

- (ii) there exists an sc -path of length $l \geq 2$ between a and b .

The conditions (ii) and (iii) of Proposition 4.8 reduce to a single condition in the necessary path condition due to the absence of orientations in a skeleton. The presence of the edges between a variable and its parent-candidates S_1 and S_2 is explicitly stated in condition (i) of Proposition 4.10. A skeleton can only be optimal if it fulfills this condition. Conversely, not every skeleton which obeys this condition is indeed optimal. Proposition 4.10 is thus only a *necessary* condition. Note that a sufficient condition can only be obtained when the orientations are not disregarded, unlike it is done here in order to obtain an efficient algorithm. It is apparent that the necessary path condition requires certain edges and paths to be present in the *vicinity*

of an *absent* edge. The size of such a vicinity can be different for each absent edge. Its "size" depends on the *number* of edges as well as on the *lengths* of the paths required by the necessary path condition. The former is determined by the order of the induced relative scores, i.e. the number of variables in the set S involved in the score $g(a, b, S)$, while the latter depends on the structure of the induced skeleton. The necessary path condition serves as the basis for our structural learning algorithm described in the next chapter, as it can be used for finding graphs which are candidates for being an optimum.

Recursive Incomplete Path in a Skeleton

The notation of the necessary path condition (cf. Proposition 4.10) can be simplified after defining the *recursive incomplete path*. Moreover, this also renders an extension of the necessary path condition possible, as described in Section 4.3.2.

Definition 4.11 (Recursive Incomplete Path in a Skeleton: SI-1-Path) *Let g be a relative scoring function. We define a recursive incomplete path, or si-1-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to have the following properties:*

- (i) *an sc-1-path is present between a and b , or*
- (ii) *the edge $a \sim b$ is absent and $\exists S_1, S_2 \subseteq \mathcal{V} \setminus \{a, b\}$ such that*
 - (a) *$g(a, b, S_1) < 0$ and $g(b, a, S_2) < 0$, and*
 - (b) *$\forall x \in S_1$ the edge $a \sim x$ is present, and*
 - (c) *$\forall y \in S_2$ the edge $b \sim y$ is present.*

The si-1-path contains both the conditions (i) and (ii) of Proposition 4.10. The only difference of Definition 4.11 to Proposition 4.10 is that the sc-1-path can be of any length, i.e. also of length 1. An si-1-path of unit length means that the corresponding edge is present, while other edges or paths are not required to be present in its vicinity. Because the condition (ii) in the definition of the si-1-path, like the condition (i) in Proposition 4.10, does not require a path to be present, we call the si-1-path a recursive *incomplete* path. With this definition, the necessary path condition 4.10 can be rewritten,

Proposition 4.12 (Necessary SI-Path Condition) *If a skeleton is a (local) optimum then there has to be an si-path between every pair of variables $a, b \in \mathcal{V}$. Conversely, if there is a pair of variables $a, b \in \mathcal{V}$ such that no si-path is between them then the skeleton cannot be a (local) optimum.*

This necessary path condition holds for the si-path in general, i.e. for all the variants of si-paths. So far, only the si-1-path has been presented. In the remainder of this chapter, we discuss

some additional variants of si-paths which can alternatively be employed by this necessary path condition. Hence, there are various variants of the necessary path condition, each of which employing a different si-path. In order to specify the particular variant of necessary path condition, we often refer to the variant of the si-path only, and it is implicitly understood that it is used in the necessary si-path condition (cf. Proposition 4.12). The different si-paths are enumerated as si-1-path, si-2-path and so on, like the sc-paths.

Discussion and Examples

Let us consider a simplistic example (cf. Figure 4.4) to get a better idea of what kind of skeletons comply with the necessary path condition. Because of the positive relative scores

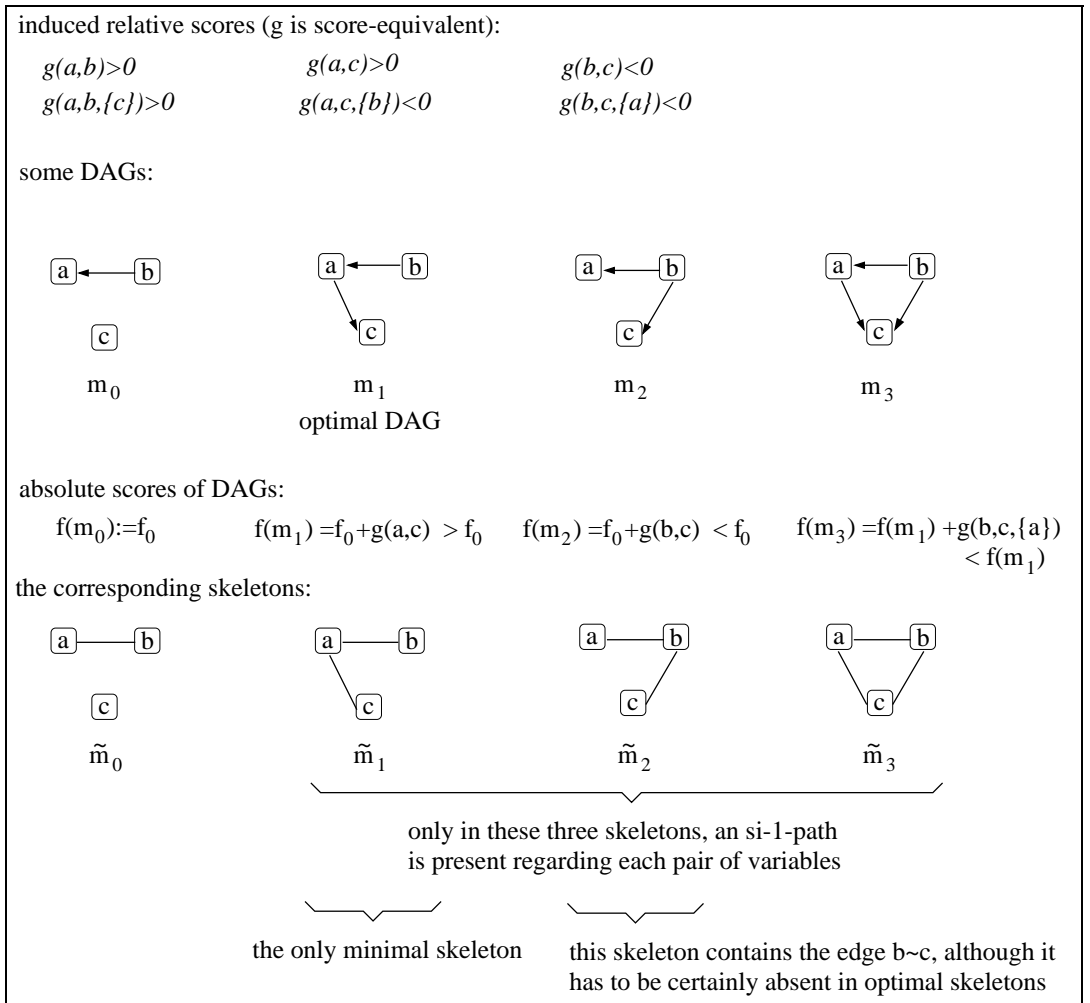


Figure 4.4: The necessary path condition can prevent from removing too many edges in the induced skeleton. Here, the minimal and the optimal skeletons are identical.

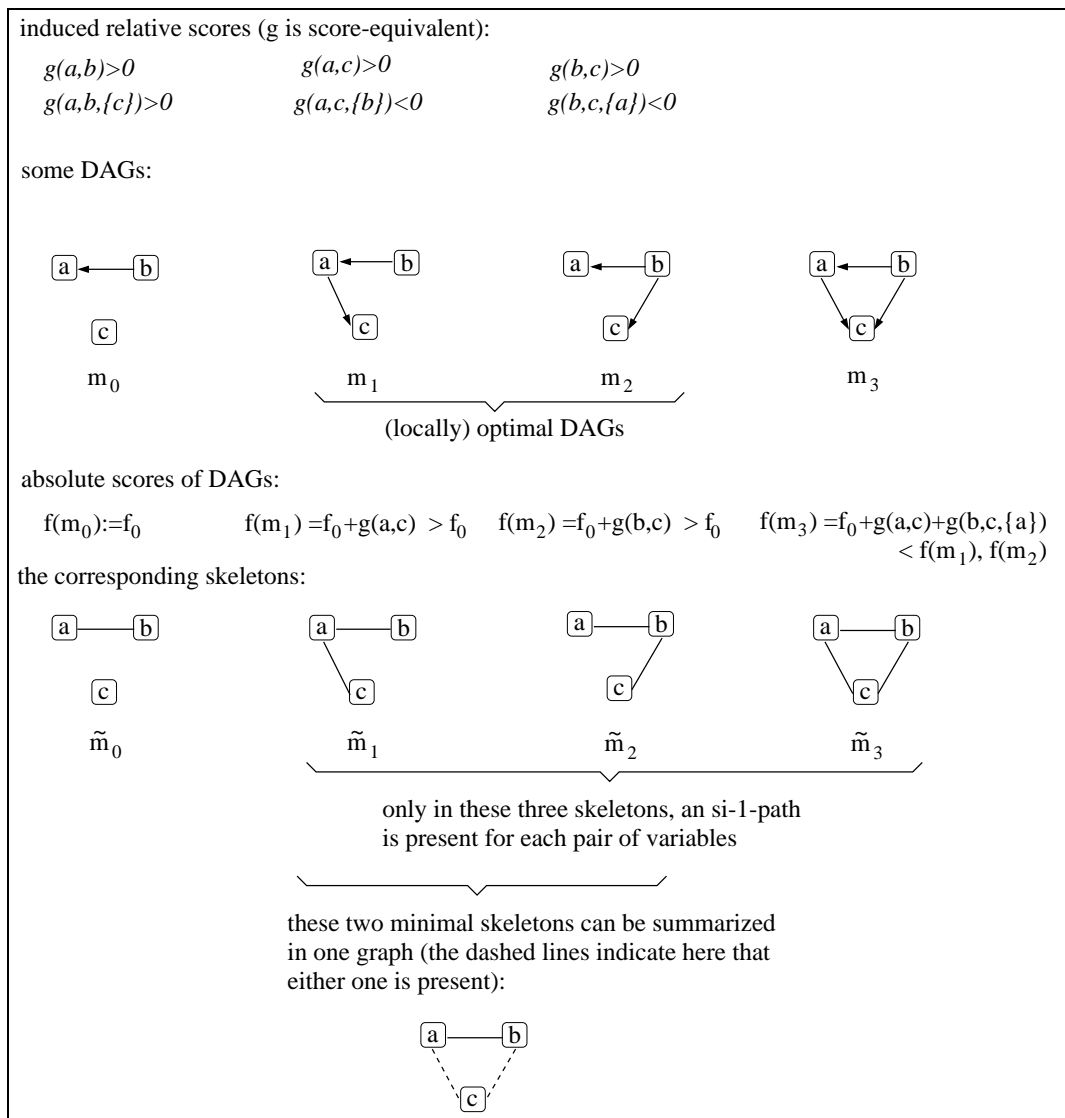


Figure 4.5: Model uncertainty: the necessary path condition can help in finding different local optima.

$g(a,b) > 0$ and $g(a,b,\{c\}) > 0$, the edge $a \sim b$ is certainly present in an optimal skeleton, according to Proposition 4.2. The edge $b \sim c$ is certainly absent, since $g(b,c) < 0$ and $g(b,c,\{a\}) < 0$. In contrast, the presence of the edge $a \sim c$ is not apparent at first glance, as there are both negative and positive scores. This means that this edge is neither certainly present nor certainly absent. Let us first consider the skeleton \tilde{m}_0 , where the latter two edges are absent. It is apparent that this skeleton does not fulfill the necessary si-1-path condition (cf. Proposition 4.12). This is because an si-1-path between a and c is only present if the edge $a \sim c$ itself is present or when the edge $b \sim c$ is present (cf. the Definitions 4.9 and 4.11). The latter is due to the score $g(a,c,\{b\}) < 0$. In contrast, the skeleton \tilde{m}_1 complies with the necessary

path condition, as the presence of the edge $a \sim c$ implies the presence of an si-path between a and c . Also the skeleton \tilde{m}_2 obeys the necessary si-path condition: here, the edge $a \sim c$ can be absent because there exists an sc-path between a and c via b . Obviously, also all skeletons which contain one of the skeletons \tilde{m}_1 or \tilde{m}_2 comply with the necessary si-path condition. In this simplistic example, this is solely the complete graph \tilde{m}_3 . Although the complete graph always fulfills the necessary si-path condition, it might not be the most interesting one. Thus, it is intuitively clear that the most informative skeletons are those from which no edge can be removed without violating the necessary path condition. If these graphs also do not contain a certainly-absent edge (cf. Proposition 4.2) then let us designate them as *minimal* skeletons. In our example, a close look at the skeletons \tilde{m}_1 and \tilde{m}_2 reveals that – although both fulfill the necessary path condition – the latter contains the certainly-absent edge $b \sim c$. Hence, this graph is not minimal, and it cannot be a (local) optimum. In contrast, the skeleton \tilde{m}_1 is minimal, and it happens to coincide with the optimum. Minimal skeletons are often identical with optimal skeletons in the examples used in this chapter. However, this does not hold in general, as will become clear in Section 4.3.4.

A slightly different example is shown in Figure 4.5. In fact, only the score $g(b, c) > 0$ is different compared to the previous example. However, the consequences are considerable, as the edge $c \sim b$ is not certainly absent in this example. Thus, skeleton \tilde{m}_2 is minimal, like skeleton \tilde{m}_1 . This is because they both fulfill the necessary si-path condition, and they do not contain a certainly-absent edge (cf. Proposition 4.2). In this example, there are hence two different minimal skeletons. It occurs that each of the minimal skeletons corresponds to a different local optimum, as becomes clear from considering the scores of the DAGs. This means that *model uncertainty* can be discovered by means of the necessary si-path condition. In general, this is only possible up to a degree, of course. The exact problem would be NP-complete, anyway. For concise representation, we display this kind of model uncertainty in a single undirected graph where dashed lines indicate that these edges might be present in some of the minimal skeletons (cf. Figure 4.5). Although the meaning of such a graph is intuitively clear, its exact definition has to be postponed to Section 5.3.

Another interesting consequence of the necessary path condition is that an edge $a \sim b$ can only be *absent* due to a score $g(a, b, S) < 0$ with $S \neq \emptyset$ when other edges and paths are *present* in the skeleton. Hence, the necessary path condition entails some *balance* between the presence and absence of the various edges. For instance, in the two minimal skeletons depicted in Figure 4.5, either one of the edges $a \sim c$ and $b \sim c$ has to be present. Both edges cannot be absent simultaneously without violating the necessary si-path condition. The necessary path condition hence yields interdependencies among the various edges. Of course, this also implies that such a condition typically yields skeletons which contain more edges than the ones induced without a necessary condition.

4.3.2 Extended Necessary Path Condition

As mentioned at the beginning of Section 4.3, the necessary path condition can be viewed as a local approximation to the learning task, since it is concerned with the edges and path present in

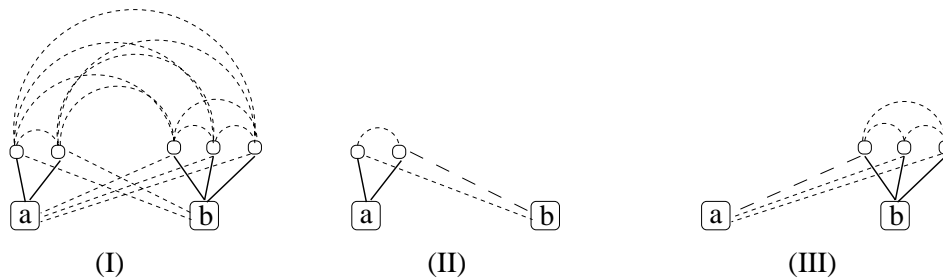


Figure 4.6: In the local approximation, the vicinity of an absent edge $a \sim b$ can be expanded to contain additional si-paths (short-dashed lines) in each of the three cases (I), (II) and (III). The edges between a node and its parent-candidates are sketched as solid lines, and a required sc-path is depicted as a long-dashed line.

the vicinity of an absent edge. Hence, it can naturally be improved by considering an expanded vicinity of each absent edge. The necessary si-1-path condition yields immediately such an extension: because an si-path has to be present between *every* pair of variables, an si-path has to be present, in particular, between the following variables if the edge $a \sim b$ is absent:

- If there is no sc-path between a and b (cf. also Figure 4.6, (I)) then an si-path has to be present
 - between a and the parent-candidates of b , and
 - between b and the parent-candidates of a , and
 - among the parent-candidates of a and b .
- If there is an sc-path between a and b (cf. also Figures 4.6, (II) and (III)) then an si-path has to be present
 - between b and each parent-candidate of a , and
 - among the parent-candidates of a ,
 or
 - between a and each parent-candidate of b , and
 - among the parent-candidates of b .

These additional si-paths extend the vicinity of an absent edge compared to the vicinity accounted for by the si-1-path. This suggests the following definitions of an si-2-path and of an sc-2-path:

Definition 4.13 (Extended Recursive Complete Path in a Skeleton: SC-2-Path) *Let g be a relative scoring function. Let $[a, b]$ be a pair of variables ($a, b \in \mathcal{V}$), and let \mathbf{X} be a set of pairs of variables such that $[a, b] \notin \mathbf{X}$. We define an extended recursive complete path, or sc-2-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to be a tuple $SC_2([a, b], \mathbf{X})$ with the following properties:*

(i) *the edge $a \sim b$ is present, or*

(ii) *the edge $a \sim b$ is absent, and $\exists S \subseteq \mathcal{V} \setminus \{a, b\}$, $S \neq \emptyset$, such that*

(a) *$g(a, b, S) < 0$, and*

(b) *$\forall x \in S$ the edge $a \sim x$ is present, and*

(c) *$\exists x \in S$: $[b, x] \notin \mathbf{X}$ and there is an sc-2-path $SC_2([b, x], \mathbf{X} \cup \{[a, b]\})$ between b and x , and*

(d) *$\forall z \in S$: $[b, z] \notin \mathbf{X}$ and there is an si-2-path $SI_2([b, z], \mathbf{X} \cup \{[a, b]\})$ between b and z , and*

(e) *$\forall w, z \in S$, $w \neq z$: $[w, z] \notin \mathbf{X}$ and there is an si-2-path $SI_2([w, z], \mathbf{X} \cup \{[a, b]\})$ between w and z ,*

or

(f) *$g(b, a, S) < 0$, and*

(g) *$\forall x \in S$ the edge $b \sim x$ is present, and*

(h) *$\exists x \in S$ such that $[a, x] \notin \mathbf{X}$ and an sc-2-path $SC_2([a, x], \mathbf{X} \cup \{[a, b]\})$ is between a and x , and*

(i) *$\forall z \in S$: $[a, z] \notin \mathbf{X}$ and there is an si-2-path $SI_2([a, z], \mathbf{X} \cup \{[a, b]\})$ between a and z , and*

(j) *$\forall w, z \in S$, $w \neq z$: $[w, z] \notin \mathbf{X}$ and there is an si-2-path $SI_2([w, z], \mathbf{X} \cup \{[a, b]\})$ between w and z .*

If the set \mathbf{X} is not explicitly stated later then it is understood that $\mathbf{X} = \emptyset$.

The sc-2-path is illustrated in Figure 4.6, (II) and (III). The conditions (d), (e), (i) and (j) represent the si-paths additionally required compared to the Definition 4.9 of an sc-1-path. Like before, the set \mathbf{X} is necessary because it accounts for the ancestral ordering in the corresponding DAG to a degree which is (small but) sufficient to ensure the sc-2-path to be well-defined. Due to this extension, the definitions of the sc-2-path and the si-2-path depend on each other, unlike the definition of an sc-1-path (cf. Definition 4.9).

Definition 4.14 (Extended Recursive Incomplete Path in a Skeleton: SI-2-Path) *Let g be a relative scoring function. Let $[a, b]$ be a pair of variables ($a, b \in \mathcal{V}$), and let \mathbf{X} be a set of pairs of variables such that $[a, b] \notin \mathbf{X}$. We define an extended recursive incomplete path, or si-2-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to be a tuple $SI_2([a, b], \mathbf{X})$ with the following properties:*

- (i) *there is an sc-2-path $SC_2([a, b], \mathbf{X})$ between a and b , or*
- (ii) *the edge $a \sim b$ is absent, and $\exists S_1, S_2 \subseteq \mathcal{V} \setminus \{a, b\}$ such that ⁸*
 - (a) *$g(a, b, S_1) < 0$ and $g(b, a, S_2) < 0$, and*
 - (b) *$\forall x \in S_1$ the edge $a \sim x$ is present, and*
 - (c) *$\forall y \in S_2$ the edge $b \sim y$ is present, and*
 - (d) *$\forall s_1, s_2 \in S_1 \cup S_2 \cup \{a, b\}, s_1 \neq s_2: [s_1, s_2] \notin \mathbf{X}$, and*
 - (e) *$\forall z \in S_1$ there is an si-2-path $SI_2([b, z], \mathbf{X} \cup \{[a, b]\})$ between b and z , and*
 - (f) *$\forall z \in S_2$ there is an si-2-path $SI_2([a, z], \mathbf{X} \cup \{[a, b]\})$ between a and z , and*
 - (g) *$\forall w, z \in S_1 \cup S_2, w \neq z$, there is an si-2-path $SI_2([w, z], \mathbf{X} \cup \{[a, b]\})$ between w and z .*

If the set \mathbf{X} is not explicitly stated later then it is understood that $\mathbf{X} = \emptyset$.

Condition (ii) of the si-2-path is illustrated in Figure 4.6, (I). The requirements (d), (e), (f) and (g) are additional to the si-1-path (cf. Definition 4.11).

The simplistic example in Figure 4.7 illustrates the main difference between the si-1-path and the si-2-path, namely that the latter accounts for an extended vicinity. The relative scores in Figure 4.7 immediately imply that the edges $a \sim d$ and $b \sim c$ are certainly absent, while the edges $a \sim c$ and $b \sim d$ are certainly present (cf. Proposition 4.2). Let us hence focus on the edges $a \sim b$ and $c \sim d$. The scores $g(c, d, \{a\}) < 0$ and $g(c, d, \{b\}) < 0$ imply that, according to the sc-1-path condition, the edge $c \sim d$ is allowed to be absent if the edges $a \sim c$ and $b \sim d$ are present. The edge $a \sim b$ is not involved in the vicinity of the edge $c \sim d$. Consequently, the minimal skeleton according to the necessary si-1-path condition contains only the edges $a \sim c$ and $b \sim d$ (cf. skeleton \tilde{m}_0 in Figure 4.7). In contrast, if the edge $c \sim d$ is absent then the extended vicinity of the si-2-path requires an si-path $SI_2([a, b], \{[c, d]\})$ to be present between a and b . Such an si-path between a and b can, however, only be present if the edge $a \sim b$ by itself exists. Due to the symmetry regarding the edges $a \sim b$ and $c \sim d$ in this example, it is apparent that the si-2-path condition yields that either one of the edges $a \sim b$ or $c \sim d$ has to be present in a minimal skeleton (cf. the skeletons \tilde{m}_1 and \tilde{m}_2 in Figure 4.7). The resulting uncertainty regarding the presence of edges can again be summarized in a single graph (cf. Figure 4.7). Summing up, the necessary si-2-path can yield additional edges

⁸If S_1 or S_2 are empty the corresponding edges or paths do not have to be present, of course.

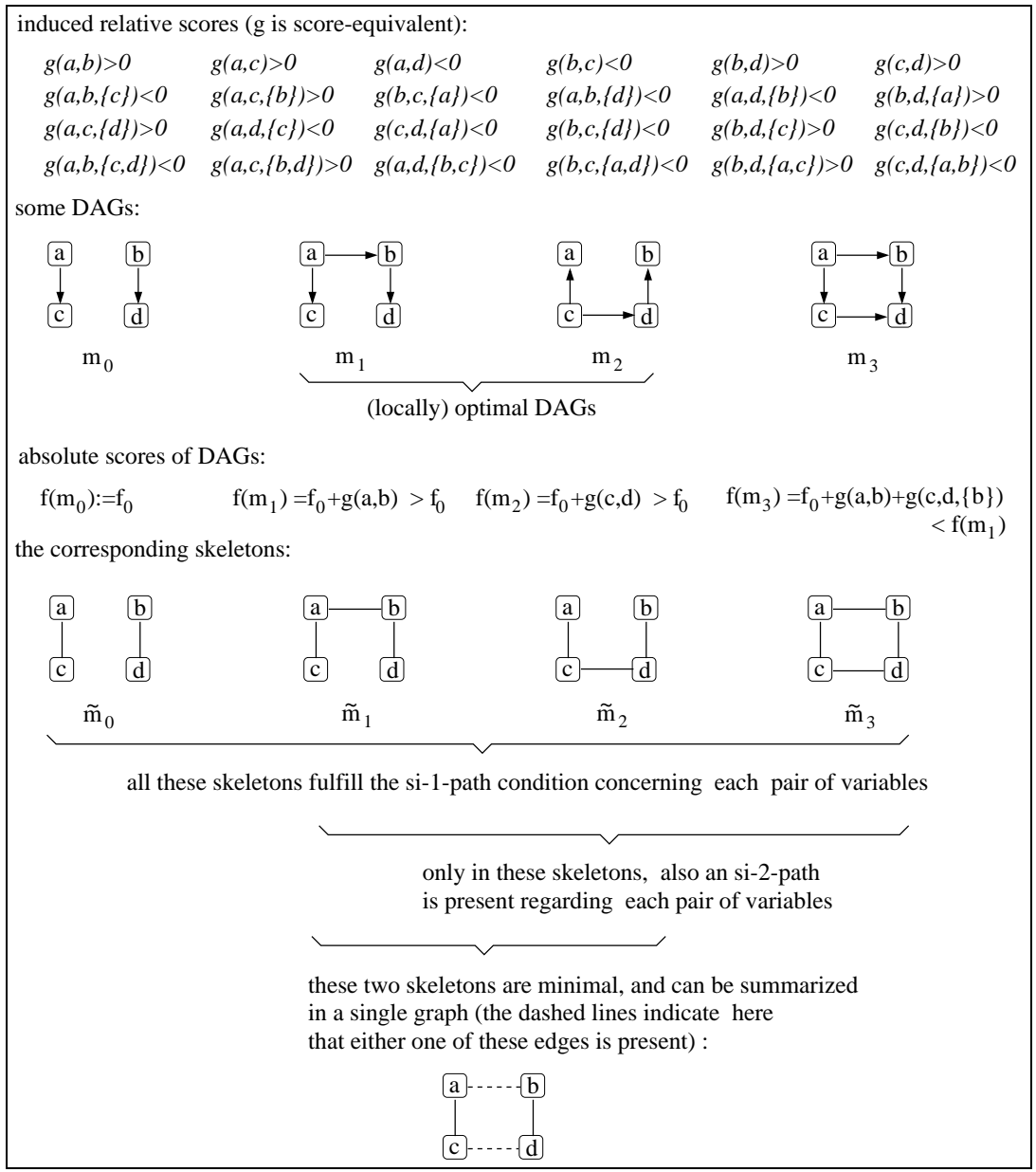


Figure 4.7: Comparison of the si-1-path and the si-2-path.

compared to the necessary si-1-path. Also, note that the edges contained in each of the minimal skeletons obeying the si-2-path condition are also present in (locally) optimal DAGs.

The extended necessary path condition entails, of course, more involved computations. In particular, if model uncertainty is discovered, a large number of edges might be involved. For this reason, efficient computations often require additional approximations, as discussed in Section 5.3.2.

4.3.3 Simplified Necessary Path Conditions

This section is concerned with necessary path conditions which are less strict than the previous ones. There are two ways for obtaining simplified conditions. First, the size of the vicinity of an absent edge can be reduced by requiring the presence of fewer edges or paths. Second, the strictness can be weakened by replacing, for instance, a necessarily present edge by a path. Also, one might resort to paths in the vicinity of an absent edge which act as conditions less strict than other kinds of paths. Hence, there are many options for replacing some of the necessarily required edges, sc-paths or si-paths by less strict ones. Let us focus on the particular case where only a *single* kind of paths is used rather than both the si-path and sc-path.

Definition 4.15 (Simplified Recursive Incomplete Path in a Skeleton: SI-3-Path) *Let g be a relative scoring function. Let $[a, b]$ be a pair of variables ($a, b \in \mathcal{V}$), and let \mathbf{X} be a set of pairs of variables such that $[a, b] \notin \mathbf{X}$. We define a simplified recursive incomplete path, or si-3-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to be a tuple $SI_3([a, b], \mathbf{X})$ with the following properties:*

- (i) *the edge $a \sim b$ is present, or*
 - (ii) *the edge $a \sim b$ is absent and $\exists \mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ such that*
 - (a) *$g(a, b, \mathcal{S}) < 0$, and*
 - (b) *$\forall s \in \mathcal{S}$ the edge $a \sim s$ is present, and*
 - (c) *$\forall s \in \mathcal{S}: [b, s] \notin \mathbf{X}$ and there exists an si-3-path $SI_3([b, s], \mathbf{X} \cup \{[a, b]\})$ between b and s ,*
- or*
- (d) *$g(b, a, \mathcal{S}) < 0$, and*
 - (e) *$\forall s \in \mathcal{S}$ the edge $b \sim s$ is present, and*
 - (f) *$\forall s \in \mathcal{S}: [a, s] \notin \mathbf{X}$ and there exists an si-3-path $SI_3([a, s], \mathbf{X} \cup \{[a, b]\})$ between a and s .*

If the set \mathbf{X} is not explicitly stated later then it is understood that $\mathbf{X} = \emptyset$.

The si-3-path requires fewer paths to be present than the si-2-path or the sc-2-path. Moreover, no sc-path is involved. The si-3-path is hence less strict than the si-2-path or the sc-2-path. In other words, whenever there is an si-2-path or an sc-2-path between two variables then there exists also an si-3-path. As a consequence, the si-3-path might yield skeletons with a reduced number of edges. The advantage of this simplification is, however, that only a single kind of paths has to be considered, which eases the implementation of the corresponding algorithm. Despite this simplification, the si-3-path is still capable of allowing for the asymmetry concerning the variables a and b when a non-score-equivalent scoring-function is used, i.e. when $g(a, b, \mathcal{S}) \neq g(b, a, \mathcal{S})$. This is because the si-3-path requires edges between each variable and

its parent-candidates \mathcal{S} , whereas si-paths have to be present between the parent-candidates \mathcal{S} and the other variable. Finally, let us comment on the simplest si-path. It is obtained from the si-3-path by replacing the *edges* between each variable and its parent-candidates by *si-paths*. This leads to the least strict necessary path condition in this thesis:

Definition 4.16 (Simplest Recursive Incomplete Path in a Skeleton: SI-4-Path) *Let g be a relative scoring function. Let $[a, b]$ be a pair of variables ($a, b \in \mathcal{V}$), and let \mathbf{X} be a set of pairs of variables such that $[a, b] \notin \mathbf{X}$. We define the simplest recursive incomplete path, or si-4-path, between two variables $a, b \in \mathcal{V}$ in a skeleton \tilde{m} to be a tuple $SI_4([a, b], \mathbf{X})$ with the following properties:*

- (i) *the edge $a \sim b$ is present, or*
- (ii) *the edge $a \sim b$ is absent and $\exists \mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ such that*
 - (a) *$g(a, b, \mathcal{S}) < 0$ or $g(b, a, \mathcal{S}) < 0$, and*
 - (b) *$\forall s \in \mathcal{S}: [b, s] \notin \mathbf{X}$ and there exists an si-4-path $SI_4([b, s], \mathbf{X} \cup \{[a, b]\})$ between b and s , and*
 - (c) *$\forall s \in \mathcal{S}: [a, s] \notin \mathbf{X}$ and there exists an si-4-path $SI_4([a, s], \mathbf{X} \cup \{[a, b]\})$ between a and s .*

If the set \mathbf{X} is not explicitly stated later then it is understood that $\mathbf{X} = \emptyset$.

Obviously, the si-4-path is symmetrical concerning the variables a and b . This is because it requires si-paths between a and each $s \in \mathcal{S}$ and between b and each $s \in \mathcal{S}$. For this reason, the si-4-path is particularly appropriate when it is used in conjunction with score-equivalent scoring-functions. Since $g(a, b, \mathcal{S}) = g(b, a, \mathcal{S})$ holds in this case, the condition (ii, a) simplifies even further. The different variants of si-paths (and sc-paths) are further discussed in Chapter 5, which is concerned with their implementations and their performance in our experiments. We also compare them with popular constraint-based approaches, which do not employ a necessary path condition.

4.3.4 Minimal Skeletons

This section summarizes the properties of *minimal* skeletons, which were first mentioned in Section 4.3.1. Since many different skeletons generally comply with the necessary si-path condition, most information is provided by those skeletons which define the border between the graphs obeying the necessary path condition and the ones which do not. This divides the search space of skeletons into two parts. We call the skeletons on this border *minimal*, when they do not only comply with the necessary path condition but also contain no certainly-absent edge (cf. Proposition 4.2). In other words, these skeletons contain a minimal number of edges

in the sense that one cannot remove any edge without violating the necessary path condition. Consequently, the set of all skeletons obeying the necessary path condition contains not only the minimal skeletons but also the graphs obtained by adding arbitrary edges to a minimal skeleton.

All the optimal skeletons are among the graphs fulfilling the necessary path condition, since the necessary path condition is a *necessary* rather than a sufficient condition for a skeleton to be optimal. Conversely, not every optimal skeleton has to be minimal. Hence, each of the minimal skeletons is optimal or it is contained in an optimal skeleton. The latter is understood in the sense that an optimal skeleton might comprise some edges additional to the minimal skeleton. In other words, a minimal skeleton is a subgraph of an optimal skeleton. For this reason, the necessary path condition can help determine edges which are present in (locally) optimal DAGs. Moreover, the induced edges are present with some confidence. However, the degree of confidence cannot be calculated by this approach. In order to compute confidence intervals, one has to apply computationally more involved approaches, like for instance the bootstrap [50,57] (cf. also Section 3.2). The constraint-based approach (applying the necessary path condition) might hence be used as a computationally efficient, but limited, alternative to the bootstrap.

Since the minimal skeletons tend to contain less edges than the optimal ones, one can only arrive at (locally) optimal skeletons if the "missing" edges are included into the minimal skeletons in a subsequent step. Since only necessary conditions are applicable in the space of skeletons, the final step of such a learning procedure has to take place in the space of DAGs. Chapter 6 is concerned with such a scheme, which is greedy for simplicity.

Another aspect of minimal skeletons is as follows. If an edge $a \sim b$ is absent due to the score $g(a, b, \mathcal{S}) < 0$ then it is so because the subset \mathcal{S} *might be* the parent set of a in the optimal DAG m . However, it does not have to hold that $\text{pa}_m(a) = \mathcal{S}$. Regarding an edge $a \sim b$ being absent from a minimal skeleton, it is hence implicitly assumed that $g(a, b, \text{pa}_m(a)) < 0$ if $g(a, b, \mathcal{S}) < 0$. Since this is not guaranteed to hold in general, minimal skeletons tend to contain less edges than optimal DAGs, implying the same conclusion as above.

Regarding multiple testing, let us consider a standard constraint-based algorithm, which removes an edge $a \sim b$ when a conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$ was found for some set $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$. When a statistical test with a significance level α , e.g. 5%, is used, one might expect at first glance that the fraction of edges erroneously found to be present is of the order of 5%, like the Type I error in each of the tests. The experiments indicate that this number is, however, much lower. A close look reveals that an edge is only present in the induced graph when the tests on conditional independence reject the hypotheses $a \perp\!\!\!\perp b \mid \mathcal{S}$ for *all* the subsets \mathcal{S} under consideration. Hence, a test which erroneously yields a dependence (Type I error) has no impact on the resulting graph as long as there is another test yielding the absence. Thus, an edge is removed when a conditional independence is induced by at least one test. For instance, in the PC algorithm [142], when tests at low orders of \mathcal{S} yield erroneously a dependence then tests of higher orders are carried out. The latter might yield a conditional independence, as tests

of higher orders are generally less reliable. Summing up, chances that an edge is erroneously present are much smaller than the significance level α used in each test, while chances that an edge is erroneously absent is much larger than the (unknown) Type II error in each test. This implies again that the graphs induced by the constraint-based approach contain only such edges which are present with a high degree of certainty. Another conclusion is that too few edges are found to be present, compared to the optimum.

4.3.5 Parsimony

The heuristics of parsimony is often used in machine learning. This is because parsimonious models tend to exhibit less over-fitting than other, more complex models. This is particularly important if the scoring function is only concerned with the fit of the model, and does not take into account its complexity. In this case, the search strategy has to account for model complexity in some sense, e.g. by preferring parsimonious models.

When learning in Bayesian networks, the usual scoring functions contain a penalty term for model complexity (cf. Section 3.1.4). As a consequence, one can expect that the optimal DAGs with respect to such a scoring function do not exhibit over-fitting. The employed search strategies can thus aim at inducing the optima.

When the necessary path condition is used for determining minimal skeletons in our approach, the scores of the *skeletons*, however, can not be calculated, as the orientations of the edges are undetermined. When there is a large number of minimal skeletons, some of which might correspond to a DAG with a higher score than others do. Since computing all those minimal skeletons can often be computationally infeasible, we apply the heuristics of parsimony, i.e. we prefer those skeletons which contain the fewest edges among all the minimal skeletons. For illustration, let us consider the simplistic example in Figure 4.8: unlike in the previous examples, the two minimal skeletons contain a different number of edges here, namely three and four edges, respectively. It is left to the reader to verify that the shown skeletons are indeed minimal. Let us only note that the score $g(a, d, \{b, c\}) < 0$ versus both the scores $g(b, d, \{a\}) < 0$ and $g(c, d, \{a\}) < 0$ plays a decisive role. The heuristics of parsimony prefers the skeleton with three edges to the other one with four edges. Applying the heuristics of parsimony hence excludes some minimal skeletons from consideration. For this reason, the global optimum need not be among the parsimonious skeletons or among the graphs obtained by adding some edges to the parsimonious skeletons. This is a main difference between minimality, as described in the previous section, and parsimony, as employed here. However, in most of our experiments the heuristics of parsimony not only entailed more efficient computations but also selected those minimal skeletons which were closer to the global optimum than the others (cf. Section 5.6.2). For these reasons, parsimony is often very useful in practice.

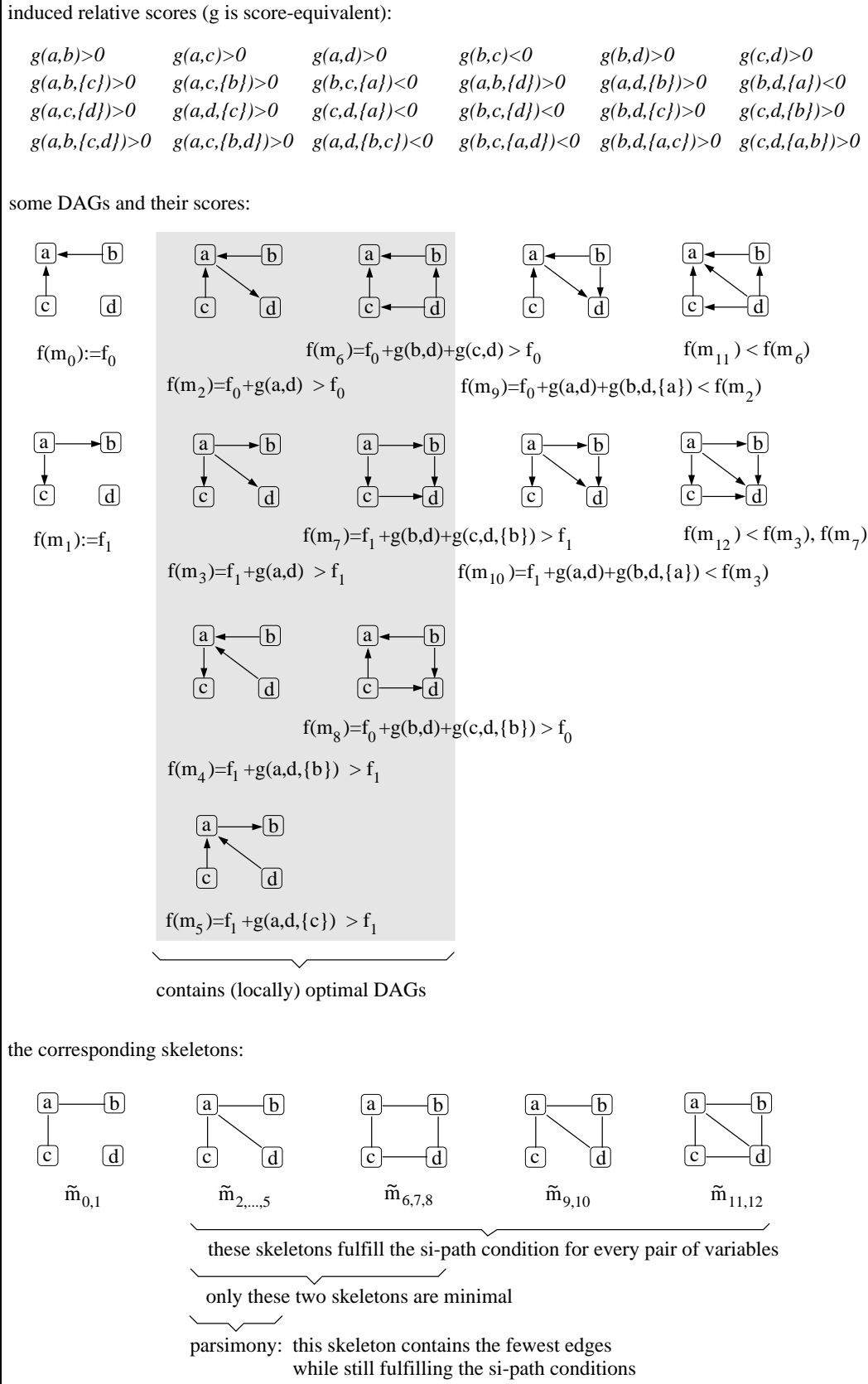


Figure 4.8: Illustration of parsimony.

4.4 Properties of the Perfect Map

Popular constraint-based learning algorithms, e.g. [142], assume that the probability distribution underlying the data is faithful (cf. also Section 2.2.4). Their aim is hence to recover the perfect map given the induced conditional independences. When the conditional independences and dependences (CIDs) are induced from *finite* data by means of statistical tests, one cannot expect in general that there is a DAG which can represent *all* the induced CIDs, even if the data was sampled from a faithful probability distribution. The reason is that a penalty concerning model complexity is inherent in a statistical test or some other decision mechanism (cf. also Section 4.1). Consequently, a test concerned with the independence of a and b conditional on some set S might yield $g(a, b, S) < 0$, and hence imply $a \perp\!\!\!\perp b \mid S$, only because the data set was small.

This suggests to represent *all* the induced conditional *dependences*, but *not all* the induced conditional *independences* in the induced DAGs. Such DAGs are hence I-maps of the induced CIDs (cf. Section 2.2.4). Moreover, these graphs might be viewed as candidates for being the perfect map of the true probability distribution underlying the data. Apparently, there can be several such I-maps. Most interesting are minimal I-maps because they represent a maximal number of the induced conditional independences.

Like in the case of optimal DAGs, we focus on a necessary condition which has to be fulfilled in a perfect map. This condition can then be employed by a learning algorithm to find graphs which are candidates for being the skeleton of the perfect map. This kind of approach was also described in [144]. Regarding the presence of an edge in a perfect map, the following proposition can easily be derived.

Proposition 4.17 *Let p be a faithful probability distribution for the variables in \mathcal{V} , i.e. there exists a perfect map m . If the edge $a \sim b$ is absent in m then*

- (i) *there is a set $S \subseteq \text{pa}_m(a) \setminus \{b\}$ in m such that $a \perp\!\!\!\perp b \mid S$ in p , and there is an $x \in S$ such that there is a path $x = x_0 \sim x_1 \sim \dots \sim x_r = b$ without converging arrows, i.e. $x_{i-1} \rightarrow x_i \leftarrow x_{i+1}$ ($i = 1, \dots, r-1$), in m ,*

or

- (ii) *there is a set $S \subseteq \text{pa}_m(b) \setminus \{a\}$ in m such that $a \perp\!\!\!\perp b \mid S$ in p , and there is an $x \in S$ such that there is a path $x = x_0 \sim x_1 \sim \dots \sim x_r = a$ without converging arrows, i.e. $x_{i-1} \rightarrow x_i \leftarrow x_{i+1}$ ($i = 1, \dots, r-1$), in m .*

Proof. This follows directly from the definition of a perfect map (cf. Definition 2.4) and the d-separation criterion (cf. Definition 2.1). \square

It is apparent that this condition is very similar to the ones applying to optimal graphs (cf. Section 4.2). Like in the optimal DAG, there need not be a path without converging arrows

between every $s \in \mathcal{S} \subseteq \text{pa}_m(b) \setminus \{a\}$ and a in the perfect map. In fact, even if $\mathcal{S} \neq \emptyset$ is minimal, i.e. $a \perp\!\!\!\perp b \mid \mathcal{S}$ and $\forall \mathcal{S}' \subset \mathcal{S} : a \not\perp\!\!\!\perp b \mid \mathcal{S}'$, it is only guaranteed that there is at least one variable $x \in \mathcal{S}$ such that there is a path without converging arrows between x and a . This is illustrated in Figure 4.9, where the set $\mathcal{S} = \{c, d, e\}$ is minimal such that a and b are d-separated by \mathcal{S} , and hence $a \perp\!\!\!\perp b \mid \mathcal{S}$. It is obvious that there exists only one path without converging arrows between an $s \in \mathcal{S}$ and a , namely $a \rightarrow c$. Along the paths between a and $d \in \mathcal{S}$ or between a and $e \in \mathcal{S}$ is a collider at variable c .

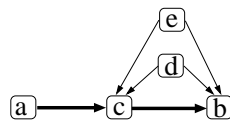


Figure 4.9: When given $a \perp\!\!\!\perp b \mid \{c, d, e\}$, there need not exist more than one path without converging arrows between a and b in a perfect map.

Proposition 4.17 leads immediately to a necessary path condition applying to the skeleton of the perfect map:

Proposition 4.18 (Necessary SC-P-Path Condition) *Let p be a faithful probability distribution for the variables in \mathcal{V} , i.e. there exists a perfect map m . Between every pair of variables $a, b \in \mathcal{V}$ there has to be an sc-1-path in the skeleton \tilde{m} if the two variables are not marginally independent.*

The definition of an sc-1-path, which is originally concerned with optimal skeletons (cf. Definition 4.9), applies also to the skeleton of the perfect map when $g(a, b, \mathcal{S}) < 0$ is understood as an induced conditional independence $a \perp\!\!\!\perp b \mid \mathcal{S}$.

Apparently, if two variables are marginally dependent, there has to be an sc-1-path in the skeleton of the perfect map. This is the main difference to the necessary si-path condition concerning optimal skeletons (cf. Proposition 4.12), where an si-path rather than an sc-path is required between every pair of variables. Hence, an edge which is present in a minimal skeleton complying with the necessary si-path condition (concerning optimal skeletons) is also present in the minimal skeleton fulfilling the necessary sc-p-path condition. However, the latter might contain additional edges, as illustrated in Figure 4.10. Because the variables a and b are marginally dependent due to the score $g(a, b) > 0$, the necessary sc-p-path condition requires an edge to be present between the connected component involving a, c and d and the other one comprising b, e and f . In contrast, such an edge is absent in the optimal DAG, and it is not required by the necessary si-path condition applying to optimal skeletons (cf. Proposition 4.12).

Although the necessary sc-p-path condition is asymptotically correct if there exists a perfect map, it can require edges to be present which are indeed absent in optimal DAGs when given finite samples. Hence, this asymptotic result cannot simply be applied to finite data. The same applies also to an early version of the PC algorithm, also called PC* algorithm, as the required paths are derived from the perfect map [140, 142]. In order to induce only those edges which

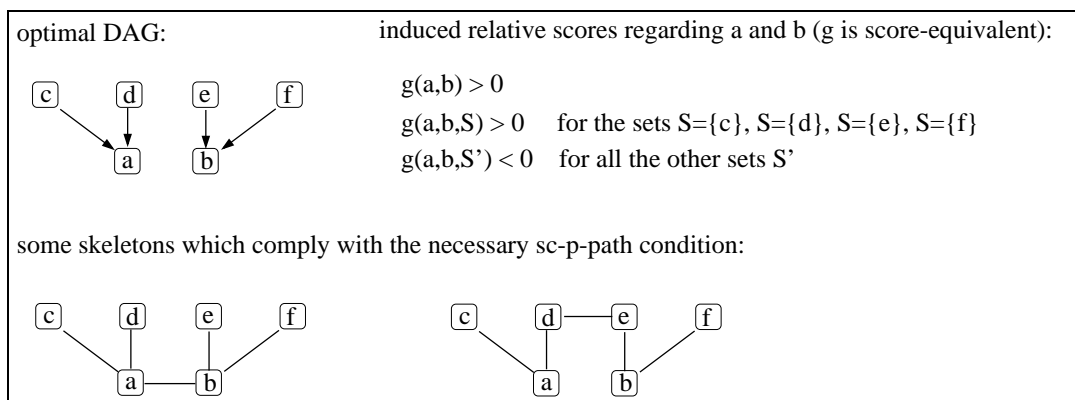


Figure 4.10: If the variables a and b are marginally dependent, there has to be an sc-path (rather than an si-path) present between them in the skeleton of the assumed perfect map.

are present in optimal DAGs, one thus has to use the necessary *si*-path condition derived from properties of *optimal* DAGs.

5

Learning the Presence of Edges

After the necessary path condition was derived in the previous chapter, in this chapter we are concerned with the structural learning algorithm aimed at finding the corresponding parsimonious skeletons, based on the induced relative scores. We focus on how such an algorithm works and what computational aspects have to be considered. The necessary si-path condition (cf. Proposition 4.12) is represented by a *set of rules*, which can efficiently be simplified by the learning algorithm. A skeleton complies with the necessary path condition if and only if a rule is fulfilled for each absent edge. A main difference between the presented extension and state-of-the-art constraint-based algorithms is that possibly *several* graphs can be found. Moreover, the multiple solutions can be displayed in a single graph, easing the interpretation. The basic ideas described in this chapter were published in [144]. In various experiments, the proposed approach is compared to other popular learning algorithms.

5.1 Rules

Since the sc-paths and si-paths are defined recursively (cf. Chapter 4), we use two kinds of rules for representing each of which, the *c-rule* and the *i-rule*. In the following, the most involved variant of rules is described, namely the one corresponding to si-2-paths and sc-2-paths (cf. the Definitions 4.14 and 4.13). The other variants of the learning algorithm, based on simpler paths like the sc-1-path, si-1-path, si-3-path or si-4-path, can be obtained analogously. In particular, the algorithms employing the si-3-path or si-4-path are considerably simpler than the variant described here, because the former do not involve sc-paths.

A c-rule is a quadruple $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$, and an i-rule is a triple $IR(X, \mathbf{E}, \mathbf{I})$, where $X = [a, b] = [b, a]$ is a pair of variables $a, b \in \mathcal{V}$, and \mathbf{E} , \mathbf{C} and \mathbf{I} are sets of pairs of variables. According to the necessary si-path condition and the definitions of the sc-paths and si-paths, the following interpretation is suggested: $X = [a, b]$ represents the edge $a \sim b$ which is

possibly absent. If it is absent then

- an edge has to be present between the variables x and y if $[x, y] \in \mathbf{E}$,
- an sc-path has to be present between the variables x and y if $[x, y] \in \mathbf{C}$, and
- an si-path has to be present between the variables x and y if $[x, y] \in \mathbf{I}$.

Hence, a pair of variables can be interpreted as an edge, an sc-path or an si-path depending on the context. For brevity, we hence call the elements of \mathbf{E} , \mathbf{C} and \mathbf{I} edges, sc-paths and si-paths, respectively. The sets \mathbf{E} , \mathbf{C} and \mathbf{I} represent the conditions under which the edge X can be absent in a skeleton such that the necessary si-path condition is fulfilled. We say that a rule is fulfilled when all the conditions represented by the sets \mathbf{E} , \mathbf{C} and \mathbf{I} are met.

Once the relative scores have been computed, each of which is transformed into a rule. For clarity, the details on how to (efficiently) compute the scores are postponed to Section 5.4. The following scheme for generating the rules is a direct consequence of the definitions of the sc-2-path and the si-2-path (cf. the Definitions 4.13 and 4.14): for each relative score $g(a, b, \mathcal{S}) < 0$ with $a, b \in \mathcal{V}$, where the set $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$ is not empty, a c-rule $CR([a, b], \mathbf{E}, \mathbf{C}, \mathbf{I})$ is generated with

- $\mathbf{E} = \{[a, s] : s \in \mathcal{S}\}$,
- $\mathbf{C} = \{[b, s] : s \in \mathcal{S}\}$,
- $\mathbf{I} = \{[b, s] : s \in \mathcal{S}\} \cup \{[s_1, s_2] : s_1, s_2 \in \mathcal{S}, s_1 \neq s_2\}$.

While such c-rules correspond to sc-paths (cf. Definition 4.13), si-paths are accounted for by i-rules as follows: regarding two variables $a, b \in \mathcal{V}$, an i-rule $IR([a, b], \mathbf{E}, \mathbf{I})$ is created if there are two relative scores $g(a, b, \mathcal{S}) < 0$ and $g(b, a, \mathcal{S}') < 0$ with disjoint and non-empty sets $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{V} \setminus \{a, b\}$. The condition sets \mathbf{E} and \mathbf{I} are given by

- $\mathbf{E} = \{[a, s] : s \in \mathcal{S}\} \cup \{[b, s'] : s' \in \mathcal{S}'\}$,
- $\mathbf{I} = \{[b, s] : s \in \mathcal{S}\} \cup \{[a, s'] : s' \in \mathcal{S}'\} \cup \{[s_1, s_2] : s_1, s_2 \in \mathcal{S} \cup \mathcal{S}', s_1 \neq s_2\}$.

Note that, in this definition of an i-rule, we require $\mathcal{S} \cap \mathcal{S}' = \emptyset$. In case that $\mathcal{S} \cap \mathcal{S}' = \mathcal{T} \neq \emptyset$, both the edges $a \sim t$ and $b \sim t$ ($t \in \mathcal{T}$) have to be present. Since this implies the existence of an sc-path between a and b , there is already a c-rule accounting for this case. The restriction $\mathcal{S} \cap \mathcal{S}' = \emptyset$ hence helps avoid the creation of essentially identical rules. Apart from that, the i-rule $IR([a, b], \emptyset, \emptyset)$ is generated for every $a, b \in \mathcal{V}$ with $g(a, b, \emptyset) < 0$. This is because the edge $a \sim b$ can be absent without requiring any edges or paths to be present. Positive scores, i.e. $g(a, b, \mathcal{S}) > 0$, do not entail the creation of a rule. When the used scoring function g is score-equivalent, one may not forget to generate the rules corresponding to both the scores $g(b, a, \mathcal{S}) < 0$ and $g(a, b, \mathcal{S}) < 0$.

All the rules are collected in the set of rules \mathfrak{R} . In general, there can be several different sets S for each edge $a \sim b$ such that $g(a, b, S) < 0$ or $g(b, a, S) < 0$. This implies that there can be many different rules concerning an edge $a \sim b$. Since only a negative relative score entails the creation of a rule, it is clear that there is no rule in \mathfrak{R} corresponding to a certainly-present edge $a \sim b$, for which holds that $\forall S \subseteq \mathcal{V} \setminus \{a, b\} : g(a, b, S) > 0 \wedge g(b, a, S) > 0$.

Since the definitions of the rules follow immediately from the sc-2-path and the si-2path, it is clear that a skeleton complies with the necessary si-path condition (cf. Proposition 4.12) if a rule is satisfied for each absent edge.

5.2 Simplifying the Rules

The minimal skeletons can be determined on the basis of the set of rules \mathfrak{R} , since the latter represents the necessary path condition in the algorithm. In general, it is not obvious which edges are absent in the minimal skeletons. The only exceptions are certainly-present and certainly-absent edges (cf. Proposition 4.2). The task is hence to find out about the presence or absence of the remaining edges. This can be done by two equivalent, but in some sense opposite, approaches. Let us first consider the more intuitive procedure. It is illustrated in Figure 5.1.

In this simplistic example, it may not be clear immediately whether the edges $a \sim b$ and $a \sim d$ have to be present or absent in a minimal skeleton. Instead of considering a graph, let us focus on the rules concerning the edge $a \sim b$, i.e. the rules (I) and (II). According to rule (II), if the edge $a \sim b$ is absent then the edge $b \sim d$ and an sc-path between a and d have to be present. While the edge $b \sim d$ is certainly present, the existence of the sc-path between a and d is not apparent. Due to the recursive definition of an sc-path, however, one can use rule (III) regarding the edge or sc-path between a and d . When the c-rule (III) is inserted into rule (II), as depicted in Figure 5.1, then the new rule (V), $CR([a, b], \mathbf{E}, \mathbf{C}, \mathbf{I})$, is such that the condition sets \mathbf{E} , \mathbf{C} and \mathbf{I} contain solely edges which are certainly present. Hence, this rule is fulfilled and the minimal skeleton does not contain the edge between a and b .

This example illustrates that the absence of the edge between a and b can be determined by considering solely the set of rules \mathfrak{R} . A graph was not used in this process. Having simplified the rules, the minimal skeleton(s) can immediately be specified.

5.2.1 Removing the Conditions Apparently Fulfilled

The computational disadvantage of the above procedure is that the number of rules as well as their complexity increase, because new rules are generated. Let us hence turn to the alternative procedure which is computationally more efficient. The idea is as follows: the condition sets of the rules $CR([a, b], \mathbf{E}, \mathbf{C}, \mathbf{I})$ or $IR([a, b], \mathbf{E}', \mathbf{I}')$ often require such edges or paths to be present which are indeed present, e.g. certainly-present edges. Since the presence of the latter edges is guaranteed, they need not be stated explicitly in the condition sets. The rules can thus be simplified by eliminating these edges from the condition sets \mathbf{E} , \mathbf{C} and \mathbf{I} , as an edge is also an

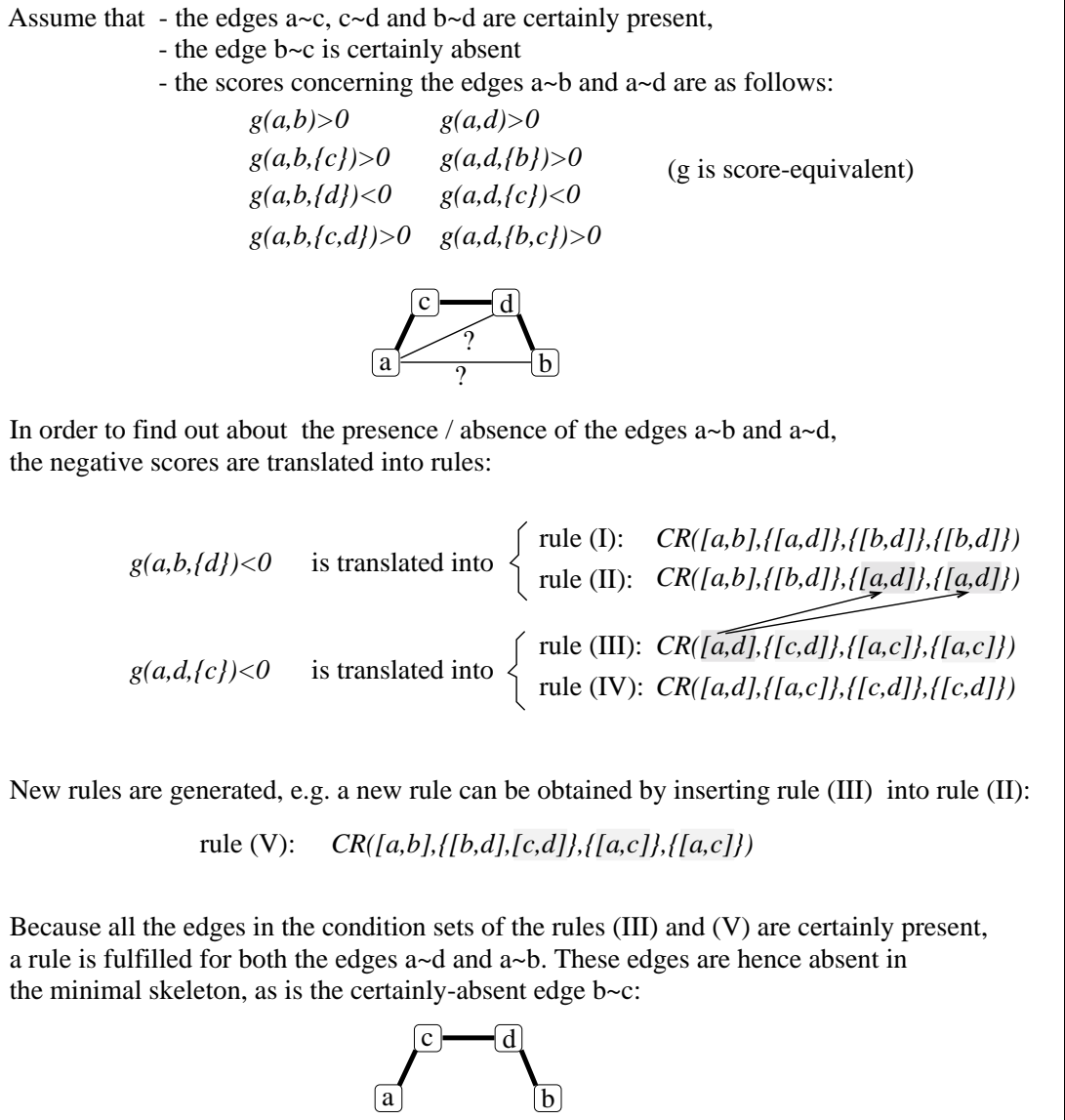


Figure 5.1: The absence of the edges $a \sim b$ and $a \sim d$ from the minimal skeleton can be induced by inserting rules into one another.

sc-path and an si-path. Let us reconsider the example in Figure 5.1 for illustration. Since the edges $a \sim c$, $c \sim d$ and $b \sim d$ are certainly present, they can be removed from the condition sets \mathbf{E} , \mathbf{C} and \mathbf{I} of each rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$. This is depicted in Figure 5.2. After this elimination, particularly simple rules of the kind $CR([a, d], \emptyset, \emptyset, \emptyset)$ occur. This has two consequences. First, such a rule states that the edge $a \sim d$ is absent in *all* the minimal skeletons. The reason is that all its condition sets are empty, and hence this rule is always fulfilled. This implies also that the edges and paths required by the original rule are present in *all* the minimal skeletons.

The rules, as depicted in the previous example, can be simplified in an equivalent, but computationally more efficient way:

rule (I): $CR([a,b],[[a,d]],[[b,d]],[[b,d]])$

rule (II): $CR([a,b],[[b,d]],[[a,d]],[[a,d]])$

rule (III): $CR([a,d],[[c,d]],[[a,c]],[[a,c]])$

rule (IV): $CR([a,d],[[a,c]],[[c,d]],[[c,d]])$

Because the edges $a \sim c$, $c \sim d$ and $b \sim d$ are certainly present, they can be removed from the condition sets in each rule:

rule (I'): $CR([a,b],[[a,d]],\emptyset,\emptyset)$

rule (II'): $CR([a,b],\emptyset,[[a,d]],[[a,d]])$

rule (III'): $CR([a,d],\emptyset,\emptyset,\emptyset)$

rule (IV'): $CR([a,d],\emptyset,\emptyset,\emptyset)$

\Rightarrow the edge $a \sim d$ is definitely absent and
an sc-path between a and d is definitely present

The simplified rules (III') and (IV') can now be substituted into the rule (II'):

rule (II''): $CR([a,b],\emptyset,\emptyset,\emptyset)$

\Rightarrow the edge $a \sim b$ is definitely absent

Hence, the minimal skeleton is:

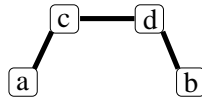


Figure 5.2: Simplifying the rules is an equivalent, but computationally more efficient alternative to generating new rules like in Figure 5.1.

The second consequence is hence that an sc-path is present between a and d , even if the edge $a \sim d$ is absent, because the simplified rule is a c -rule. This implies that c -rules of the kind $CR(X, \emptyset, \emptyset, \emptyset)$ can be used for further simplifying those rules which contain X in one of their condition sets C or I . This is achieved by removing also X from those sets. As more and more rules of the kind $CR(Y, \emptyset, \emptyset, \emptyset)$ occur during the process of simplifying the rules, they can be used for simplifying others. This is depicted in the second step in Figure 5.2, where the edge $a \sim b$ is found to be absent in minimal skeletons.

For brevity, we call an edge *definitely absent* or *definitely present* if it is absent or present in *all* the minimal skeletons. In particular, every certainly-present or certainly-absent edge is also definitely-present or definitely-absent. Conversely, a definitely-absent edge need not be certainly absent. In Figure 5.2, for instance, the edges $a \sim b$ and $a \sim d$ are definitely but not certainly absent. The notion of definitely-present (absent) edges is thus more general than the one of certainly-present (absent) edges. The rules in \mathfrak{R} can be simplified to help find

definitely-absent edges as follows:

(i) Every c-rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ with an $Y \in \mathbf{E}$ such that

- Y is a certainly-present edge

can be changed to the simpler c-rule $CR(X, \mathbf{E} \setminus \{Y\}, \mathbf{C}, \mathbf{I})$.

(ii) Every c-rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ with an $Y \in \mathbf{C}$ such that

- Y is a certainly-present edge, or
- there is a c-rule $CR(Y, \emptyset, \emptyset, \emptyset)$

can be replaced by the c-rule $CR(X, \mathbf{E}, \emptyset, \mathbf{I})$.

(iii) Every c-rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ with an $Y \in \mathbf{I}$ such that

- Y is a certainly-present edge, or
- there is a c-rule $CR(Y, \emptyset, \emptyset, \emptyset)$, or
- there is an i-rule $IR(Y, \emptyset, \emptyset, \emptyset)$

can be altered into the c-rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I} \setminus \{Y\})$.

(iv) Every i-rule $IR(X, \mathbf{E}, \mathbf{I})$ can be replaced by a simpler i-rule analogously to the procedures (i) and (iii).

In procedure (ii), the set \mathbf{C} simplifies to the empty set, because *at least* one sc-path has to be present according to the Definition 4.13 of an sc-2-path. Consequently, once an sc-path $Y \in \mathbf{C}$ is determined to be present, all the alternative sc-paths in \mathbf{C} need not be considered any more, and can hence be removed. In (iii), a certainly-absent edge need not be mentioned explicitly, unlike a certainly-present one. This is because a certainly-absent edge $a \sim b$ is translated into rules, particularly into the i-rule $IR([a, b], \emptyset, \emptyset, \emptyset)$, while a certainly-present edge is not reflected by a rule. In the Definitions 4.13 and 4.14 of an sc-2-path $SC_2([a, b], \mathbf{W})$ and of an si-2-path $SI_2([a, b], \mathbf{W})$ between two variables $a, b \in \mathcal{V}$, it is not allowed that the pairs of variables in the set \mathbf{W} are present along any of the paths required by their recursive definitions. This is automatically ensured by the above schemes applying to rules, because the simplification process starts out with certainly-present and certainly-absent edges, and continues only with definitely-absent edges. Thus, the above procedures comply with the necessary path condition and the definitions of sc-paths and si-paths in the previous chapter. They can hence be used for finding (some of the) definitely-absent edges.

In the remainder of this section, let us reconsider some of the previous examples in the light of rules. First, in the simplistic example in Figure 4.4, the score $g(a, c, \{b\}) < 0$ entails the rules $CR([a, c], \{[a, b]\}, \{[b, c]\}, \{[b, c]\})$ and $CR([a, c], \{[b, c]\}, \{[a, b]\}, \{[a, b]\})$. They can be simplified, because the edge $a \sim b$ is certainly present (cf. Proposition 4.2). The resulting rules regarding the edge $a \sim c$ are denoted in Table 5.1: neither one of these two rules simplifies

simplified rules
$CR([a, c], \emptyset, \{[b, c]\}, \emptyset)$
$CR([a, c], \{[b, c]\}, \emptyset, \emptyset)$

Table 5.1: The rules regarding the edge $a \sim c$ after their simplification. This continues the example in Figure 4.4.

to $CR([a, c], \emptyset, \emptyset, \emptyset)$. Both state that the edge $a \sim c$ can only be absent if the edge $b \sim c$ is present. Since the latter is certainly absent, the rules yield that the edge $a \sim c$ has to be present in the minimal skeleton, which agrees with the result found previously.

The second example is slightly more involved, namely the one shown in Figure 4.5. The scores $g(a, c, \{b\}) < 0$ and $g(b, c, \{a\}) < 0$ transform into the rules denoted in Table 5.2. Only the certainly-present edge $a \sim b$ leads to a simplification of the condition sets (cf. Table 5.2). Obviously, neither one of the edges $a \sim c$ and $b \sim c$ can be absent in all the minimal skeletons. In fact, the simplified rules imply that the edge $a \sim c$ can only be absent if $b \sim c$ is present, and vice versa. This is in accordance with the results in Figure 4.5.

Regarding the third example, the rules resulting from the scores in Figure 4.8 are given in Table 5.3. The certainly-present edges $a \sim b$ and $a \sim c$ entail a simplification of the rules. It follows that the edges $a \sim d$, $b \sim d$ and $c \sim d$ are not definitely-absent. Furthermore, it is apparent that a rule is fulfilled for each edge when the single edge $a \sim d$ is present, or when *both* the edges $b \sim d$ and $c \sim d$ are contained in the minimal skeleton. Again, this result agrees with the one obtained in Figure 4.8.

These examples illustrate how the set of rules can be simplified greatly so that the structure of the minimal skeletons can immediately be read off the rules. In general, however, it is not possible to induce all the definitely-absent edges by means of the above procedures. This is because the given simplifications are merely simple schemes rather than a necessary and sufficient condition for determining such edges. A further simplification of the rules can be achieved by the more powerful procedure described in the next section. As it is computationally more involved than the above procedures, the above procedures might be applied in a pre-processing step. Since a necessary and sufficient condition can be very involved due to the recursive definition of the paths, this is accounted for in the subsequent step described in Section 5.3.2.

rules generated according to the scores	simplified rules
$CR([a, c], \{[a, b]\}, \{[b, c]\}, \{[b, c]\})$	$CR([a, c], \emptyset, \{[b, c]\}, \{[b, c]\})$
$CR([a, c], \{[b, c]\}, \{[a, b]\}, \{[a, b]\})$	$CR([a, c], \{[b, c]\}, \emptyset, \emptyset)$
$CR([b, c], \{[a, b]\}, \{[a, c]\}, \{[a, c]\})$	$CR([b, c], \emptyset, \{[a, c]\}, \{[a, c]\})$
$CR([b, c], \{[a, c]\}, \{[a, b]\}, \{[a, b]\})$	$CR([b, c], \{[a, c]\}, \emptyset, \emptyset)$

Table 5.2: The rules before and after their simplification. This continues the example depicted in Figure 4.5.

rules generated according to the scores	simplified rules
$CR([b, d], \{[a, b]\}, \{[a, d]\}, \{[a, d]\})$	$CR([b, d], \emptyset, \{[a, d]\}, \{[a, d]\})$
$CR([b, d], \{[a, d]\}, \{[a, b]\}, \{[a, b]\})$	$CR([b, d], \{[a, d]\}, \emptyset, \emptyset)$
$CR([c, d], \{[a, c]\}, \{[a, d]\}, \{[a, d]\})$	$CR([c, d], \emptyset, \{[a, d]\}, \{[a, d]\})$
$CR([c, d], \{[a, d]\}, \{[a, c]\}, \{[a, c]\})$	$CR([c, d], \{[a, d]\}, \emptyset, \emptyset)$
$CR([a, d], \{[a, b], [a, c]\}, \{[b, d], [c, d]\},$ $\{[b, d], [c, d], [b, c]\})$	$CR([a, d], \emptyset, \{[b, d], [c, d]\}, \{[b, d], [c, d]\})$
$CR([a, d], \{[b, d], [c, d]\}, \{[a, b], [a, c]\},$ $\{[a, b], [a, c], [b, c]\})$	$CR([a, d], \{[b, d], [c, d]\}, \emptyset, \emptyset)$

Table 5.3: These rules refer to the example in Figure 4.8. We omitted the rules concerning certainly-absent edges.

5.2.2 Condition Graph

The rules can be further simplified by the procedures presented in this section. The relationships among the rules have thus to be considered in more detail. This can be achieved by employing a meta-graph, which we call a *condition graph*. Each pair of variables $X = [a, b]$, $a, b \in \mathcal{V}$, corresponds to a node in the condition graph. Hence, if a skeleton comprises n variables, the condition graph contains $\binom{n}{2}$ nodes. The condition graph is a directed graph. Like in a DAG, we define the parents $\mathbf{PA}(X)$ of a node X in the condition graph to be the set of nodes such that there is a directed arc from each parent to X .¹ A condition graph is built according to a set of rules \mathfrak{R} in such a way that the parents of a node X are given by

$$\mathbf{PA}(X) = \{Y : CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I}) \in \mathfrak{R}, Y \in \mathbf{E} \cup \mathbf{C} \cup \mathbf{I}\} \cup \{Y : IR(X, \mathbf{E}, \mathbf{I}) \in \mathfrak{R}, Y \in \mathbf{E} \cup \mathbf{I}\}.$$

This means that, if a pair of variables Y is in one of the condition sets of a rule $CR(X, \cdot, \cdot, \cdot)$ or $IR((X, \cdot, \cdot))$ then there is a directed arc $Y \rightarrow X$ in the condition graph. In contrast to DAGs, directed cycles are allowed to be present in a condition graph. In fact, directed cycles are one of the most interesting structures in such a graph. This is illustrated in Figure 5.3: in the condition graph (I), the arc $[b, c] \rightarrow [a, c]$ indicates that the edge $a \sim c$ can only be absent in the minimal skeleton if the edge $b \sim c$ is present. Since the later is certainly absent in the example shown in Figure 4.4, the edge $a \sim c$ has to be present according to the rules. In the condition graphs (II) and (III) in Figure 5.3, the directed cycles indicate that the presence of the involved edges depends on each other: in (II), either one of the edges $a \sim c$ or $b \sim c$ has to be present in the minimal skeletons (cf. also Figure 4.5), whereas in (III) either the single edge $a \sim b$ or both the edges $c \sim d$ and $b \sim c$ have to be present (cf. also Figure 4.8). This shows that directed cycles are related to uncertainty regarding the presence of edges in minimal skeletons (cf. Section 4.3.1).

¹For clarity, we use "variable" and "edge" in skeletons, whereas "node" and "arc" refer to the condition graph.

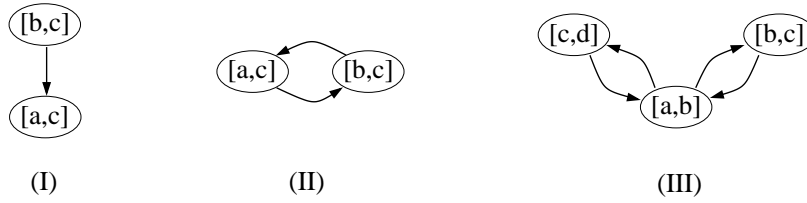


Figure 5.3: The condition graphs (I), (II) and (III) correspond to the simplified rules in the Tables 5.1, 5.2 and 5.3, respectively. Only the relevant nodes are shown.

With this definition of the condition graph, we can now further simplify the set of rules \mathfrak{R} . First, let us focus on the condition set \mathbf{I} of a rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ or $IR(X, \mathbf{E}, \mathbf{I})$. The general procedure is to cycle through the following three steps until no more simplification can be achieved:

- (i) build up the condition graph according to the current set of rules,
- (ii) choose a rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ or $IR(X, \mathbf{E}, \mathbf{I})$ and a $Y \in \mathbf{I}$,
- (iii) if the condition graph does not exhibit a directed cycle involving the edge $Y \rightarrow X$ then the rule chosen in (ii) can be simplified to $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I} \setminus \{Y\})$ or $IR(X, \mathbf{E}, \mathbf{I} \setminus \{Y\})$, respectively.

The idea is that the absence of a directed cycle involving the edge $Y \rightarrow X$ indicates the presence of an si-path $SI(Y, \{X\})$ in all the minimal skeletons. Besides the condition set \mathbf{I} , also the condition set \mathbf{C} can be simplified. However, this is only possible under additional restrictions, because the sc-path is more specific than the si-path. This can be achieved by cycling through the following steps:

- (i) build up the condition graph according to the current set of rules,
- (ii) choose a rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and a $Y \in \mathbf{C}$,
- (iii) this rule can be simplified to $CR(X, \mathbf{E}, \emptyset, \mathbf{I})$ if
 - the condition graph does not exhibit a directed cycle involving the edge $Y \rightarrow X$, and
 - the rules regarding Y and all its ancestors $\mathbf{AN}(Y)$ are all c-rules.

In a condition graph, let the ancestors $\mathbf{AN}(Y)$ of a node Y be defined to contain all those $Z \in \mathbf{AN}(Y)$ such that there is a directed path $Z \rightarrow \dots \rightarrow Y$. In this procedure, the additional restriction on Y and $\mathbf{AN}(Y)$ ensures that the paths between y and z ($Y = [y, z]$) are sc-paths rather than si-paths in a minimal skeleton. The absence of a cycle involving $Y \rightarrow X$ indicates, like above, that there is an sc-path $SC(Y, \{X\})$, as required by Definition 4.13. The simplification of $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ to $CR(X, \mathbf{E}, \emptyset, \mathbf{I})$ is a consequence of the fact that the presence

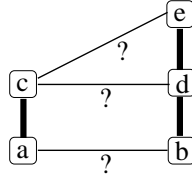


Figure 5.4: The presence or absence of the edges $a \sim b$, $c \sim d$ and $c \sim e$ is not clear initially.

of *at least* one sc-path is required (cf. Definition 4.13). We note that chances to simplify the condition set \mathbf{C} are very limited in general. This is because i-rules are also contained in \mathfrak{R} in general. The set of edges \mathbf{E} can only be simplified if there is a $Y = [y, z] \in \mathbf{E}$ corresponding to a certainly-present edge $y \sim z$. Indeed, the set \mathbf{E} is simplified this way by procedure (i) in the previous section. Hence, the condition graph does not yield an additional simplification of the set \mathbf{E} . Therefore, one can expect that only the condition set \mathbf{I} of a rule $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ or $IR(X, \mathbf{E}, \mathbf{I})$ can be further simplified in many cases.

This is illustrated in the following example (cf. Figure 5.4). Assume that the edges $a \sim c$, $b \sim d$ and $d \sim e$ are certainly present, whereas the edges $a \sim d$, $a \sim e$, $b \sim c$ and $b \sim e$ are certainly absent. Moreover, let all the relative scores concerning the edges $a \sim b$, $c \sim d$ and $c \sim e$ be positive, except for the scores $g(a, b, \{c\}) < 0$, $g(a, b, \{d\}) < 0$, $g(c, d, \{e\}) < 0$ and $g(c, e, \{d\}) < 0$. The relative scoring function g is assumed to be score equivalent. Let us focus on the presence or absence of the edges $a \sim b$, $c \sim d$ and $c \sim e$ in the following. First, the scores are translated into the corresponding rules (cf. Section 5.1). Second, with the help of the certainly-present and certainly-absent edges, the rules can be simplified as described in the previous section. The results are shown in Table 5.4. It is obvious that the edges $c \sim d$ and $c \sim e$ cannot be absent simultaneously in a minimal skeleton. However, it is not apparent

rules generated according to the scores	simplified rules
$CR([a, b], \{[a, c]\}, \{[b, c]\}, \{[b, c]\})$	$CR([a, b], \emptyset, \{[b, c]\}, \emptyset)$
$CR([a, b], \{[b, c]\}, \{[a, c]\}, \{[a, c]\})$	$CR([a, b], \{[b, c]\}, \emptyset, \emptyset)$
$CR([a, b], \{[b, d]\}, \{[a, d]\}, \{[a, d]\})$	$CR([a, b], \emptyset, \{[a, d]\}, \emptyset)$
$CR([a, b], \{[a, d]\}, \{[b, d]\}, \{[b, d]\})$	$CR([a, b], \{[a, d]\}, \emptyset, \emptyset)$
$IR([a, b], \{[a, c], [b, d]\}, \{[a, d], [b, c], [c, d]\})$	$IR([a, b], \emptyset, \{[c, d]\})$
$IR([a, b], \{[a, d], [b, c]\}, \{[a, c], [b, d], [c, d]\})$	$IR([a, b], \{[a, d], [b, c]\}, \{[c, d]\})$
$CR([c, d], \{[d, e]\}, \{[c, e]\}, \{[c, e]\})$	$CR([c, d], \emptyset, \{[c, e]\}, \{[c, e]\})$
$CR([c, d], \{[c, e]\}, \{[d, e]\}, \{[d, e]\})$	$CR([c, d], \{[c, e]\}, \emptyset, \emptyset)$
$CR([c, e], \{[d, e]\}, \{[c, d]\}, \{[c, d]\})$	$CR([c, e], \emptyset, \{[c, d]\}, \{[c, d]\})$
$CR([c, e], \{[c, d]\}, \{[d, e]\}, \{[d, e]\})$	$CR([c, e], \{[c, d]\}, \emptyset, \emptyset)$

Table 5.4: The rules before and after their simplification. The rules of certainly-absent edges are omitted.

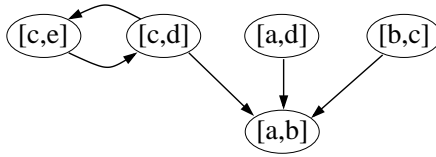


Figure 5.5: The condition graph built according to the simplified rules given in Table 5.4 (only the relevant nodes are shown).

from the simplified rules whether the edge $a \sim b$ has to be present. Let us hence consider the condition graph corresponding to the simplified rules. As described above, the parents of the nodes $[a, b], [c, d]$ and $[c, e]$ are given by $\mathbf{PA}([a, b]) = \{[b, c], [a, d], [c, d]\}$, $\mathbf{PA}([c, d]) = \{[c, e]\}$ and $\mathbf{PA}([c, e]) = \{[c, d]\}$, respectively. The resulting condition graph is shown in Figure 5.5. It exhibits a directed cycle involving the nodes $[c, d]$ and $[c, e]$. In this example, the directed cycle is due to the fact that either one of the edges $c \sim d$ or $c \sim e$ has to be present in the minimal skeletons. In contrast, none of the arcs pointing towards $[a, b]$ is involved in a directed cycle. This implies that the si-paths $SI([a, d], \{[a, b]\})$, $SI([b, c], \{[a, b]\})$ and $SI([c, d], \{[a, b]\})$ have to be present in all the minimal skeletons. According to the definition of an si-path (cf. Section 4.3), the three pairs $[a, d]$, $[b, c]$ and $[c, d]$ can thus be removed from the condition sets \mathbf{I} of the rules regarding $[a, b]$. Since there is only $[c, d]$ in these condition sets \mathbf{I} (cf. Table 5.4), $[a, d]$ and $[b, c]$ do not lead to a simplification. For this reason, only the rules $IR([a, b], \emptyset, \{[c, d]\})$ and $IR([a, b], \{[a, d], [b, c]\}, \{[c, d]\})$ can be reduced to $IR([a, b], \emptyset, \emptyset)$ and $IR([a, b], \{[a, d], [b, c]\}, \emptyset)$, respectively (cf. Table 5.4). This simplification corresponds to the above procedure employing the condition graph. The simplified rule $IR([a, b], \emptyset, \emptyset)$ reveals that the edge $a \sim b$ is definitely absent, leading to the minimal skeletons and to the condition graph shown in Figure 5.6.

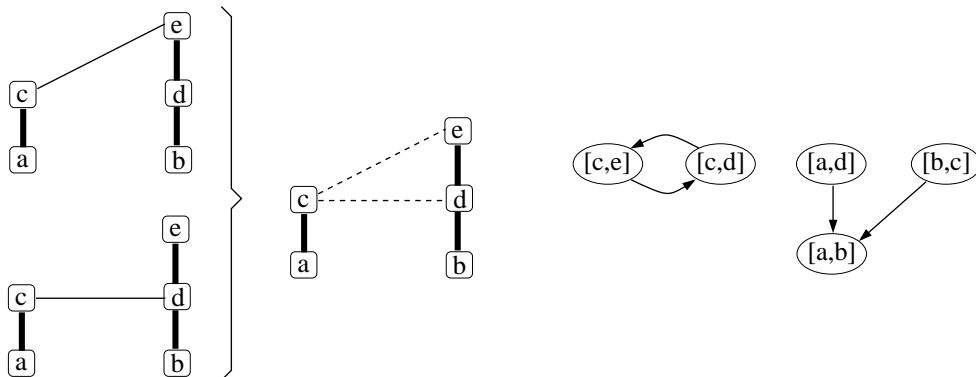


Figure 5.6: By means of the condition graph, it could be determined that the edge $a \sim b$ is definitely absent in the (two) minimal skeletons (left), which are summarized in the graph shown in the center. The simplified condition graph, which splits into two disconnected parts, is depicted on the right.

5.2.3 Reducing the Number of Rules

Besides simplifying the condition sets, also the number of rules can be reduced. Since an edge is absent from a minimal skeleton if *at least one* rule is fulfilled, a complex rule which contains a simpler one can be removed from the set of rules \mathfrak{R} . This is because the simpler rule is always fulfilled when the more complex one is satisfied. This leads immediately to the following simplifications:

- (i) if $CR(X, \mathbf{E}_1, \emptyset, \mathbf{I}_1), CR(X, \mathbf{E}_2, \mathbf{C}_2, \mathbf{I}_2) \in \mathfrak{R}$ such that $\mathbf{E}_1 \subseteq \mathbf{E}_2$ and $\mathbf{I}_1 \subseteq \mathbf{I}_2$ then remove the latter rule from \mathfrak{R} ,
- (ii) if $CR(X, \mathbf{E}_1, \emptyset, \mathbf{I}_1), IR(X, \mathbf{E}_2, \mathbf{I}_2) \in \mathfrak{R}$ such that $\mathbf{E}_1 \subseteq \mathbf{E}_2$ and $\mathbf{I}_1 \subseteq \mathbf{I}_2$ then eliminate the latter one from \mathfrak{R} ,
- (iii) if $IR(X, \mathbf{E}_1, \mathbf{I}_1), IR(X, \mathbf{E}_2, \mathbf{I}_2) \in \mathfrak{R}$ such that $\mathbf{E}_1 \subseteq \mathbf{E}_2$ and $\mathbf{I}_1 \subseteq \mathbf{I}_2$ then exclude the latter one from \mathfrak{R} .

Regarding (i), it is important that the simpler rule has an empty set \mathbf{C} of sc-paths, indicating that the required sc-path has already been induced to be present. Hence, an additional sc-path among the candidates in the possibly larger set \mathbf{C}_2 need not be found. In (ii), the empty set $\mathbf{C} = \emptyset$ implies again that an sc-path has already been found. Note also that the c-rule and the i-rule cannot be interchanged in (ii), since the sc-path is more specific than the si-path. Of course, these three procedures can be extended in various ways, as they are apparently necessary conditions for finding minimal skeletons. It is obvious that the set of rules before and after applying (i) through (iii) yields the same minimal skeletons.

For illustration, let us reconsider the simplistic example in the Figures 5.1 and 5.2. However, let also the score $g(a, d, \{b, c\})$ be negative, besides $g(a, d, \{c\})$. Then the former entails the rule $CR([a, d], \{[a, b], [a, c]\}, \{[b, d], [c, d]\}, \{[b, c], [b, d], [c, d]\})$. Since the rule concerning $g(a, d, \{c\}) < 0$ can be simplified to $CR([a, d], \emptyset, \emptyset, \emptyset)$ the former rule obviously contains the simplified one, and the more complex rule can hence be removed from the set of rules \mathfrak{R} according to (i).

Note that, once a c-rule regarding an edge X has been completely simplified, i.e. it is of the kind $CR(X, \emptyset, \emptyset, \emptyset)$, then all other rules, i.e. c-rules *and* i-rules, concerning this edge can be removed. In contrast, solely the i-rules regarding an edge X can be eliminated from \mathfrak{R} in general after the i-rule $IR(X, \emptyset, \emptyset)$ has been induced, as an si-path is less specific than an sc-path. This is, however, only relevant when X appears in the condition sets of other rules, i.e. when arcs point away from the node X in the condition graph. This suggests a further simplification: if there is an i-rule $IR(X, \emptyset, \emptyset)$, and if the node X is purely endogenous, i.e. no arcs point away from it, then all other rules concerning the edge X , including the c-rules, can be removed from the set of rules \mathfrak{R} . In the example shown in Figure 5.6, the condition graph simplifies to the one in Figure 5.7.

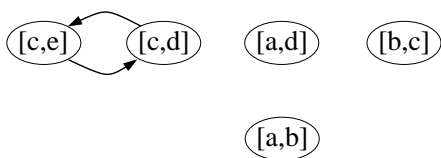


Figure 5.7: Due to the i-rule $IR([a, b], \emptyset, \emptyset)$, all the arcs pointing to the purely endogenous node $[a, b]$ can be removed from the condition graph.

5.3 Ambiguous Regions and Summary Graph

The previous section was concerned with inducing (most of) the edges which are absent in *all* the minimal skeletons, i.e. definitely-absent edges. In the following, we focus on the remaining edges X , namely the ones for which the steps described so far did not yield a completely simplified rule. The aim is to find out whether such an edge X is absent or present in a minimal skeleton. The crucial difference between such an edge and a definitely-absent one is that the former might be present in *some* of the minimal skeletons while absent in *some* others, whereas the latter edges are absent in *all* the minimal skeletons. Let us designate such edges as *ambiguous edges*. For example (cf. Figure 4.8), the edge $b \sim c$ is definitely absent, whereas the ambiguous edges $a \sim d$, $b \sim d$ and $c \sim d$ are present in either one of the minimal skeletons.

When there is more than one minimal skeleton complying with the necessary path condition (cf. Proposition 4.12), the various graphs differ in the presence of some ambiguous edges. Hence, ambiguous edges play a crucial role in model uncertainty regarding the presence of edges. In order to get an overview of the possible structures of the different skeletons, the ambiguous edges and the certainly-present ones can be depicted in a single graph. Let such a graph be named *summary graph*, because it summarizes the possible structures of all the induced skeletons. In a summary graph, solid lines indicate edges which are certainly present, while ambiguous edges are denoted in different line styles. Missing edges correspond to definitely-absent ones. All the induced skeletons contain the edges found to be certainly-present, while they differ regarding the ambiguous edges. Summary graphs have already been used in the Figures 4.5, 4.7 and 5.6, where their meaning was intuitively clear. We have first published this concept of visualizing model uncertainty by means of a single graph in [144]. Subsequently, it was also used in the PRONEL project [120] and in [85]. In the latter, uncertainty regarding the presence of certain structures was calculated by Bayesian model-averaging. Although this is straightforward from a theoretical point of view, it is computationally very involved in general.

In Figure 5.8 (II), a summary graph with 10 variables is shown. In this simplistic example, we have essentially combined the Figures 4.4, 4.5 and 4.8 into one graph so that the reader is already familiar with the individual parts. The condition graph looks like in Figure 5.9, and the rules concerning the ambiguous edges are understood to be analogous to the previous examples. The relations among the rules, as shown in the condition graph in Figure 5.9, reveal that the presence of the ambiguous edges $t \sim v$ and $u \sim v$ can depend on each other. However, the presence of the latter edges is *independent* of the presence of the ambiguous edges $w \sim z$, $x \sim$

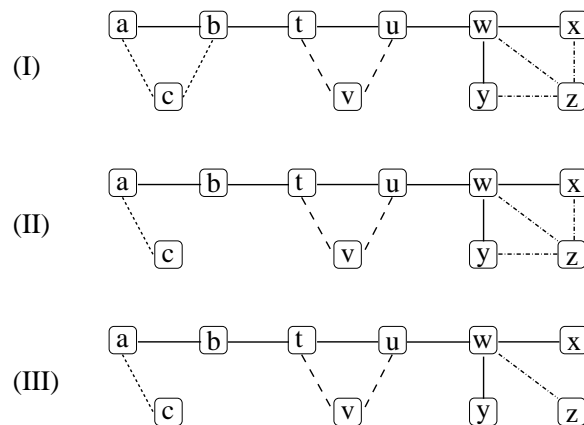


Figure 5.8: The summary graph of a simplistic domain with 10 variables at different stages of simplification: (I) preliminary, (II) minimal, and (III) parsimonious summary graph.

z and $y \sim z$. This suggests to collect all the edges whose presence can depend on each other in the same set, which we call an *ambiguous region*. Besides the cycles in the condition graph, the connected components are hence very important. A connected component comprises all those nodes which are connected with one another by an *undirected* path when the directions of the edges are ignored. Hence, the presence of edges belonging to different ambiguous regions is by definition independent of each other. Particularly in skeletons involving a reasonably large number of variables, we found ambiguous regions to occur very often in our computer experiments (cf. the Sections 5.6 and 6.3). Ambiguous edges belonging to the same ambiguous region are shown in the same line-style in the summary graph. Figure 5.8 (II) depicts three ambiguous regions, namely the first one contains the single edge $a \sim c$, the second region comprises the edges $t \sim v$ and $u \sim v$, and the third one is made up of the edges $w \sim z$, $x \sim z$ and $y \sim z$.

The independence of different ambiguous regions has two important consequences. First, a summary graph can be understood more easily than a list of all the induced skeletons. This is because the induced skeletons are identical regarding the definitely-present edges, and they differ concerning the ambiguous edges only. Additionally, the independence of different ambiguous regions helps in understanding the variability of the graph. However, the various structures present in each ambiguous region can not be read off a summary graph. For instance, the summary graph in Figure 5.8 (II) does not indicate that the minimal structures in the rightmost ambiguous region involve either the edge $w \sim z$ or both the edges $x \sim z$ and $y \sim z$. This information can in parts be read off the condition graph, or it can be given, for instance, by enumerating the different minimal structures in each ambiguous region. In the PRONEL project, this information could be explored interactively by the user [120]. Second, the independence of different ambiguous regions reduces the computational effort considerably, as it allows to search for the various structures separately in each ambiguous region. All the different minimal skeletons can eventually be obtained by combining the different structures found in the various ambiguous regions. Summing up, even a large number of different minimal skeletons

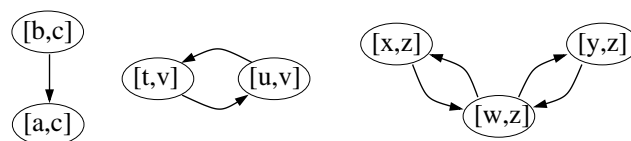


Figure 5.9: The condition graph shows the relations among the rules in the simplistic domain depicted in Figure 5.8.

can efficiently be computed due to the independence of the ambiguous regions, and they can all be visualized in a single summary graph.

5.3.1 Definitions

Having motivated the concept of ambiguous regions in the summary graph, let us now define a *preliminary ambiguous region*. It is based on a set of rules \mathfrak{R} simplified by the procedures described in the previous sections. A preliminary ambiguous region is the set \mathbf{A} of all nodes X which belong to the same connected component in the condition graph. Let two nodes belong to the same connected component in a directed graph if there is a path between them in the undirected version of that graph. Since different ambiguous regions are not connected in the condition graph by definition, they are independent. The set of rules \mathfrak{R} can hence be divided into subsets of rules $\mathfrak{R}_i^{\mathbf{A}}$ where each of which corresponds to a different ambiguous region \mathbf{A}_i . Hence, only the presence of the edges involved in the same region can depend on each other.

According to this definition, a preliminary ambiguous region can contain also definitely-absent edges, besides ambiguous ones. A preliminary ambiguous region thus comprises all the edges whose presence *might* depend on each other in the minimal skeletons. In other words, it can also comprise edges which are not present in any minimal skeleton. In Figure 5.9, for example, the preliminary condition graph exhibits three preliminary ambiguous regions. The left one contains the edges $a \sim c$ and $b \sim c$ according to the condition graph in Figure 5.9, although the edge $b \sim c$ is definitely absent, as sketched in Figure 4.8. The *preliminary summary graph* is shown in Figure 5.8 (I).

A *minimal ambiguous region* comprises only those edges of a preliminary ambiguous region which are actually present in *at least one* of the minimal skeletons. A summary graph is minimal if it displays the minimal ambiguous regions. On the left hand side in Figure 5.8 (II), the minimal ambiguous region does not involve the edge $b \sim c$, as it is definitely absent. It solely contains the edge $a \sim c$. Since this edge is present in *all* the minimal skeletons, we also call it a *definitely-present* edge.

When applying the heuristics of parsimony (cf. Section 4.3.5), an ambiguous region might be reduced even further, leading to a *parsimonious ambiguous region*. Such an ambiguous region solely comprises the edges present in parsimonious skeletons, as defined in Section 4.3.5. This is illustrated in the ambiguous region on the right hand side in Figure 5.8 (III), where the minimal skeletons contain either the single edge $w \sim z$ or both the edges $w \sim x$ and $w \sim y$.

(cf. also Figure 4.8). Only the edge $w \sim z$ is, however, present in a parsimonious skeleton, and is hence present in the parsimonious ambiguous region shown in Figure 5.8 (III). In general, a *parsimonious summary graph* can be considerably simpler than a preliminary one. This is also confirmed by the experiments in Section 5.6.2, in particular in the Figures 5.17 and 5.18. This suggests to present the parsimonious summary graph to the user, rather than the preliminary one, since the latter tends to contain larger ambiguous regions, rendering the interpretation more difficult.

5.3.2 Search within each Ambiguous Region

After the different preliminary ambiguous regions have been determined, each of which can be considered separately due to their mutual independence. As a consequence, the complexity of determining the minimal skeletons is greatly reduced. In each of the ambiguous regions \mathbf{A}_i , the subset of rules $\mathfrak{R}_i^{\mathbf{A}}$ determines the different structures present in minimal skeletons. In general, it is too involved to read the structures of the minimal skeletons directly off the subset of rules. The following exact and approximate search strategies can be applied to find the different minimal or parsimonious structures also in the case of quite large ambiguous regions.

A minimal or parsimonious structure in an ambiguous region \mathbf{A}_i is a set of present edges such that for each absent edge $X \in \mathbf{A}_i$ a rule is fulfilled in $\mathfrak{R}_i^{\mathbf{A}}$. Additionally, this set of edges has to be minimal or parsimonious in the sense outlined in the Sections 4.3.4 and 4.3.5. Once these structures are known, the minimal or parsimonious skeletons can be constructed immediately by merging the minimal or parsimonious structures of the different ambiguous regions in all possible combinations. The number of induced skeletons is hence given by the product of the numbers of minimal or parsimonious structures found in each of the independent ambiguous regions.

Exact Search for Parsimonious Structures

If an ambiguous region is reasonably small, it is feasible to search for *all* the minimal or parsimonious structures. Since a preliminary ambiguous region corresponds to a connected component in the condition graph, it can contain nodes X which are not part of a directed cycle in the condition graph. This suggests a pre-processing step which further simplifies an ambiguous region before the minimal or parsimonious structures are actually determined. This can be achieved by repeatedly applying the following scheme until no further simplification is possible.

If a purely exogenous node $X = [x, y]$, i.e. a node from which all the arcs point away in the condition graph, corresponds to a definitely-absent edge $x \sim y$ in the skeleton then

- (i) remove all the rules $CR(Y, \mathbf{E}, \mathbf{C}, \mathbf{I})$, $IR(Y, \mathbf{C}, \mathbf{I})$ where $X \in \mathbf{E}$ or $X \in \mathbf{C}$ from the set of rules,
- (ii) simplify the set of rules by the procedures described in Section 5.2, and
- (iii) build a condition graph according to the new set of rules.

In step (i), this scheme removes those rules which cannot be satisfied in any case. This leads to a simplified set of rules which might then be further reduced in step (ii). Since the removal of rules from \mathcal{R}^A entails the elimination of edges from the condition graph, a previously connected component might split up into different smaller connected components. One can look at each of the smaller connected components as a new ambiguous region, and each of which can then be simplified separately. This renders particularly efficient computations possible. When edges are removed in step (i), it might turn out that all the rules $CR(Y, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and $IR(Y, \mathbf{C}, \mathbf{I})$ corresponding to an edge Y have been removed. This implies that such an edge Y is present in all the minimal skeletons, like a certainly-present edge. It is called a *definitely-present* edge. A definitely-present edge induced in step (i) can imply the presence of some sc-paths and si-paths. Subsequently, this can lead to considerable simplifications of the set of rules in step (ii).

This is illustrated in the connected component in Figure 5.10 (I), where it is assumed that the nodes S and X correspond to definitely-absent edges in the minimal skeletons: step (i) of the above scheme removes those rules which entail the elimination of the arcs $S \rightarrow X$ and $S \rightarrow T$ in the condition graph. This splits the connected component into two smaller ones. Because T is not related to a definitely-absent edge in the skeleton, all the rules $CR(T, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and $IR(T, \mathbf{C}, \mathbf{I})$ can be removed in step (ii). This entails that the edge T has to be definitely present in the skeleton. Moreover, the rules $CR(U, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and $IR(U, \mathbf{C}, \mathbf{I})$ regarding the edge U can now be simplified, namely when $T \in \mathbf{E}$, $T \in \mathbf{C}$ or $T \in \mathbf{I}$. This entails at least the arc $T \rightarrow U$ to be removed from the condition graph (cf. Figure 5.10 (II)). In the second cycle, the above scheme can now be applied to the purely exogenous node X which is assumed to correspond to a definitely-absent edge in this example. At least the arc $X \rightarrow Y$ can thus be eliminated from the condition graph. Finally, two ambiguous regions are obtained which are

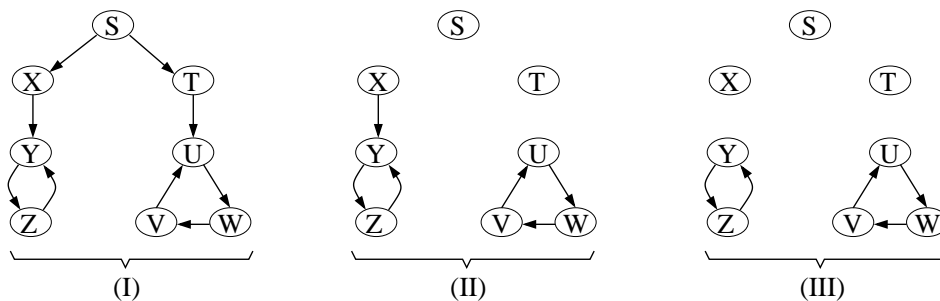


Figure 5.10: A simplistic example of a connected component which can be further simplified. (I) depicts the original component, (II) and (III) sketch the simplifications achieved after one and two cycles, respectively.

considerably smaller than the original one (cf. Figure 5.10 (III)), and the edge T is found to be definitely present.

After this simplification, the algorithm may carry out exhaustive search to find the various minimal or parsimonious structures. In an ambiguous region involving r edges, there are 2^r different structures to be checked for their compliance with the subset of rules \mathcal{R}^A . For each structure, this can, for instance, be checked by using a copy \mathcal{R}_c^A of the subset \mathcal{R}^A as follows: the edges being present in this structure are considered to be definitely present, i.e. all the c-rules and i-rules concerning these edges are removed from \mathcal{R}_c^A . The structure complies with the rules if \mathcal{R}_c^A can be simplified by the procedures described in Section 5.2 such that, for each absent edges Y , a rule can completely be reduced. Among all the structures complying with the subset of rules \mathcal{R}^A , the minimal ones are then given by the ones which do not contain another structure fulfilling the rules.

When the heuristics of parsimony is applied, the complexity of search strategies can be reduced. The concept of parsimony was introduced in Section 4.3.5. All the possible structures can be checked in ascending order $k = 0, \dots, r$ regarding the number of edges being present. Once a structure containing k_p edges is found to comply with the subset of rules \mathcal{R}^A , the structures corresponding to higher orders $k > k_p$ need not be checked any more. The number of different structures which have to be considered at each order k equals $\binom{r}{k}$. Hence, the complexity of finding all the parsimonious structures in an ambiguous region is bound by $\mathcal{O}(r^{k_p})$, where k_p denotes the number of edges in a parsimonious structure fulfilling the rules. In particular, if the number of edges k_p is quite small, this procedure is tractable even for reasonably large values of r . For instance, in our alarm network experiments in Section 5.6.2, we allowed for preliminary ambiguous regions with up to $r \approx 15$ edges, as $k_p \approx 2$ occurred to be very small.

Approximate Search for Parsimonious Structures

In the case of very large ambiguous regions where the above search strategies are infeasible, we found the following approximations very effective. The aim is to simplify the subset of rules of a large ambiguous region such that the latter is likely to split up into, possibly several, smaller regions. Due to their mutual independence, the minimal structures can then be determined separately in each of the smaller regions. This can considerably reduce the complexity of the search. As an alternative to simplifying large ambiguous regions approximately, a less strict variant of the necessary path condition might be applied. This tends to yield smaller ambiguous regions, and exact search might be tractable (cf. the experiments in Section 5.6.2, in particular Figure 5.25).

Let us describe two heuristics for approximately reducing a large connected component in the condition graph. They are both applicable when a node X with a *large* number of parents is present in a connected component. There are two disparate causes for such a large number of parents.

First, the simplified rules $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and $IR(X, \mathbf{E}, \mathbf{I})$ can have condition sets \mathbf{E} , \mathbf{C} and \mathbf{I}

with many different pairs of variables. The absence of the edge X can thus require many other edges, sc-paths and si-paths to be present in the skeleton. On the other hand, all these edges and paths need not be present if the edge X is present. Applying the heuristics of parsimony, one might hence favor the *presence* of the single edge X . In other words, a complex rule is unlikely to be fulfilled in a parsimonious skeleton. This suggests the heuristic approach of removing involved rules from the set \mathfrak{R}^A , while only the simpler rules concerning each edge X remain in the set \mathfrak{R}^A . Hence, if one of the simpler rules is fulfilled then the edge X is absent from the induced skeleton. In case that all rules concerning an edge X have been removed due to their complexity, the edge X becomes definitely-present in the induced skeletons. A consequence of heuristically removing complex rules from \mathfrak{R}^A is that the skeletons eventually induced might contain a few edges additional to the (unknown) parsimonious skeletons complying with the original set of rules. In the algorithm, a threshold value for the maximal number of elements in the sets \mathbf{E} , \mathbf{C} and \mathbf{I} has thus to be chosen carefully.

Note that a simplified rule comprising large condition sets can only origin from a relative score $g(x, z, S) < 0$ with a *large* set S . Such a score is thus ignored when the corresponding rule is removed by this heuristics. Since independence tests of high orders in S are usually quite unreliable, this might be a reasonable approach. However, all rules stemming from a relative score $g(x, z, S) < 0$ with a *large* set S may not be removed by this heuristics, as a complex rule might simplify considerably.

Second, in the condition graph, a large number of parents of a node X can also be caused by a large number of different rules $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ and $IR(X, \mathbf{E}, \mathbf{I})$. Unlike in the former case, each of the condition sets might be quite small. When there is a large number of quite simple rules concerning the same edge, it can be expected that one of which may be fulfilled by chance. This suggests the heuristics of considering such an edge to be definitely *absent* from the skeleton, and of removing the rules concerning the edge X from \mathfrak{R}^A . Moreover, the rules $CR(X, \emptyset, \emptyset, \emptyset)$ and $IR(X, \emptyset, \emptyset)$ are inserted into \mathfrak{R}^A so that the edge X is treated like a definitely-absent one in the subsequent steps, i.e. the inserted rules imply that there is an sc-path as well as an si-path assumed to be present. In the subsequent steps, a considerable simplification of \mathfrak{R}^A can hence be expected. It is obvious that this heuristics can eventually lead to skeletons containing fewer edges than the (unknown) minimal or parsimonious ones obeying the original set of rules \mathfrak{R}^A . As noted in Section 4.3.4, minimal skeletons tend to contain a smaller number of edges than optimal DAGs. For this reason, the qualitative property of the induced skeletons is retained by this heuristics, namely that the found skeletons contain fewer edges than the optimal ones.

Let us also look at these two heuristics from the point of inducing the perfect map. In the first heuristics, if all the simplified rules concerning an edge $X = [x, z]$ are quite complex, then there have to exist *many different* paths between x and z , and each of which has to be blocked in order to achieve a d-separation of the two variables. In contrast, if there is a large number of simple rules regarding an edge $X = [x, z]$, then there have to be *long* paths between x and z . This is because many different variables can be chosen to block the paths such that d-separation occurs. These considerations help choose reasonable threshold values in these heuristics.

Finally, let us mention a third heuristics applying to definitely-absent edges involved in preliminary ambiguous regions: when an i-rule regarding a definitely-absent edge $X = [x, z]$ could be reduced completely, none of the c-rules regarding such an edge need be completely simplified. This suggests a heuristic simplification, namely to remove all c-rules $CR(X, \mathbf{E}, \mathbf{C}, \mathbf{I})$ once the i-rule $IR(X, \emptyset, \emptyset)$ has been induced. Additionally, the completely reduced c-rule $CR(X, \emptyset, \emptyset, \emptyset)$ might be inserted into the set of rules \mathcal{R}^A , like in the second heuristics. This ensures that the resulting skeletons cannot contain more edges than the minimal ones complying with the original set of rules.

A variant of these heuristics is based on *two* necessary si-paths, a strict one and a less rigorous one. Since the ambiguous regions due to a less rigorous necessary si-path condition can be expected to be quite small, an exact search for the minimal structures is usually tractable. The edges found to be present can then be used for searching in a, possibly large, ambiguous region due to the stricter necessary si-path condition: in such an ambiguous region, those edges can be considered as definitely present which were found to be present in a parsimonious skeleton obeying the less strict necessary si-path condition. In the worst case, however, there can be as many different choices for these edges as there are parsimonious skeletons according to the less rigorous condition. Thus, this heuristics becomes infeasible when the less strict si-path condition yields too many parsimonious skeletons. However, if it was tractable in our experiments, it lead to better results than obtained by means of the other approximate search strategies in this section.

5.4 Computing the Scores

This section focuses on computing the relative scores required by the above procedures for constructing the skeletons. Computing the relative scores can be very time-consuming: particularly in the case of discrete variables, in many experiments [23, 24] more than 90% of the overall computation time was necessary for computing the scores, whereas only a small fraction of the time was required by the above scheme for computing the, possibly different, skeletons (see also Section 5.5). Therefore, a great deal of computation time can be saved by computing the scores efficiently as well as by skipping the evaluation of those scores which are not required by the learning algorithm for determining the network structure.

5.4.1 Representation of the Data

Computations can be accelerated by choosing an appropriate representation of the sufficient statistics implied by the data. In the case of a multinomial distribution for discrete variables, a sufficient statistics is given by the cell counts of the different configurations in the contingency table. Concerning domains with Gaussian-distributed variables, the correlation matrix provides a sufficient statistics for calculating the scores. In domains involving a large number of discrete variables, the contingency table with the cell counts of all the joint states is usually many orders of magnitude larger than computers can currently fit into their memories. Consequently,

a straightforward solution is to keep the cases of the data set itself in the memory. Hence, the computation of a score entails to cycle through the cases of the data set and to count the occurrences of the different configurations of the variables in the subset of interest. The complexity of this procedure increases linearly in the number of cases in the data set. Of course, more sophisticated representations can often be chosen. For instance, in the software package CoCo [5], the representation is chosen dependent on the number of variables in a domain. Further techniques are described in [29].

5.4.2 Indirect Evaluation

When a score is calculated from given data, computing the cell counts is the most time-consuming task in general. However, many relative scores need not be computed directly from the data, because all the scores are not independent of each other. This was already mentioned in Section 3.1, in particular see the Equations 3.1.6 and 3.1.9. The former equation can be applied to all relative scoring functions g , while the latter leads to an additional speed-up when the used scoring function is score equivalent. When a score is computed from other scores according to these equations, it is called an *indirect evaluation* of that score. Efficient computation can be achieved by processing the relative scores in *ascending order*. The order of a relative score $g(a, b, S)$ is given by the number of variables in the set $S \subseteq \mathcal{V} \setminus \{a, b\}$, denoted as $|S|$. This means that all the scores with $S = \emptyset$ are computed first, then all the scores involving sets S with one variable, after that with two variables, and so on.

Without applying any of the two Equations 3.1.6 and 3.1.9, in a domain with n variables the number of all relative scores is given by

$$\sum_{i=2}^n i(i-1) \binom{n}{i} = \sum_{i=0}^{n-2} n(n-1) \binom{n-2}{i} = n(n-1)2^{n-2}. \quad (5.4.1)$$

This is because the relative score $g(a, b, S)$ has three arguments, namely the two variables a and b , which cannot be swapped in general, as well as a set $S \subseteq \mathcal{V} \setminus \{a, b\}$. This yields immediately the expression in the center of Equation 5.4.1. The sum of binomial coefficients simplifies in the well-known manner. The first expression in Equation 5.4.1 follows from first choosing a set $\mathcal{U} \subseteq \mathcal{V}$ with at least two elements and subsequently selecting the distinct variables a and b from \mathcal{U} so that S is determined by $\mathcal{U} \setminus \{a, b\}$. The interdependencies of the relative scores according to Equation 3.1.6 reduce the number of independent scores, which have to be computed directly from the data, to

$$\sum_{i=2}^n i \binom{n}{i} = n2^{n-1} - n. \quad (5.4.2)$$

This holds both for score-equivalent as well as for non-score-equivalent scoring-functions. Once a relative score $g(a, b, \mathcal{U} \setminus \{a, b\})$ has been computed, the score $g(a, c, \mathcal{U} \setminus \{a, c\})$ is given by Equation 3.1.6 for all $c \in \mathcal{U} \setminus \{a\}$. A necessary requirement is, of course, that the scores $g(x, y, S)$ of the lower orders $|S| < |\mathcal{U} \setminus \{a, b\}|$ are known. This is the case when the

scores are evaluated in ascending order of S . Hence, regarding each subset $\mathcal{U} \subseteq \mathcal{V}$ with at least two variables, only $|\mathcal{U}|$ scores are independent of each other, and have to be computed directly from the data. This leads to Equation 5.4.2. If the scoring function is score-equivalent, Equation 3.1.9 entails an additional decrease in the number of independent scores: only

$$\sum_{i=2}^n \binom{n}{i} = 2^n - n - 1 \quad (5.4.3)$$

different scores have to be computed directly from the data. This is because Equation 3.1.9 yields the relative scores $g(a, b, \mathcal{U} \setminus \{a, b\})$ for all $a, b \in \mathcal{U} \subseteq \mathcal{V}$. The only requirement is the knowledge of a single score $g(x, y, \mathcal{U} \setminus \{x, y\})$ for some $x, y \in \mathcal{U}$ and of the scores of lower orders.

In domains with discrete variables, the computational effort for the indirect evaluation of a score is essentially negligible compared to the one required for computing the cell counts. The Equations 5.4.1 and 5.4.2 indicate hence that the computation time can be reduced by a factor of about $n/2$ when all the relative scores are evaluated. Regarding a score-equivalent scoring-function, this factor is even larger, namely about $n^2/4$ (cf. Equations 5.4.1 and 5.4.3). Since the overall effort grows exponentially with n , the computation of all the scores is only feasible in domains with a rather small number of variables. Domains with a large number of variables require some (heuristic) simplifications: usually one resorts to the computation of scores of low orders only [142]. Also in this case, the Equations 3.1.6 and 3.1.9 can be exploited when the relative scores are evaluated in ascending order. When all scores $g(a, b, S)$ of a given order $r = |S|$ are computed, the number of scores evaluated directly from the data can be reduced by a factor of $r + 1$ (cf. the terms within the sums in the Equation 5.4.1 and 5.4.2). This holds for any relative scoring function. Moreover, if the scoring function is score-equivalent, this factor amounts to $(r + 2)(r + 1)$ (see the Equations 5.4.1 and 5.4.3). Hence, the computation time can often be reduced by an order of magnitude. The storage of the computed scores of low orders was not a problem in our computer experiments.

Finally, let us briefly consider the number of independent terms involved in a decomposable *absolute* scoring function f . Its decomposability entails that for each variable $v \in \mathcal{V}$ and its parents $\text{pa}(v)$ a term $f_v(v \mid \text{pa}(v))$ has to be computed. The number of these terms, which have to be computed directly from the data, is given by

$$\sum_{i=0}^{n-1} n \binom{n-1}{i} = n2^{n-1} = \sum_{i=1}^n i \binom{n}{i}. \quad (5.4.4)$$

If the scoring function is score equivalent, the number of independent terms is considerably smaller than in Equation 5.4.4. Due to the relation specified in Equation 3.1.11, only a single term $f_a(a \mid \mathcal{U} \setminus \{a\})$, $a \in \mathcal{U}$, has to be computed directly from the data for each subset $\mathcal{U} \subseteq \mathcal{V}$ with a least one variable. Then the terms $f_b(b \mid \mathcal{U} \setminus \{b\})$ can be evaluated by means of Equation 3.1.11 for all $b \in \mathcal{U}$ given the scores of lower orders. Hence, the number of independent terms

is given by

$$\sum_{i=1}^n \binom{n}{i} = 2^n - 1. \quad (5.4.5)$$

The Equations 5.4.4 and 5.4.5 correspond to the Equations 5.4.2 and 5.4.3 concerning relative scoring functions. Apparently, n additional terms have to be computed for the absolute scoring function. The reason is that the absolute scoring function involves an additional independent term concerning each of the n variables in the Bayesian network. This gives rise to an additional absolute score, for instance the score of the empty DAG $f(m_{\text{empty}}) = \sum_{v \in \mathcal{V}} f_v(v | \emptyset)$, which is often a meaningless constant in Equation 3.1.7. Such a term does not occur in a relative scoring function, as the latter is the difference of absolute scoring functions.

5.4.3 Reducing the Number of Computed Scores

Instead of first computing all the scores and then finding the skeletons, one can also alternate between these two procedures. As a benefit, the number of the scores actually computed can be reduced considerably when reasonable heuristics are employed. The idea is that the graph induced at some intermediate stage of the learning process can help decide which scores are worth being computed next. Typically, one chooses such scores which can be expected to have an impact on the induced network structures. Popular heuristics require the relative scores to be computed in ascending order. In the PC algorithm, the intermediate graph is updated each time a conditional independence is found [142]. We propose to update such an intermediate graph only at the end of each round, i.e. after the computation of the scores of a fixed order is completed. As an advantage, the proposed scheme is independent of the sequence in which the algorithm proceeds through the variables. A drawback is, however, that it might entail the computation of a larger number of scores than the PC algorithm.

Typically, such heuristics only compute those scores $g(a, b, S)$ which are related to an edge $a \sim b$ present in the intermediate graph [142]. Moreover, only these scores $g(a, b, S)$ are considered where $S \subseteq \text{ne}(a) \setminus \{b\}$, with $\text{ne}(a)$ denoting the neighbors of a in the intermediate graph. This heuristics is motivated by the fact that S has to be (a subset of) the parents of a (see also Section 3.1.2). This restriction on S results in a considerable reduction in the number of scores which have to be computed, and hence saves a lot of computation time. In [23, 24], another heuristics is used. It is based on the tree construction algorithm by Chow and Liu [32].

Because the summary graph (cf. Section 5.3) serves as the intermediate graph in our algorithm, let us consider the evolution of such an intermediate graph in more detail. Two important issues are pointed out in the following. The first one is illustrated in the example shown in Figure 5.11. Assume that, after the round of zeroth order, the summary graph looks like the left one in Figure 5.11. In the round concerned with the scores of first order, let the following scores be negative: $g(a, b, \{e\})$, $g(a, e, \{x\})$, $g(a, e, \{y\})$, $g(b, e, \{c\})$, and $g(b, e, \{d\})$. This causes the edges $a \sim b$, $a \sim e$ and $b \sim e$ to become ambiguous edges in the preliminary summary graph. In the round of second order, assume the scores $g(a, b, \{c, d\}) < 0$ and $g(a, b, \{x, y\}) < 0$

are found. As a consequence, the edge $a \sim b$ becomes definitely absent in the intermediate summary graph, and so do the edges $a \sim e$ and $b \sim e$. This shows that it is important to keep on computing scores regarding ambiguous edges, although scores favoring the absence of such an edges have already been found in earlier rounds.

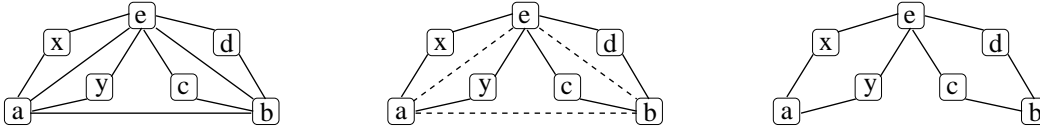


Figure 5.11: In this simplistic example, the summary graphs based on the scores up to zeroth, first and second order are shown from left to right.

The second important issue is that, although an edge is absent from the intermediate graph at the current round, this edge might be present in the intermediate graph at a later round. This is caused by ambiguous edges in the intermediate graph. Let us reconsider the example shown in Figure 4.8. After the round concerned with scores of first order, the ambiguous edges $b \sim d$ and $c \sim d$ are absent from the intermediate graph, as illustrated in Figure 5.12. Consequently, no scores of second order are subsequently computed concerning the edges $b \sim d$ and $c \sim d$. However, these edges are present in the intermediate graph after the round of second order is completed. Their presence indicates, however, that also scores of second (and higher) order have to be taken into account when determining the skeletons. The learning algorithm has thus to make up for scores whose computation was possibly skipped due to the temporary absence of some edges from the intermediate graph.



Figure 5.12: This example rests on the one in Figure 4.8. The left summary graph is obtained by taking into account the scores up to first order only, whereas the right one accounts for all orders.

A drawback of this heuristics is that the skeletons induced from a reduced number of scores can contain additional edges, compared to the graphs learned from the full record of scores. Another consequence is that the number of induced skeletons might be decreased. The reason is that the heuristics might prevent from computing some scores $g(a, b, S) < 0$ which possibly imply the absence of an edge $a \sim b$. This situation arises when the set S has to contain some variables which are not among the neighbors of a or b , because the required edges have already been removed in a previous round due to other scores. Chances for this to happen can be reduced, for instance, by using a "window" for the threshold value γ to which the scores $g(a, b, S)$ are compared: a quite large value of γ tends to yield quite many ambiguous edges in the summary graph, while a rather small value entails an increased number of definitely-present edges. Combining the edges from both summary graphs reduces chances that relevant edges are absent from the new intermediate graph.

In our computer experiments, it occurred very rarely that an additional edge was mistakenly present in the induced skeletons. This is usually caused by another effect: a variable a often has more neighbors in the (undirected) intermediate graph than it would have parents in the corresponding DAG. Hence, the intermediate graph entails the computation of scores of higher orders than one would take into account when dealing with DAGs. Scores involving high orders can, however, favor the absence of an edge mainly because the penalty for model complexity dominates. In order to prevent the algorithm from considering scores of unreasonably high orders, the following heuristics appeared useful in our experiments: a score $g(a, b, S)$ is only computed if the number of variables adjacent to a by a definitely-present edge, but not by an ambiguous one, is at least as large as the order of S in the current round. The variables allowed to be in S can then be adjacent to a by a definitely-present edges or an ambiguous one.

The heuristics which only consider such scores $g(a, b, S)$ where S is among the neighbors of a can be understood as a relaxed version of the necessary path condition (cf. Section 4.3.1), as the latter also requires the variables in S to be adjacent to a . Hence, even without applying the necessary path condition, this heuristics can be expected to lead to some improvements regarding the induced skeleton, namely a reduction in the number of edges mistakenly absent. This is supported by the experiments concerned with the PC and SGS algorithms [142].

5.4.4 Parallel Computing

As the scores can be computed independently of each other in each round of a given order, this task is appropriate for parallel computing. We implemented a simple master-slave scheme, which was very efficient in our computer experiments. The implementation was based on the software package called *Parallel Virtual Machine (PVM)* [121]. The slave processes were distributed over the various computers in the local cluster. Usually, we used about ten computers. The data was locally available in each slave process, which ensured efficient computations of the scores. The master process was in charge of distributing the tasks from a shared queue containing all the scores which had to be computed. The common queue served as a simple device for dynamic load balancing, which was important as the local cluster was quite heterogeneous. The results of the computations were returned to the master process upon completion. Having computed all the scores of a fixed order, the master process had to update the intermediate graph guiding the evaluation of the scores in the next round. As already mentioned, this task is considerably less time consuming than computing the scores. For this reason, it was not a bottle-neck in the master-slave scheme. Also, the time spent on message passing was rather negligible compared to the time spent on computing the scores.

5.4.5 Positive Threshold Values

Relative scoring functions favor the presence of an edge if the score is positive, whereas its absence is favored by a negative score. Consequently, the threshold value $\gamma = 0$ was used in the discussions so far. As already mentioned in Section 3.1.4, however, one may employ

$g(a, b, S)$	$G(a, b, S)$	Evidence for the presence of edge $a \leftarrow b$ when $S = \text{pa}(a) \setminus \{b\}$
$0 \dots 1$	$1 \dots 3$	not worth more than a bare mention
$1 \dots 3$	$3 \dots 20$	positive
$3 \dots 5$	$20 \dots 150$	strong
> 5	> 150	very strong

Table 5.5: Some positive threshold values and their interpretation (adapted from [93], originally applying to the Bayes factor).

a threshold value $\gamma \neq 0$ in practice. In particular, a positive value is often used in order to include an edge into the graph only if it leads to a notable improvement in the absolute score.

A larger threshold value typically entails a decreased number of edges being present in the induced skeletons. This consequence is in some sense similar to the fact that the minimal skeletons tend to contain a smaller number of edges than the optimal graphs. Hence, a positive threshold value usually does not affect the latter property of the induced minimal or parsimonious skeletons, while it leads to an improved robustness of the algorithm. A threshold value can be interpreted in terms of evidence, as depicted in Table 5.5. The effect of different threshold values was examined in the experiments in Section 5.6.2.

Note that the ambiguous edges are qualitatively different from the edges additionally included into the graph when the threshold value is decreased. This is because the necessary path condition yields ambiguous edges independently of the exact values of the corresponding scores (once they are smaller than the threshold value).

5.5 Efficiency of the Algorithm

As structural learning in Bayesian networks is an NP-hard problem [14, 27, 84], heuristic approaches are necessary. However, they can be quite time-consuming, as well. The constraint-based approach has proven to be very efficient in many experiments, e.g. [23, 24, 142]. This holds in particular when the induced graphs are rather sparse, as this allows to determine the graph from a reasonably small number of scores.

The computation of each score depends linearly on the number N of cases if the record of cases is stored in memory, as typically done in domains with a large number of discrete variables. The heuristics for reducing the number of computed scores appeared to be very important for decreasing the computational effort. As already mentioned in [142], the overall number n of nodes in the graph as well as the maximal number of neighbors n_{dp} of a node determine the number of scores which have to be computed from the data. When a score-equivalent relative

scoring-function is used, this number is bound by

$$\sum_{i=0}^{n_{dp}} \binom{n}{i+2}, \quad (5.5.1)$$

as derived in Equation 5.4.3. It is assumed that, in the worst case, for each pair of variables all the scores up to the order n_{dp} have to be evaluated. The complexity of this scheme unfortunately increases exponentially with n_{dp} . This explains why it is particularly efficient when the induced graphs are sparse, i.e. each node can be expected to have a few neighbors only. As discussed in Section 1.1, a Bayesian network cannot be considered an appropriate model for domains which require the induced DAG to be dense.

In the experiments described in the remainder of this chapter, computing the scores required more than 90% of the overall computation time. The extension of the constraint-based approach by means of the necessary path condition, as presented in this thesis, hence entails a rather negligible increase in computation time. Once the scores have been calculated, the parsimonious skeletons could be induced by means of the set of rules within a few seconds. Of course, this task depends crucially on the size of the induced ambiguous regions. For that reason, the computation time required for simplifying the rules and for inducing the parsimonious structures varied between 1 and 10 seconds. This duration occurred to be quite independent of the strictness of the si-path employed by the necessary path condition. Of course, also this part can be very involved in the worst case. An exact analysis of the complexity of the various procedures is, however, very tedious. Fortunately, the worst case usually does not occur in practice. In order to keep the computation time at some acceptable duration, involved cases have to be tackled by approximations, described in Section 5.3. This ensures that the computation time is bound in practical applications. Nevertheless let us briefly consider some procedures in more detail.

The set of rules is simplified by repeatedly cycling through all the rules (cf. Section 5.2). Let the number of rules, and hence the maximal length of a cycle, be denoted by l . Let r be the maximal length of a rule, i.e. the number of pairs of variables contained in the condition sets. In each cycle, only a definitely-present edge or a definitely-absent one can entail a simplification of the rules. In order to render a simplification in the subsequent cycle possible, at least one definitely-present or definitely-absent edge has thus to be found in each cycle. Since a skeleton with n variables contains at most $k = \binom{n}{2}$ edges, at most $2k$ cycles have thus to be carried. The factor of 2 stems from the assumption that, in the worst case, first the i-rule and then the c-rule regarding each edge reduces completely. The overall complexity of this scheme is thus of the order of $\mathcal{O}(lkr)$ in the worst case.

The complexity of finding all the minimal or parsimonious structures in the largest ambiguous region determines the complexity of inducing the different skeletons. This is because the presence of edges in different ambiguous regions is independent of each other. The complexity of exhaustive search for the minimal structures grows obviously exponentially with the number of edges contained in a preliminary ambiguous region (cf. Section 5.3.2). The parsimonious structures in an ambiguous region can be determined with a reduced complexity, which grows

exponentially with the number of edges actually found to be present in the parsimonious structures. Very large ambiguous regions can be tackled by the approximations described in Section 5.3 in order to keep the necessary computation time at some acceptable duration.

5.6 Experiments

In this section, various experiments are carried out with the presented learning algorithm aimed at finding the parsimonious skeletons. The benefits of employing the necessary path condition are examined given various sample sizes. Moreover, the different si-paths used in the necessary path condition are compared to each other as well as to other approaches commonly used. Since the induced graphs are skeletons rather than DAGs, we use artificial data sampled from a Bayesian network, because this allows us to compare the induced structures with the original one in a qualitative way. DAGs rather than skeletons are determined in the experiments in Section 6.3, rendering a quantitative comparison possible. Also real-world data can hence be used there.

5.6.1 A Simplistic Example

Our first experiment is concerned with a very small Bayesian network comprising only three variables. The original network structure is depicted in Figure 5.13. This DAG might already be familiar to the reader, as it appeared in several examples in the previous chapter (cf. the Figures 4.4 and 4.5 as well as the Tables 5.1 and 5.2). In order to specify a quantitative model, let the continuous random variables a , b and c depend on each other like $b = a$, $a = m_{ab}b + \varepsilon_a$ and $c = m_{cb}b + \varepsilon_c$ with the parameters $m_{ab} = 3.0$ and $m_{cb} = 0.3$, where ε_i ($i \in \{a, b, c\}$) are Gaussian distributed random variables with unit variance and zero mean. Hence, the variables a and b are strongly correlated, whereas the correlation of c with a or b is relatively small.

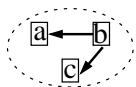


Figure 5.13: The original DAG.

From the probability distribution described by this Bayesian network, data sets of various sizes are sampled, ranging from 10 up to 10,000 cases. These data sets are then given to the learning algorithm. In this example, the Bayesian Information Criterion² (BIC) serves as a score-equivalent scoring-function g , and the threshold value is chosen to be $\gamma = 0$. The learning

²Since this domain comprises continuous variables with a Gaussian distribution, the BIC given in Equation 3.1.19 is based on $d(a \perp\!\!\!\perp b \mid S) = -N/2 \cdot \log(1 - \hat{\varrho}_{ab,S}^2)$, where $a, b \in \mathcal{V}$ and $S \subseteq \mathcal{V} \setminus \{a, b\}$. Note that a factor of 2 is again omitted, compared to the standard notation. The partial correlation can, for instance, be evaluated recursively according to $\hat{\varrho}_{ab,S \cup \{c\}} = (\hat{\varrho}_{ab,S} - \hat{\varrho}_{ac,S} \hat{\varrho}_{bc,S}) / (\sqrt{1 - \hat{\varrho}_{ac,S}^2} \sqrt{1 - \hat{\varrho}_{bc,S}^2})$, where $\hat{\varrho}_{ab,\emptyset} := \hat{\varrho}_{ab}$ is the estimated correlation between a and b . The degrees of freedom equal $d = 1$.

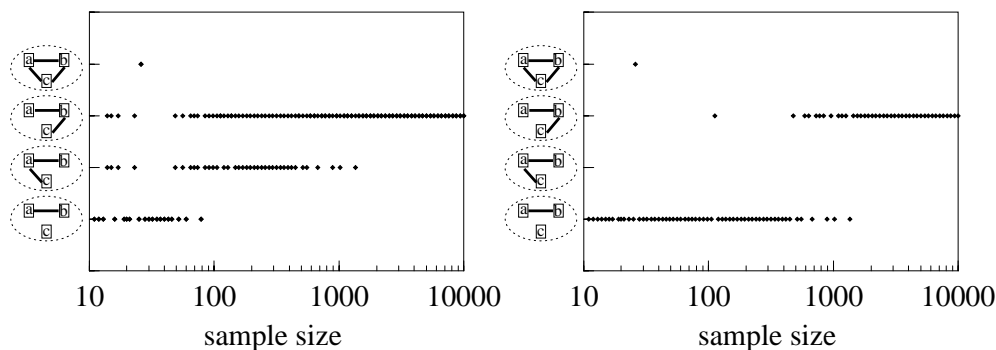


Figure 5.14: The induced skeletons depend on the sample size. The results of the algorithm employing the necessary si-1-path condition are depicted in the left diagram, whereas the right one shows the results of a typical constraint-based approach not using the necessary path condition, e.g. the PC or SGS algorithm [142]. The BIC is used as the scoring function.

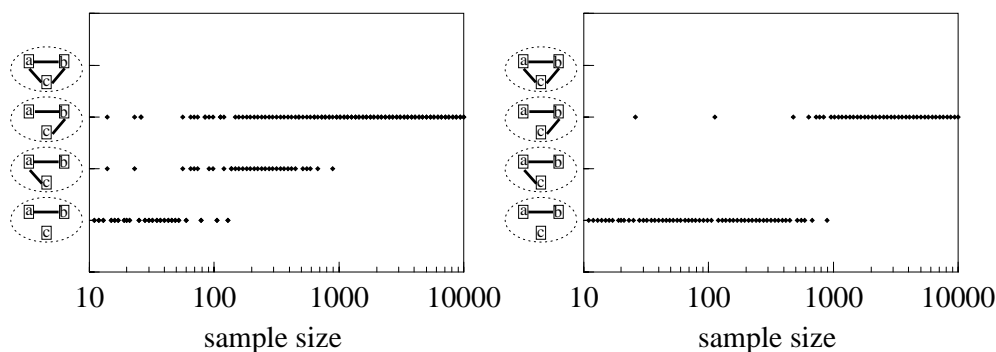


Figure 5.15: This experiment is identical with the one in Figure 5.14, except that the χ^2 -test serves as sort of a relative scoring function.

algorithm is based on the necessary path condition employing the si-1-path (cf. Definition 4.11 and Proposition 4.12).

The skeletons learned from the various samples are shown in Figure 5.14: when given rather large sample sizes, the skeleton of the original DAG can be recovered. As the sample size decreases, model uncertainty arises. By means of the necessary path condition, two different skeletons are induced from most of the samples containing between about 60 and 600 cases. This was discussed in the example shown in Figure 4.5 and in Table 5.2. In this regime, *some* dependence of c on the other two variables is implied by the data. However, the algorithm cannot distinguish between the graphs containing either the edge $a \sim c$ or $b \sim c$. If the size of the data set is smaller than 60 cases, the skeleton with the single edge $a \sim b$ is derived from the data, representing the strong correlation between these two variables.


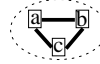

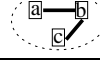



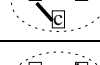

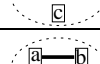

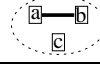


	negative scores regarding		skeletons induced by means of	
	edge $a \sim c$	edge $b \sim c$	nec. path cond. (si-1-path)	no cond.
(1)	–	–		
(2)	$g(a, c, \{b\})$	–		
(3)	$g(a, c, \emptyset)$ $g(a, c, \{b\})$	– $g(b, c, \{a\})$		
(4)	–	$g(b, c, \{a\})$		
(5)	– $g(a, c, \{b\})$	$g(b, c, \emptyset)$ $g(b, c, \{a\})$		
(6)	$g(a, c, \{b\})$	$g(b, c, \{a\})$		
(7)	$g(a, c, \emptyset)$ $g(a, c, \{b\})$	$g(b, c, \emptyset)$ $g(b, c, \{a\})$		

Table 5.6: This table shows the correspondence between the scores and the induced skeletons, with and without employing the necessary path condition. All the scores not denoted in this table are understood to be positive.

This experiment was also carried out with an algorithm not employing a necessary path condition (see Figure 5.14), e.g. the SGS or PC algorithm [142] using the *BIC*. Such an algorithm always induces a single skeleton and hence does not account for model uncertainty. This is because the data is assumed to imply a faithful probability distribution so that a (unique) perfect map exists. However, this assumption does not necessarily hold for data sets of *finite* size. This is apparent for most samples containing between 60 and 600 cases in this example (see Figure 5.14), as the skeleton with the single edge $a \sim b$ is induced from most of these samples. This means that the variable c appears to be marginally independent of both the other variables, although this does not hold according to the induced scores. In contrast, the necessary path condition yields those skeletons which represent the dependence of c on one of the other variables (cf. the found uncertainty regarding the edges $a \sim c$ and $b \sim c$).

Table 5.6 depicts the correspondence between the induced skeletons and the scores calculated from the data sets. The edge $a \sim b$ is always present, as implied by the scores $g(a, b, \emptyset) > 0$ and $g(a, b, \{c\}) > 0$. In (3), (5) and (6), the necessary path condition yields skeletons with more edges than there are found by established constraint-based approaches, not employing such a condition (cf. also the Figures 4.4 and 4.5 as well as the Tables 5.1 and 5.2). Model uncertainty regarding the presence of edges – as discovered by the necessary path condition – occurs in case (6).

We note that the parsimonious skeletons induced with the help of the necessary path condition and the si-1-path (cf. Section 4.3.1) coincide with the minimal skeletons as well as with the

optimal ones in this simplistic experiment. This can be seen, for instance, in the approach shown in the Figures 4.4 and 4.5 in Section 4.3.1. In (3) in Table 5.6, for example, the edge $a \sim c$ is certainly absent due to the scores $g(a, c, \emptyset) < 0$ and $g(a, c, \{b\}) < 0$, whereas the score $g(b, c, \emptyset) > 0$ favors the presence of the edge $b \sim c$.

In a variant of the above experiment, let us also consider the SGS and PC algorithms [142] employing the χ^2 -test³ rather than the *BIC* as sort of a relative scoring function (cf. also Section 3.1.4). A significance level of $\alpha = 1\%$ is used in the experiments shown in Figure 5.15. Obviously, the results are very similar to the ones in Figure 5.14, as one might have expected. The χ^2 -test and the *BIC* are closely related in this experiment, as they yield identical scores at the sample size $n_o \approx 735$, where their penalties regarding model complexity take on equal values, namely $\log(n_o) = \chi^2_{1-\alpha=0.99}(d_f = 1) \approx 6.6$ (d_f denotes the degrees of freedom, see also the first footnote in this section). This is because the penalty term in the *BIC* depends on the sample size n , while the one in the χ^2 -test does not. These two relative scoring functions are identical concerning model fit, as both assess it in terms of maximum likelihood. At small samples ($n < n_o$), the χ^2 -test entails a larger penalty regarding model complexity than the *BIC*, and the other way round for $n > n_o$. This explains why the range of sample sizes for which more than one skeleton is induced extends to both slightly larger and slightly smaller values in Figure 5.14 than it does in Figure 5.15.

5.6.2 The Alarm Network

The following experiments are carried out with various data sets sampled from the probability distribution described by the alarm network [8], the most popular benchmark for assessing structural learning algorithms. The alarm network is described in more detail in Appendix B. Its structure is shown in Figure 5.16. The alarm network is made up of 37 variables, which causes the search space to be extremely large, namely it contains more than $2^{37 \cdot 36/2} > 10^{200}$ DAGs (cf. Equation 3.3.2).

Preliminary versus Parsimonious Summary Graph

In many publications, data sets with 10,000 cases were used in alarm-network experiments, and the induced DAGs were very close to the original structure, see e.g. [23,24,33,34,79,81]. In the following, we applied our learning algorithm to a data set with 3,000 cases. In particular, the variant employing the si-4-path in the necessary path condition was used (cf. Definition 4.16), the least strict alternative among the four si-paths described in Section 4.3. Moreover, the posterior probability with conjugate priors served as the relative scoring function (cf. Section 3.1.4). An equivalent sample size of $N' = 1$ and a threshold value of $\gamma = 3$ was chosen (cf. Table 5.5). Also, we committed ourselves to a uniform prior for the DAGs, i.e. $p(m) = \text{const.}$

³In domains with continuous variables, the PC algorithm originally tests on vanishing partial correlations in a slightly different way, while it applies the χ^2 -test to discrete variables only. Here, we only like to use *some* frequentist *test* which can easily be compared to the *BIC*.

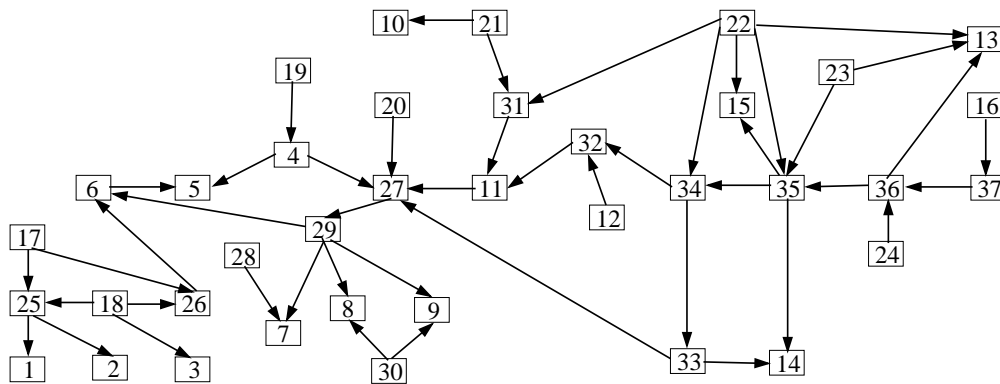


Figure 5.16: This graph displays the original alarm-network structure [8]. A key to the various variables is given in Appendix B.

The preliminary summary graph induced from this sample is shown in Figure 5.17. Apparently, a large number of edges is common to all the parsimonious skeletons, i.e. 34 edges are certainly present. The parsimonious skeletons can only differ from each other regarding the presence of the ambiguous edges involved in the 7 ambiguous regions. Because the ambiguous regions (F)⁴ and (G) render the preliminary summary graph rather complex, the ambiguous edges are also given in Table 5.7.

The different parsimonious structures in each of the ambiguous regions are given in Table 5.8. Apparently, in each of the ambiguous regions (A) through (D), either one of the involved ambiguous edges is present in a parsimonious skeleton. This is very similar to the example discussed in Figure 4.5 and Table 5.2. The ambiguous region (E) comprises basically two different structures, namely the one containing the *single* edge $x_{22} \sim x_{34}$ and the other one comprising *both* the edges $x_{14} \sim x_{22}$ and $x_{22} \sim x_{32}$. This is essentially identical with the example shown in Figure 4.8. Only the first alternative is parsimonious. The second alternative is thus disregarded in a *parsimonious* ambiguous region, as discussed in Section 4.3.5. In the ambiguous regions (F) and (G), it occurs only one parsimonious structure, similarly to (E). Each of which contains two edges. Consequently, there is only one parsimonious structure in each of the ambiguous regions (E), (F) and (G). The edges which are not contained in one of the parsimonious structures are definitely absent, and they can be excluded from the ambiguous regions (E), (F) and (G). This leads to the *parsimonious* ambiguous regions and to the *parsimonious* summary graph sketched in Figure 5.18. The latter is considerably simpler than the preliminary one (cf. the Figures 5.17 and 5.18). Hence, the structure of the parsimonious summary graph might be much easier to interpret than the preliminary one.

⁴The ambiguous edge $x_6 \sim x_9$ (and the absence of the edge $x_6 \sim x_8$) in the ambiguous region (F) entails some asymmetry regarding the variables x_8 and x_9 , which is unexpected. The explanation is that the single score $g(x_8, x_{29}, \{x_{27}, x_9, x_7\}) < \gamma$ regarding the edge $x_8 \sim x_{29}$ was induced, whereas both the scores $g(x_9, x_{29}, \{x_{27}, x_8, x_7\}) < \gamma$ and $g(\mathbf{x}_9, \mathbf{x}_{29}, \{\mathbf{x}_6, x_7, x_8\}) < \gamma$ concerning the edge $x_9 \sim x_{29}$ were derived. As can be seen (cf. the variables denoted in bold face), the latter score causes the edge $x_8 \sim x_9$ to be ambiguous, because the only induced scores regarding the edges $x_8 \sim x_8$ and $x_6 \sim x_9$ were $g(x_6, x_8, \{x_{29}\}) < \gamma$ and $g(\mathbf{x}_6, \mathbf{x}_9, \{\mathbf{x}_{29}\}) < \gamma$, respectively.

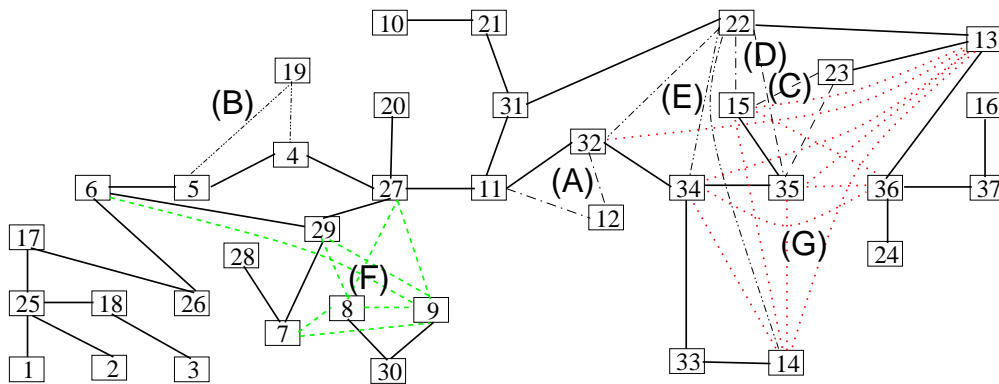


Figure 5.17: This preliminary summary graph depicts the 34 certainly-present edges as well as the 7 preliminary ambiguous regions found. At this stage, the different parsimonious structures are not yet known.

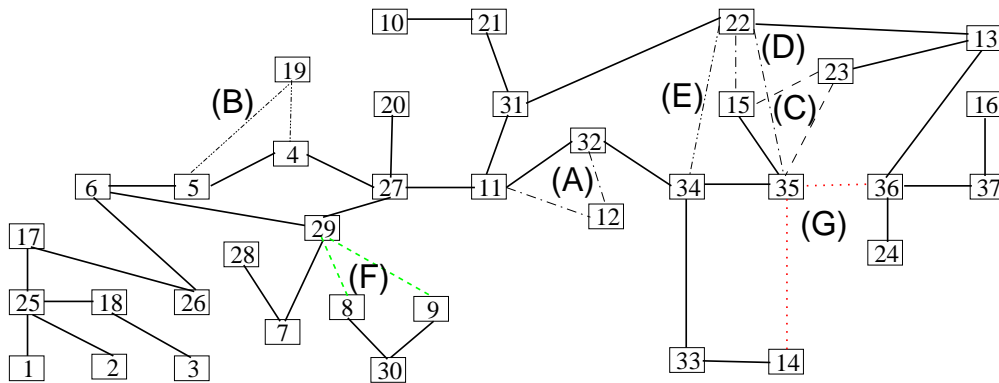


Figure 5.18: This parsimonious summary graph displays all the 16 parsimonious skeletons induced from a sample of 3,000 cases. It is considerably less involved than the preliminary summary graph in Figure 5.17.

Table 5.8 shows that there are two parsimonious structures in each of the ambiguous regions (A) through (D), and only one alternative in (E), (F) and (G). Since different ambiguous regions are independent of each other (cf. Section 5.3), the number of parsimonious skeletons is given by $2^4 \cdot 1^3 = 16$. Each parsimonious skeleton contains 9 ambiguous edges besides the 34 definitely-present ones. This amounts to a total of 43 edges in each of the parsimonious skeletons. In the skeletons which have the most edges in common with the original alarm-network structure (cf. Figure 5.16) only 3 edges are missing, namely the edges $x_{18} \sim x_{26}$, $x_{27} \sim x_{33}$ and either one of the ambiguous edges $x_{15} \sim x_{22}$ or $x_{22} \sim x_{35}$, whereas an edge which is absent in the original DAG is not induced to be present. This supports that the edges found to be present by the constraint-based approach are so with some certainty (cf. Section 4.3.4).

<i>preliminary</i> ambiguous regions	number of involved edges	ambiguous edges
(A)	2	$x_{11} \sim x_{12}, x_{12} \sim x_{32}$
(B)	2	$x_4 \sim x_{19}, x_5 \sim x_{19}$
(C)	2	$x_{15} \sim x_{23}, x_{23} \sim x_{35}$
(D)	2	$x_{15} \sim x_{22}, x_{22} \sim x_{35}$
(E)	3	$x_{14} \sim x_{22}, x_{22} \sim x_{32}, x_{22} \sim x_{34}$
(F)	8	$x_6 \sim x_9, x_7 \sim x_8, x_7 \sim x_9, x_8 \sim x_9, x_8 \sim x_{27},$ $x_9 \sim x_{27}, x_8 \sim x_{29}, x_9 \sim x_{29}$
(G)	11	$x_{13} \sim x_{14}, x_{13} \sim x_{15}, x_{13} \sim x_{32}, x_{13} \sim x_{34}, x_{13} \sim x_{35},$ $x_{14} \sim x_{15}, x_{14} \sim x_{34}, x_{14} \sim x_{35}, x_{15} \sim x_{36}, x_{34} \sim x_{36},$ $x_{35} \sim x_{36}$

Table 5.7: The induced preliminary ambiguous regions and their ambiguous edges.

<i>parsimonious</i> ambiguous regions	parsimonious structures	number of different structures	overall number of involved edges
(A)	<ul style="list-style-type: none"> • $x_{11} \sim x_{12}$ • $x_{12} \sim x_{32}$ 	2	2
(B)	<ul style="list-style-type: none"> • $x_4 \sim x_{19}$ • $x_5 \sim x_{19}$ 	2	2
(C)	<ul style="list-style-type: none"> • $x_{15} \sim x_{23}$ • $x_{23} \sim x_{35}$ 	2	2
(D)	<ul style="list-style-type: none"> • $x_{15} \sim x_{22}$ • $x_{22} \sim x_{35}$ 	2	2
(E)	<ul style="list-style-type: none"> • $x_{22} \sim x_{34}$ 	1	1
(F)	<ul style="list-style-type: none"> • $x_8 \sim x_{29}, x_9 \sim x_{29}$ 	1	2
(G)	<ul style="list-style-type: none"> • $x_{14} \sim x_{35}, x_{35} \sim x_{36}$ 	1	2

Table 5.8: The different parsimonious structures in each of the ambiguous regions.

Various Sample Sizes

The above experiment was also carried out when given different sample sizes, ranging from 1,000 up to 50,000 cases, where five different samples of each size were rendered. The induced parsimonious skeletons were compared to the original alarm network structure. For brevity, let us call an edge to be mistakenly present (absent) in the remainder of this chapter if it is present (absent) in an induced skeleton, although it is absent (present) in the *original* alarm-network structure. Note that this term applies only with respect to the *original* graph and not to the optimum, with respect to the employed scoring function. In the experiments described in Section 6.3.1, for instance, although the edge $x_{15} \sim x_{22}$ is mistakenly absent (i.e. with respect to the original DAG), its absence actually leads to a DAG with a score higher than the one of the original DAG. Its absence might thus be considered "correct" with respect to this optimal graph.

Figure 5.19 shows that, over a large range of sample sizes, the overall number of induced edges is very close to the number of 46 edges present in the original DAG. In contrast, the number of definitely-present edges is much smaller and drops more quickly when the sample size decreases. Moreover, the number of definitely-present edges appears to approach the number of 46 edges quite slowly as the sample size increases, although it can be expected to reach the correct number in the asymptotic limit [142]. We note that the definitely-present edges correspond to the ones induced by the PC algorithm when the posterior probability is applied as the relative scoring function instead of the original χ^2 -tests. Obviously, the ambiguous edges play an important role when recovering the original structure. The Figures 5.19 and 5.20 show that, as the sample size diminishes, the number of ambiguous edges rises. In contrast, the number of definitely-present edges decays, indicating a reduced confidence in the presence of several edges.

Since only a *necessary* (rather than a sufficient) path condition can be applied, the induced skeletons tend to contain a smaller number of edges than there are present in the optimal skeleton (cf. Section 4.3.4). In particular at relatively small sample sizes, this difference can be considerable, as revealed by the rapid increase in the number of mistakenly-absent edges (cf. Figure 5.20). Since different parsimonious skeletons contain a different number of mistakenly-absent and mistakenly-present edges, we refer to the one having the most edges in common with the original DAG. Figure 5.20 shows that the number of mistakenly-present edges is quite small over the entire range of sample sizes, as expected from the considerations in the Sections 4.3.4 and 5.4.3.

The Figures 5.20 and 5.21 show that the number of ambiguous edges which are simultaneously present in an induced skeleton is approximately the same as the number of ambiguous regions. Hence, the average number of ambiguous edges which are simultaneously present in a parsimonious ambiguous region is about 1 for all sample sizes. This renders an efficient search for the different parsimonious structures in these experiments possible, as described in Section 5.3.2.

The number of parsimonious skeletons as well as the number of ambiguous regions generally

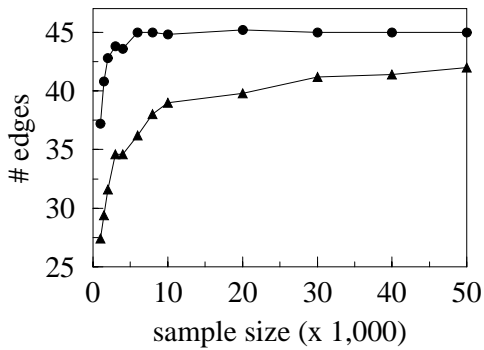


Figure 5.19: In the induced parsimonious skeletons, the number of definitely-present edges (●) is considerably smaller than the overall number of edges (●), i.e. the edges which are ambiguous or definitely present (mean of 5 samples of each size).

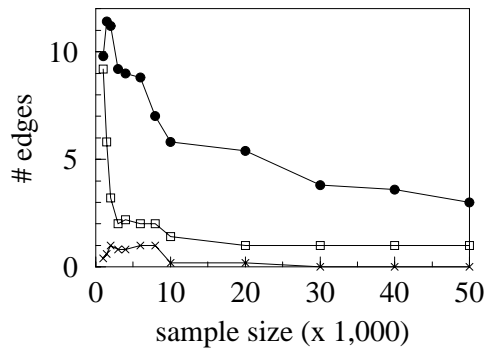


Figure 5.20: The number of ambiguous edges (●) rises as the sample size decreases. Some edges are mistakenly absent (□), and a few are mistakenly present (×) in the parsimonious skeletons closest to the original one (mean of 5 samples of each size).

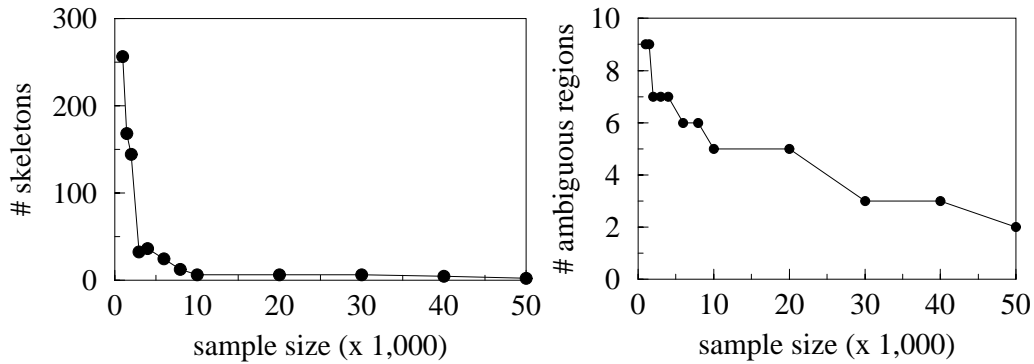


Figure 5.21: The number of parsimonious skeletons (left) and the number of ambiguous regions (right) induced from various sample sizes (median of 5 samples of each size).

increase when the sample size drops (cf. Figure 5.21). This reflects an increase in model uncertainty. Unlike the number of ambiguous regions, the number of skeletons rises quite dramatically when given samples with fewer than 3,000 cases. This is because the number of induced skeletons is the *product* of the number of parsimonious structures in each of the ambiguous regions (cf. Section 5.3). Roughly speaking, one can hence expect an exponential dependence of the number of skeletons on the number of ambiguous regions.

Different SI-Paths

In the following, let us consider the four variants of si-paths introduced in Section 4.3. They entail different variants of the necessary si-path condition. In short, the si-paths can be described as follows: given a score $g(a, b, \mathcal{S}) < \gamma$ regarding the edge $a \sim b$, $\mathcal{S} \subseteq \mathcal{V} \setminus \{a, b\}$, where γ denotes the threshold value,

- the si-4-path leads to the least strict necessary path condition, requiring si-paths between a and each $s \in \mathcal{S}$ as well as between b and each $s \in \mathcal{S}$,
- the si-3-path is more rigorous than the si-4-path, namely it calls for the presence of edges rather than paths between a and each parent candidate $s \in \mathcal{S}$,
- the si-1-path involves sc-1-paths, and is thus even stricter,
- the si-2-path leads to the strictest necessary path condition, requiring si-paths among an increased number of variables.

This leads to four variants of the learning algorithm based on the four variants of the necessary path condition. Each variant is applied to the data sets sampled from the alarm network in the same fashion as described above. The edges present in the induced skeletons are summarized in the Figures 5.22 and 5.23. Apparently, the skeletons induced without applying a necessary path condition contain a considerably smaller number of edges than the skeletons derived by any one of the four variants using the necessary path condition. The differences among the four variants occur to be rather small in this experiment. In detail, the skeletons induced by means of the si-4-path (least strict) seem to be most different from the others: the si-4-path yields a slightly smaller number of (ambiguous) edges than the other three si-paths when given rather small sample sizes in Figure 5.22. Moreover, Figure 5.23 shows that the si-4-path yields the smallest number of edges mistakenly present. The difference between the definition of the si-4-path and the other three si-paths is that the latter require also the presence of *edges* rather than the presence of *paths* only. This enables the three stricter si-paths to find a few additional edges to be present. However, chances are relatively large that some of these additional edges are mistakenly present (cf. Figure 5.23). The two most rigorous si-paths perform very similarly regarding both the number of edges correctly present as well as the ones mistakenly present. This might be expected, since also their definitions are very alike (cf. the Definitions 4.11 and 4.14).

The si-4-path (least strict) has the following desirable property regarding the computational effort (cf. the Figures 5.24 and 5.25): as the sample size decreases, the *number* of preliminary ambiguous regions rises, whereas the number of edges contained in the largest preliminary ambiguous region stays approximately put. Since the complexity of finding the parsimonious structures in the ambiguous regions depends critically on the *size* of the preliminary ambiguous regions rather than on their number, the si-4-path allows an exact search for almost all the data sets in this experiment, and only a few require an approximate search (see also Section 5.3.2). In contrast, concerning the three strictest si-paths, the *size* of the largest preliminary ambiguous

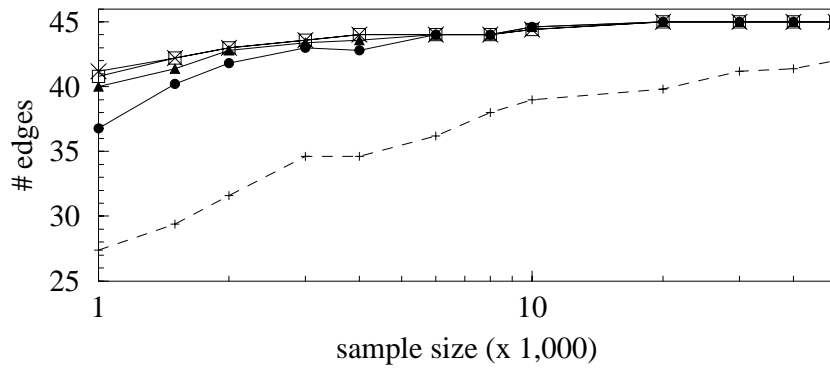


Figure 5.22: The number of the edges correctly present due to the si-4-path (●), si-3-path (triangles), si-2-path (□), and si-1-path (×) used in the necessary path condition. In comparison to that, the dashed line sketches the number of edges found without applying a necessary path condition (mean of 5 samples of each size).

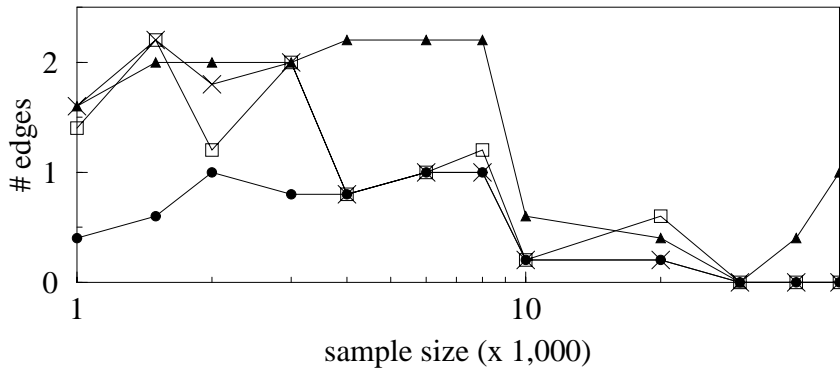


Figure 5.23: The least strict si-4-path (●) yields the smallest number of edges mistakenly present (mean of 5 samples of each size). Regarding the si-3-path (triangles), si-2-path (□), and si-1-path (×), the strictness of the employed si-path does not seem to have a very decisive impact on the number of edges erroneously present.

region is quite large and increases as the sample size drops, while the number of preliminary ambiguous regions tends to grow only slightly. Hence, an approximate search is essential for efficient computations in these cases. Since such an approximate search typically does not find all the parsimonious structures, it is crucial to determine the "important" ones which eventually lead to (close to) optimal graphs. Moreover, the number of induced parsimonious skeletons does not provide much insight, since it is strongly affected by the approximations. Only in the case of the least strict si-4-path, where an approximate search usually is not necessary, this number might provide some insight (cf. Figure 5.21).

Apart from the necessity of applying approximate search strategies in large preliminary ambiguous regions, strict si-paths can often lead to summary graphs which are difficult to interpret. For this reason, it might be favorable to display the summary graph induced by means of

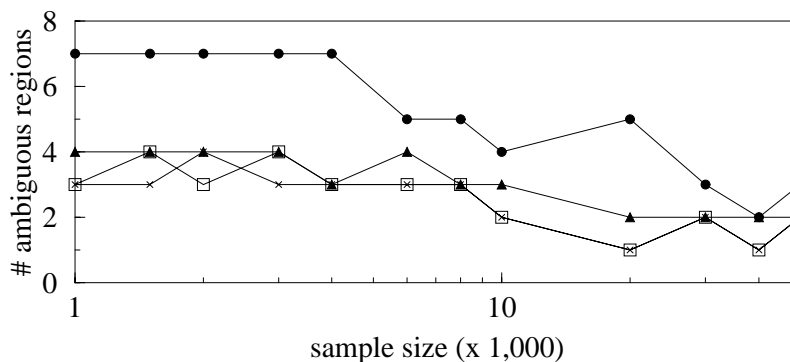


Figure 5.24: The number of preliminary ambiguous regions in the graphs determined on the basis of the si-4-path (●), si-3-path (triangles), si-2-path (□), and si-1-path (×) (median of 5 samples of each size).

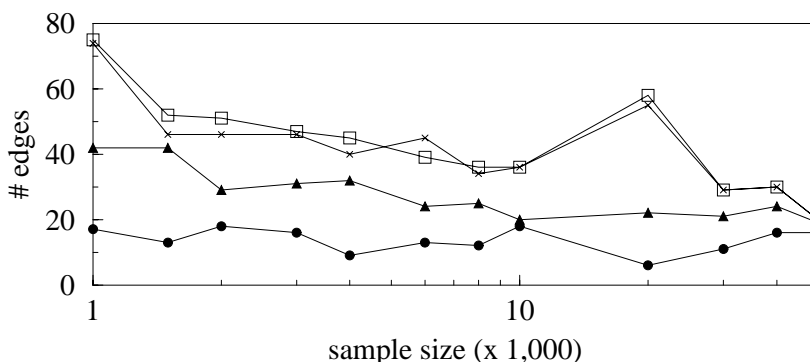


Figure 5.25: The number of edges involved in the largest preliminary ambiguous region is shown for the si-4-path (●), si-3-path (triangles), si-2-path (□), and si-1-path (×) (median of 5 samples of each size).

the least strict si-4-path to the user.

When a strict si-path is employed, the increase in the size of parsimonious ambiguous regions is caused by two facts. First, a stricter si-path yields ambiguous regions of increased sizes. These regions might then merge with each other. As a result, a particularly large ambiguous region is created, while the number of preliminary ambiguous regions is decreased at the same time. Second, the number of computed scores is subject to the heuristics which focus on the neighbors of a variable in the intermediate graph (cf. Section 5.4.3). A stricter si-path yields a denser intermediate graph, and hence a larger number of scores of increased orders are evaluated. One can thus expect a larger number of scores to be smaller than the chosen threshold value. This is depicted in Figure 5.26. Note that the number of scores of zeroth and first order have to be identical for all si-paths because so is the intermediate graph up to first order. The scores of high orders entail the creation of a larger number of rules as well as

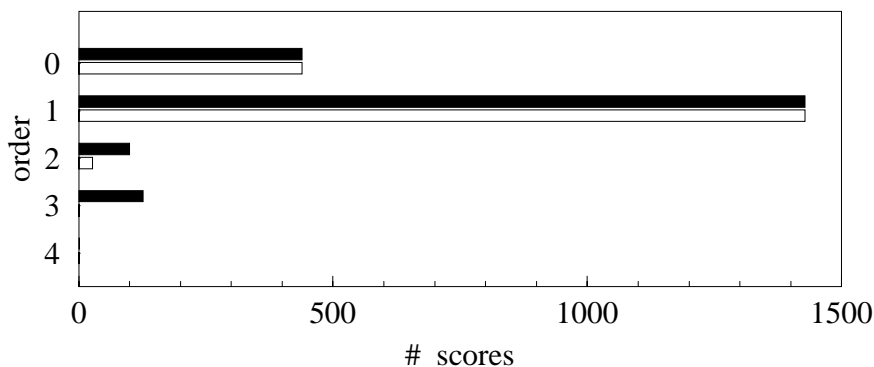


Figure 5.26: The number of scores $g(a, b, S) < \gamma$ found by the algorithm based on the (least strict) si-4-path (white) and the (most rigorous) si-2-path (black) is shown as a function of the order $r = |\mathcal{S}|$ (mean of 5 samples with 1,000 cases each).

of more complex ones. This can additionally increase the number of edges in an ambiguous region.

An increased number of scores of high orders due to strict si-paths can sometimes yield a reduced number of edges being present in the induced skeletons. This can occur because the algorithm can only employ a *necessary* path condition. Moreover, also the approximate search strategies applied to large ambiguous regions can yield the absence of an increased number of edges. This explains why the si-1-path can happen to yield slightly more edges than the (strictest) si-2-path, e.g. at sample size 1,000 in Figure 5.22.

Different Threshold Values

The number of edges present in the induced skeletons can depend on the threshold value γ . In the following experiment, we used the least strict si-path, namely the si-4-path, in the necessary path condition. The posterior probability with conjugate priors and the equivalent sample size $N' = 1$ served as the scoring function, like before. Table 5.5 shows that the choice $\gamma = 0$ entails an edge to be included into the induced skeletons if its presence is favored more than its absence by the relative score. The value $\gamma = 3$ yields its presence only if it is strongly favored, and $\gamma = 10$ corresponds to an extremely strong confidence in its presence. As expected, the number of edges present in the induced skeletons generally decreases as the values of γ grows (cf. Table 5.9). Regarding large sample sizes, the difference in the induced skeletons diminishes. This is not unexpected, as the relative scores approach $\pm\infty$ in the asymptotic limit. Hence, the induced skeletons can be expected to become independent of a particular finite value of γ when the data set is sufficiently large. Conversely, the difference in the induced number of edges becomes notable when given small sample sizes (cf. Table 5.9). Apparently, the impact of a large value of γ increases considerably when the sample size drops. As noted in Section 3.1.4, a positive threshold value γ can also be understood as *a priori* penalizing graphs with

sample size	overall number of present edges			number of mistakenly-present edges		
	$\gamma = 0$	$\gamma = 3$	$\gamma = 10$	$\gamma = 0$	$\gamma = 3$	$\gamma = 10$
1,000	40.0	37.2	31.4	0.6	0.4	0.8
1,500	41.6	40.8	37.4	0.4	0.6	0.6
2,000	43.4	42.8	40.4	0.8	1.0	0.4
3,000	43.8	43.8	42.4	0.8	0.8	0.8
4,000	44.4	43.6	42.6	0.8	0.8	1.0
6,000	44.6	45.0	44.2	0.6	1.0	1.0
8,000	45.0	45.0	45.0	1.0	1.0	1.0
10,000	44.8	44.8	45.0	0.2	0.2	0.6
20,000	45.2	45.2	45.2	0.2	0.2	0.2
30,000	45.0	45.0	45.0	0.0	0.0	0.0
40,000	45.0	45.0	45.0	0.0	0.0	0.0
50,000	45.0	45.0	45.0	0.0	0.0	0.0

Table 5.9: The induced skeletons can depend on the applied threshold value γ .

a large number of edges. The effect of γ can hence be viewed as a consequence of the prior probability having an increased impact on the posterior when the sample size drops.

The value of γ does not seem to have a strong impact on the number of edges mistakenly present (cf. Table 5.9). Moreover, a small non-negative value of γ tends to yield slightly fewer edges mistakenly present than does a larger value of γ in Table 5.9. This might be unexpected according to the previous remarks. A thorough examination reveals, however, that this is caused by the heuristics reducing the number of computed scores (cf. Section 5.4.3): a large value of γ can cause that an increased number of edges is absent due to scores of low orders. Hence, a smaller number of scores of increased orders is computed subsequently. This can, however, leave an increased number of edges present in the graph, in particular those edges whose absence can be determined solely on the basis of negative scores of *high* orders.

Various Scoring Functions

Besides the threshold value γ , also the scoring function itself can have an impact on the induced skeletons. The skeletons entailed by the following scoring functions are compared to each other in this section (cf. also Section 3.1.4):

- the posterior probability with conjugate priors and the equivalent sample size $N = 1$; the threshold value is $\gamma = 3$,
- the Bayesian Information Criterion (*BIC*) and $\gamma = 3$,
- the Akaike Information Criterion (*AIC*) and $\gamma = 3$,
- the popular χ^2 -test, with the significance level $\alpha = 1\%$.

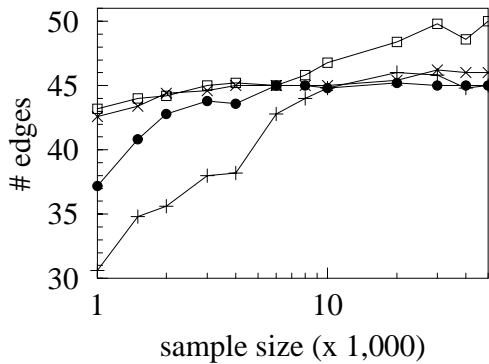


Figure 5.27: The number of edges due to the χ^2 -test (\square), AIC (\times), BIC ($+$), and posterior probability (\bullet). The least strict necessary path condition is used, and the mean of 5 samples of each size is shown.

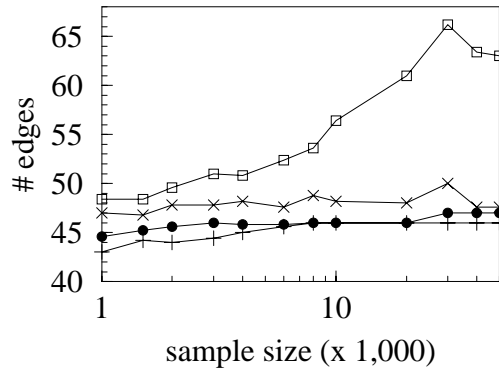


Figure 5.28: The K2 algorithm [34] is used to induce the number of edges which is close to optimum with respect to the χ^2 -test (\square), AIC (\times), BIC ($+$), and posterior probability (\bullet) (mean of 5 samples of each size).

In these experiments, the learning algorithm is based on the least strict necessary path condition employing the si-4-path (cf. Definition 4.16). Figure 5.27 shows the number of edges present in the induced skeletons according to the various scoring functions. As expected, the number of edges generally drops when the sample size decreases. However, the number of edges varies considerably with respect to the different scoring functions. Apparently, the χ^2 -test yields the most edges. In contrast, the BIC removes the most edges, given samples with fewer than 10,000 cases. The skeletons due to the AIC comprise a slightly smaller number of edges than the skeletons yielded by the χ^2 -test, in particular when the samples are smaller than 10,000 cases. The various behaviors can be understood when the different penalties regarding model complexity are considered, since model fit is measured in the same manner by all three scoring functions (except for the posterior probability), namely in terms of the maximum likelihood ratio.

For illustration, assume that the degrees of freedom d_i , as given in Equation 3.1.21, range between 2 and 10 for most of the relative scores. These are reasonable values because most variables are binary, and only some have 3 or 4 states (cf. Table B.1). Moreover, most of the relative scores are of zeroth or first order (cf. also Figure 5.26). The values of the various penalty terms are shown in Table 5.10, where the *effective* penalty term is denoted, i.e. it includes also the threshold value γ acting as an additional penalty. Apparently, the AIC entails slightly larger effective penalties than the χ^2 -test. However, this is mainly because of the threshold value $\gamma = 3$. In the case of a vanishing threshold value, the AIC involves a slightly smaller effective penalty than the χ^2 -test. Since these penalties are approximately the same, also the numbers of induced edges are nearly the same when given samples with fewer than about 10,000 cases. In contrast, the penalty term in the BIC takes on much larger values even at the relatively small sample size of 1,000 cases. Hence, a considerably smaller number of edges is entailed to be present than it is done by the other scoring functions, given samples with

score	effective penalty term	penalties for	
		$d_f = 2$	$d_f = 10$
<i>BIC</i>	$d_f \log(N)/2 + \gamma$	9.9	37.5
<i>AIC</i>	$d_f + \gamma$	5.0	13.0
χ^2 -test	$\chi^2_{1-\alpha=0.99}(d_f)/2$	4.6	11.6

Table 5.10: The effective penalties (penalty term + threshold value γ) involved in the different relative scores, given a sample size of $N = 1,000$ and the threshold value $\gamma = 3$.

fewer than about 10,000 cases. The number of edges in Figure 5.27 suggests that the penalty term inherent in the posterior probability is somewhere between the ones of the *AIC* and the *BIC*. Since the *BIC* is an asymptotic approximation to the posterior probability, the numbers of edges induced by means of the two scoring functions approach each other at large sample sizes.

Figure 5.27 suggests that the χ^2 -test is the "best" scoring function regarding samples with 6,000 or fewer cases. This is meant in the sense that the number of induced edges is closest to the number of 46 edges present in the original alarm-network structure. Furthermore, the posterior probability seems to be worse than the *AIC* at small sample sizes. However, these conclusions do not hold when Figure 5.28 is taken into account, which shows the numbers of edges close to optimum, with respect to the various scoring functions:⁵ These numbers of edges were found by the K2 algorithm [33,34], known for finding DAGs very close to optimum when a correct ordering on the variables is given as input. This is supported by the alarm-network experiments [33, 34, 81], where the K2 algorithm only tended to include an edge too many compared to the optimal DAG. Let us also note that greedy local search, starting out from the original network structure, led to numbers very similar to the ones in Figure 5.28. Obviously, the original network structure does not necessarily coincide with the DAG being optimal with respect to the different scoring functions. In particular, the "optimum" with respect to the χ^2 -test contains considerably more edges than the original DAG, whereas the *AIC* yields only slightly more edges than the original 46 edges (cf. Figure 5.28). The posterior probability as well as the *BIC* yield a slightly smaller number than 46 edges when given small data sets, but both seem to approach the correct number of 46 edges at large sample sizes. Hence, the Figures 5.27 and 5.28 support that the skeletons found by the constraint-based approach, possibly employing a necessary path condition, tend to contain a smaller number of edges than the optimal DAGs, irrespective of the scoring function used (cf. Section 4.3.4).

Regarding the χ^2 -test employed in the constraint-based approach, two counteracting effects are apparent. On the one hand, the χ^2 -test entails "optimal" DAGs which contain (considerably) more edges than the original one. On the other hand, the constraint-based approach tends to yield skeletons with fewer edges than optimal. In the alarm-network experiments, these two

⁵There is no absolute scoring function corresponding to the χ^2 -test applied like a "relative" scoring function, as mentioned in Section 3.1.4. Hence, the depicted numbers correspond to the DAGs which cannot be improved significantly by including an additional edge.

effects appear to balance in the sense that the induced skeletons contain neither considerably fewer nor notably more edges than the original alarm-network structure for a large range of sample sizes. However, this may not hold in general when given other data sets.

Reduced Number of Scores

Finally, let us take a look at the heuristics aimed at reducing the number of relative scores which have to be computed. As described in Section 5.4.3, these heuristics focus on the neighbors of each variable in the intermediate graph. The obtained results are shown in Figure 5.27, where skeletons with more than 46 edges are found from samples containing over 10,000 cases. In contrast, Figure 5.29 sketches the results obtained *without* this heuristics. These skeletons were determined on the basis of *all* relative scores up to third order. Calculating all the scores of fourth or higher orders was computationally infeasible in the alarm-network experiment. Without applying this heuristics a decreased number of edges is found, as expected, since the induced skeletons are based on an increased number of scores in this case. These skeletons are very close to the original graph, except for extremely small samples. Furthermore, the skeletons obtained without this heuristics contain only a very small number of edges erroneously (cf. Figure 5.29). Hence, when all the scores are taken into account, the constraint-based approach and the χ^2 -test seem to balance extremely well in this experiment, in the sense that the induced skeletons are very close to the original structure over a large range of sample sizes.

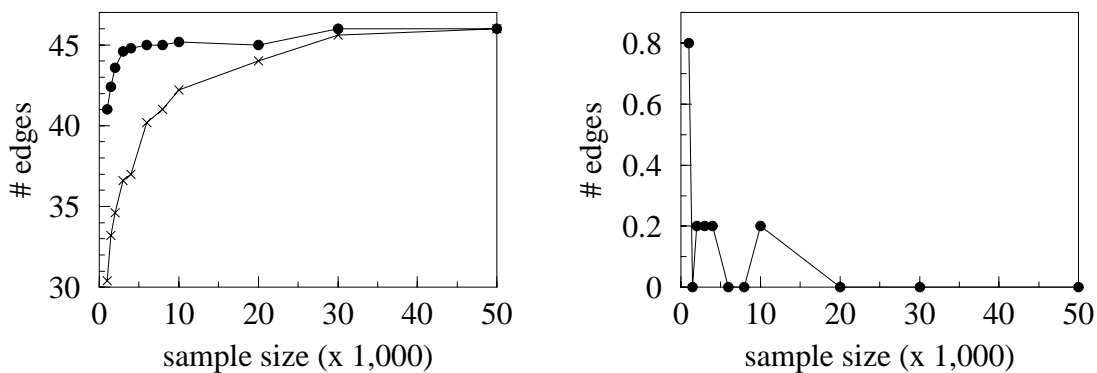


Figure 5.29: In the left diagram, the number of edges present in the induced skeletons is based on *all* scores up to third order, i.e. without the heuristics reducing the number of scores. This number approaches the correct value of 46 edges at sample size 50,000. The skeletons are induced by means of the least strict necessary path condition (●). For illustration, also the number of certainly-present edges is shown (×), as it equals the number of edges found by the SGS algorithm [142]. The right graph shows that only an extremely small number of edges is erroneously present. The χ^2 -test with significance level $\alpha = 1\%$ served as the "relative" scoring function, and the mean of 5 samples of each size is shown.

5.7 Discussion

When given *finite* sample sizes, the necessary path condition led to considerable improvements in our computer experiments, in particular when the data set was of some "medium" size, i.e. not extremely small. One cannot expect to induce some reasonable structure from very small samples, anyway. Our experiments support that all constraint-based approaches – whether they employ a necessary path condition or not – induce the same graph, i.e. the perfect map, in the asymptotic limit if the data was sampled from a faithful probability distribution (cf. also Section 4.1).

In our experiments, the differences among the various variants of si-paths are rather negligible compared to state-of-the-art constraint-based algorithms, which do not employ a necessary path condition, because they assume a faithful probability distribution. The benefits of a necessary path condition applied to finite data are twofold. First, the number of induced edges is considerably increased in our experiments, and is hence closer to the optimal number. Moreover, possibly several skeletons are found instead of a single graph. Model uncertainty regarding the presence of edges can thus be discovered. Of course, this is only possible to a limited degree, as the exact problem is intractable in large domains. The induced skeletons can be visualized in a single graph, the so-called summary graph which utilizes the concept of ambiguous regions and the fact that the various skeletons have many edges in common [144]. This so-called summary graph is typically much easier to interpret than an enumeration of the various induced skeletons. In particular, it is often beneficial to consider the parsimonious summary graph rather than the preliminary one, as shown in our experiments.

Moreover, the influence of the used threshold value γ was found to be rather small in our alarm-network experiments. We also compared the various scoring functions with each other, including the popular χ^2 -test. It was discussed that the "optimal" DAG with respect to the χ^2 -test contains too many edges. As the constraint-based approach tends to yield too few edges, these two effects seem to balance, and the resulting skeletons were very close to the original DAG, in particular when the heuristics for reducing the number of computed scores was not used. The optimal DAG with respect to the posterior probability appeared to be closest to the original alarm-network structure. This suggests to use this scoring function also in the constraint-based approach.

Not much attention has been paid to an early version of the PC algorithm which required certain undirected paths to be present [140]. This variant was also mentioned in [142], where it was called PC* algorithm. It starts out with the complete skeleton, and removes an edge from the current graph when a conditional independence is induced. A test on the independence of the variables a and b conditional on the set $S \subseteq \mathcal{V} \setminus \{a, b\}$ is only carried out when this set is a subset of the neighbors of a or b , and when each $s \in S$ lies on an undirected path between a or b in the current skeleton. This algorithm differs in several respects from the presented algorithm employing the necessary path condition. The main difference between this scheme and the necessary path condition is the *kind* of paths considered. Both the si-path and the sc-path are not only concerned with the presence of certain edges but also with the induced independences and

dependences. This reduces the number of "paths" under consideration greatly. Moreover, the si-path and the sc-path are derived from DAGs which are optimal with respect to the used scoring function, while the undirected paths required by the PC* algorithm are a consequence of the assumed perfect map. At finite data, the paths required by the latter can, however, entail too many edges being present, compared to the optimal DAG. This was discussed – with a slightly different focus – in Section 4.4, where it was shown that an undirected path is not necessarily required to be present in an optimal DAG. Since the PC* algorithm starts out with a complete skeleton, the number of undirected paths in the intermediate graph can be extremely large at early stages of the learning process. In fact, it was found to be computationally infeasible, except for domains containing a very small number of variables. In contrast, the scheme proposed in this thesis was computationally efficient in our experiments. This has various reasons. First, the number of si-paths and sc-path depends on the number of conditional independences induced. Hence, our scheme does not have a bottleneck at the beginning of the learning process, like the PC* algorithm. Second, the necessary path condition is represented in terms of *rules*. As an advantage, the existence of a "path" need not be determined from a *graph*. In fact, after the rules have been simplified the graphs are immediately obtained. Moreover, our algorithm is capable of inducing possibly several skeletons, which reflect model uncertainty (to some limited degree, of course), whereas the PC* algorithm induces as single graph assumed to be the perfect map.

5.8 Incorporating Prior Knowledge

The algorithm, as presented so far, is completely data-driven. In many practical situations, some additional information might be at hand, particularly when the directed edges in the DAG are interpreted in a causal manner. In these cases, the presence or absence of some edges as well as their orientations might be known *a priori*. This knowledge can stem from common sense or laws of nature, for instance. Temporal ordering on the variables, if known, can often help determine the orientations of edges.

Prior knowledge can easily be included into the scoring function, especially in a Bayesian setting when the posterior probability or an approximation to it is employed. An *a priori* belief which favors a structure to some *degree* can, for instance, be accounted for by constructing an *a priori* DAG, and by penalizing an induced DAG dependent on its structural difference to it. This is described in [81], where also measures for structural differences are discussed. In general, when such a measure of structural differences accounts for the orientations of edges, the resulting prior probabilities of equivalent DAGs cannot be expected to be identical, i.e. *prior equivalent*. Hence, such a scoring function is not score-equivalent in general. It is usually desired that the measure for structural differences is such that the resulting (absolute) scoring function stays decomposable, which allows relative scoring functions to be used. When score-equivalence is not given, the relative scores $g(a, b, \mathcal{S})$ and $g(b, a, \mathcal{S})$ typically are not equal (cf. Section 3.1.3).

The remainder of this section is concerned with prior knowledge where the expert is *sure* about

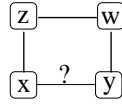


Figure 5.30: Prior knowledge can prevent the elimination of the edge $x \sim y$.

some structure. This certainty can be incorporated in the scoring function in a similar fashion as before. However, the corresponding scores approach plus or minus infinity in this case. Alternatively, this knowledge can directly be applied to the rules. This allows the computation of the relative scores without taking into account this knowledge. Let us hence consider the creation of the rules according to the si-3-path, as the latter is the simplest si-path in Section 4.3 which is capable of accounting for non-score-equivalent scoring-functions. Of course, this is only possible up to a degree, since the si-3-path applies to skeletons rather than DAGs.

Prior knowledge concerning the *presence* or *absence* of an edge can easily be incorporated into the rules. If the presence of an edge is considered to be certain, no rules are generated by the algorithm concerning this edge, like for certainly-present edges (cf. Section 5.1). In the other case, where the absence of an edge $a \sim b$ is taken for granted, the i-rule $IR([a, b], \emptyset, \emptyset)$ is created, indicating that this edge is certainly absent. Note that the si-3-path does not involve sc-paths.

Certainty regarding the *orientations* of edges can only be accounted for in a limited way by means of rules. Nevertheless, it can be useful to incorporate this knowledge in the learning algorithm already at this stage. When a score $g(a, b, \mathcal{S}) < \gamma$ is transformed into a rule, the knowledge about the orientations of edges can apply to the edge $a \sim b$ or to some of the edges $a \sim s$ where $s \in \mathcal{S}$. Let us focus on the latter case, as this is the more interesting one. A score $g(a, b, \mathcal{S}) < \gamma$ is transformed into a rule $IR([a, b], \{[a, s] : s \in \mathcal{S}\}, \{[b, s] : s \in \mathcal{S}\})$ (according to the si-3-path) only if all the variables in \mathcal{S} are allowed to be parents of variable a . In other words, if \mathcal{S} contains a variable s such that $a \rightarrow s$ is required by the prior knowledge then no rule is created.

The effect of incorporating constraints on orientations is illustrated in the simplistic example in Figure 5.30, where all edges are induced to be absent, except for $x \sim y$, $x \sim z$, $y \sim w$ and $z \sim w$. Let us now consider whether the induced score $g(x, y, \{z\}) = g(y, x, \{z\}) < \gamma$ yields the absence of the edge $x \sim y$. Without prior knowledge, this edge would clearly be absent, because z is adjacent to x and there is a path from z to y . However, when the prior knowledge requires that the edge $x \sim z$ is oriented like $x \rightarrow z$ then the score $g(x, y, \{z\}) < \gamma$ has to be disregarded. Only the score $g(y, x, \{z\}) < \gamma$ is thus transformed into a rule, namely into $IR([y, x], \{[y, z]\}, \{[x, z]\})$. Since y and z are not adjacent in the graph, the edge $x \sim y$ cannot be absent. In comparison to that, the scheme applied by the PC algorithm [142] would remove this edge because of the score $g(y, x, \{z\}) < \gamma$.⁶

⁶In the case where the edge $z \sim y$ is removed at an early stage, the heuristics of focusing on neighbors entails that the computation of the score $g(y, x, \{z\})$ is skipped, and thus the edge $x \sim y$ is induced to be present by the PC algorithm, too. However, if the edge $z \sim y$ is absent because of the score $g(z, y, \{w\}) < \gamma$, it is possible that the score $g(y, x, \{z\})$ is computed by the PC algorithm, and hence the edge $x \sim y$ is removed.

Since many scores are possibly disregarded due to constraints, the computational effort can be reduced. For instance, no scores $g(a, b, S)$ need be computed if they refer to edges $a \sim b$ which are a priori known to be present or absent. Also, when S contains a child of a , the evaluation can be skipped. This was noted in [142] concerning the incorporation of prior knowledge in the PC algorithm.

5.9 Incomplete Data

For simplicity, we assume throughout this thesis that the data is *complete*, i.e. a value is assigned to every random variable in every case contained in the given data set. In many practical situations, however, one faces the problem that the available data is *incomplete*. When the values of some random variables are missing in some cases, parameter estimation as well as structural learning become very complicated. An exact solution given incomplete data is often infeasible so that one has to resort to approximations. For an overview, the reader is referred to [35]. When data is missing, it is important to distinguish the situations where data is missing in a systematic manner and the situation where data is missing at random (cf. [130]). In the former case, it is essential to account for the specific data-censoring mechanism in order to attain reasonable results. In the latter case, more general approximations are applicable, like e.g. Monte Carlo methods or mean-field approximations (see also [73]). An overview of these two methods can be found, for instance, in [91]. Various approximations concerning the Bayesian approach are examined in [31]. When the data contains only discrete variables, also the scheme called *bound and collapse* can be applied to structural learning [124]. A very popular approach, given discrete variables, is the EM algorithm [43]. It is an iterative scheme which optimizes the likelihood function. As the latter exhibits local optima [134], the EM algorithm does not necessarily find the global optimum, though. The EM algorithm was applied to structural learning in Bayesian networks in [54].

In the PRONEL project [120], a simple and efficient approach is taken. It might, however, be regarded as a crude approximation. It is assumed that data is missing only *occasionally*, i.e. the probability that a variable in a case is not instantiated is assumed to be small. Moreover, this probability has to be independent of both the random variable itself and its value. Given these assumptions, a relative score $g(a, b, S)$ ($a, b, \in \mathcal{V}$ and $S \subseteq \mathcal{V} \setminus \{a, b\}$) is computed on the basis of only those cases which are "complete" regarding the involved variables a, b and S . The other cases are simply ignored. When the probability for a value to be missing is reasonably small, for example $p_o = 5\%$, one can expect that quite a large fraction of the data set can be used for computing the score $g(a, b, S)$, particularly when it is of reasonably low order $|S|$. Low orders are desired in hypothesis tests also because the results are more reliable in general.

For instance, a score of zeroth order involves the data concerning two variables. In this situation, a fraction of $(1 - p_o)^2 \approx 90\%$ of the cases in the data set can be expected to be complete. Scores of third order involve $2 + 3 = 5$ variables so that one can expect a fraction of $(1 - p_o)^5 \approx 77\%$ cases to be complete regarding the five variables of interest. In contrast, if the domain involves 37 variables, like in the alarm network, then only $(1 - p_b)^{37} \approx 15\%$ of

the cases can be expected to be complete. This illustrates that a large fraction of the data can be used for computing scores, which are typically of low orders, even if only a rather small fraction of the cases is actually complete.

Besides incomplete data, also *hidden* or *latent* variables might play an important role in practice. A variable is called hidden or latent if its values are never observed. When learning in Bayesian networks with hidden variables, one has to be aware of two facts. First, the so-called *aliasing* can entail problems [30,31]. Second, the probability distribution described by a Bayesian network with hidden variables belongs to the so-called *stratified* exponential family, whereas a Bayesian network without latent variables describes a distribution which belongs to the *curved* exponential family [64–66]. An important implication of this fact is that the *BIC* is a valid asymptotic approximation of the log marginal likelihood in the latter case [78], whereas this might not hold in the presence of hidden variables. For details on exponential families, the reader is referred to [6]. Moreover, let us mention that, in a causal discovery setting, the existence of hidden variables can be induced in domains comprising continuous variables with a Gaussian distribution, as described in [142].

6

Inducing DAGs given Skeletons

The previous two chapters were concerned with determining skeletons. In order to arrive at directed acyclic graphs (DAGs), this chapter focuses on an operator which orients the edges present in a skeleton. Rather than relying on the induced conditional independences and dependences, like state-of-the-art constraint-based algorithms do, this operator is aimed at inducing the DAG with the highest score from among all the DAGs with the same skeleton. This operator has proven to be quite robust in our experiments when given finite data. It was also outlined – in a slightly different context – in [143].

Constraint-based algorithms tend to yield DAGs with too few edges, compared to the optimum in the space of *all* DAGs (cf. Section 4.3.4). For this reason, a post-processing step to the constraint-based approach is necessary if one aims at inducing local optima with respect to a scoring function. A simple greedy scheme is presented. Finally, our approach is compared to established learning algorithms using artificial and real-world data in our experiments.

6.1 Orienting the Edges

In the following, we describe an operator aimed at finding the optimal orientations, with respect to a scoring function, given a skeleton. The search space is a subspace of the space of DAGs. It contains all those DAGs whose skeleton is identical with the given graph. For practical reasons, model uncertainty regarding the orientations of edges is ignored by this scheme, and exactly one DAG is thus found when given a skeleton.

When structural learning is carried out with a score-equivalent scoring-function, one can typically only aim at inducing the equivalence class rather than a particular DAG (cf. Section 3.1.3). The fact that two DAGs are equivalent if their skeletons and their colliders are identical [148] suggests to orient the edges involved in colliders first and then the remaining edges, as described in [142]. Such an approach requires that the colliders can be induced from the data given the skeleton. Typically, constraint-based approaches achieve this by combining induced conditional independences and dependences [142]. Since this approach is known to be quite

unstable when given finite data sets [142], we present an operator which is based on *scores* assigned to each of the colliders. This renders a *greedy* hill-climbing approach possible, i.e. the colliders are oriented according to their scores, beginning with the largest one.

The operator employs a *partially directed acyclic graph* (PDAG), i.e. a graph which can contain directed as well as undirected edges. At the beginning of the learning process, it contains the same (undirected) edges as the given skeleton, while some edges might be oriented at an intermediate stage. Eventually, all the edges in the PDAG are directed, representing the induced DAG. In the PDAG, a structure which is a candidate for being a collider involves three nodes a , b and c such that there is an edge between a and b as well as between b and c , and no edge between a and c . Such a triple of variables is called a *collider candidate* in the following.

6.1.1 Scores of Collider Candidates

This section is concerned with assigning a score to a collider candidate involving the variables a , b and c . Given an intermediate PDAG m_{int} , the score $g_{col}(a, b, c | m_{int})$ of a collider candidate is based on an idea very similar to the one in Section 3.1.2, which led to *relative* scoring functions. Consequently, g_{col} can be viewed as a relative scoring function regarding colliders. It is obtained by comparing such DAGs with each other which are identical except for the orientations of the edges $a \sim b$ and $b \sim c$ involved in the collider candidate. In particular, the variables a , b and c have the same parents as in the PDAG m_{int} , namely $pa_{m_{int}}(a)$, $pa_{m_{int}}(b)$, and $pa_{m_{int}}(c)$, respectively. The DAG m_{col} containing the collider $a \rightarrow b \leftarrow c$ is depicted in Figure 6.1, while the three alternative DAGs m_{\leftarrow} , m_{\rightarrow} and m_{\leftrightarrow} concerning the orientations of the edges $a \sim b$ and $b \sim c$ are shown in Figure 6.2.

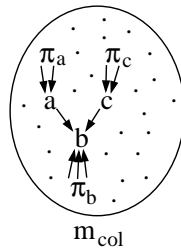


Figure 6.1: The DAG m_{col} contains the collider $a \rightarrow b \leftarrow c$. For brevity, the notation $\pi_a = pa_{m_{int}}(a)$, $\pi_b = pa_{m_{int}}(b) \setminus \{a, c\}$, and $\pi_c = pa_{m_{int}}(c)$ is used. The dots represent the remaining network structure which is identical with the ones in Figure 6.2.

Similarly to the relative scoring functions discussed in Section 3.1.2, we focus on the score of the DAG m_{col} relative to each of the three alternative DAGs. This leads to the following three relative scoring functions regarding collider candidates, where the absolute scoring function is again assumed to be decomposable:

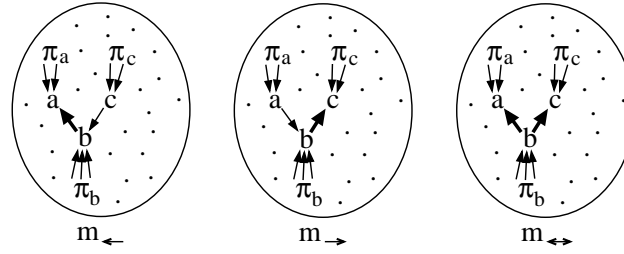


Figure 6.2: The three DAGs showing the alternative orientations of the edges $a \sim b$ and $b \sim c$ compared to the DAG m_{col} in Figure 6.1.

$$\begin{aligned} g_{\leftarrow}(a, b, c \mid m_{\text{int}}) &= f(m_{\text{col}}) - f(m_{\leftarrow}) \\ &= g(b, a, \pi_b \cup \{c\}) - g(a, b, \pi_a), \end{aligned}$$

$$\begin{aligned} g_{\rightarrow}(a, b, c \mid m_{\text{int}}) &= f(m_{\text{col}}) - f(m_{\rightarrow}) \\ &= g(b, c, \pi_b \cup \{a\}) - g(c, b, \pi_c) \\ &= g_{\leftarrow}(c, b, a \mid m_{\text{int}}), \end{aligned}$$

$$\begin{aligned} g_{\leftrightarrow}(a, b, c \mid m_{\text{int}}) &= f(m_{\text{col}}) - f(m_{\leftrightarrow}) \\ &= g_{\leftarrow}(a, b, c \mid m_{\text{int}}) + g(b, c, \pi_b) - g(c, b, \pi_c) \\ &= g_{\rightarrow}(a, b, c \mid m_{\text{int}}) + g(b, a, \pi_b) - g(a, b, \pi_a), \end{aligned}$$

where we have used the relative scoring function g (cf. Section 3.1.2)!¹ As before, the decomposability of the absolute scoring function leads to a considerable simplification. In each of the three cases, the relative scoring function only depends on the three variables involved in the collider candidate as well as on their parents, whereas the remaining variables have no impact. A single score regarding a collider candidate can be obtained by the following definition:

$$g_{\text{col}}(a, b, c \mid m_{\text{int}}) := \min\{g_{\leftarrow}(a, b, c \mid m_{\text{int}}), g_{\rightarrow}(a, b, c \mid m_{\text{int}}), g_{\leftrightarrow}(a, b, c \mid m_{\text{int}})\} \quad (6.1.1)$$

This means that the collider is compared to the best alternative. In other words, if $g_{\text{col}}(a, b, c \mid m_{\text{int}}) > 0$, this score is a lower bound for the increase in the absolute score, when these edges are oriented like a collider compared to one of the alternative graphs without that collider. If $g_{\text{col}}(a, b, c \mid m_{\text{int}}) < 0$, it is a bound for the maximal decrease in the absolute score when these edges are oriented like a collider compared to one of the alternative orientations. This definition hence increases the robustness of the operator described in the following.

¹Since the scoring function of collider candidates involves the *differences* in the relative scoring function g , the χ^2 -test is not applicable here, as there does not exist a corresponding absolute scoring function.

6.1.2 Orienting the Colliders

Let us first introduce some notation. The operator presented in the following uses two kinds of directed edges, a *proposed* orientation and a *fixed* one. The former is denoted by an arrow like " \rightarrow ", whereas the latter is sketched by a double-arrow, like " \Rightarrow ". This distinction will become clear in the next section. At the moment, only the fixed orientations (" \Rightarrow ") are relevant.

In the first step, the operator determines the colliders. A collider candidate in the intermediate PDAG m_{int} has to be of the form $a \leftrightarrow b \leftarrow c$ where $\leftrightarrow \in \{\sim, \Leftarrow\}$ and $\leftarrow \in \{\sim, \Rightarrow\}$ (" \leftrightarrow " and " \leftarrow " are dummy arrows). In particular, the edges are not allowed to be already oriented like a collider $a \Rightarrow b \Leftarrow c$, or like $a \Leftarrow b$ or $b \Rightarrow c$. After the score $g_{col}(a, b, c | m_{int})$ has been calculated for each collider candidate (cf. Section 6.1.1), the collider candidate with the largest score $g_{col}(a, b, c | m_{int}) > \gamma$ is oriented like $a \Rightarrow b \Leftarrow c$, where γ is the used threshold value (cf. Table 5.5). For example (cf. Figure 6.4 (1)), assume that both the colliders $a \rightarrow b \leftarrow c$ and $b \rightarrow c \leftarrow d$ are favored by the scores $g_{col}(a, b, c | m_{int}) > \gamma$ and $g_{col}(b, c, d | m_{int}) > \gamma$. Apparently, both colliders cannot be present simultaneously. This might indicate model uncertainty regarding the orientations of edges. Since the operator is greedy, this inconsistency is resolved by choosing the collider with the largest score. In this example, this is assumed to be the collider $b \rightarrow c \leftarrow d$.

After a collider has been oriented, it has to be ensured that a directed cycle cannot occur in the PDAG after an additional edge will be oriented. This is accounted for by orienting such edges in the contrary direction such that the occurrence of a directed cycle is avoided. For simplicity, the scoring function is disregarded at this step. After a collider or an edge have been oriented in the PDAG m_{int} , for some $a, b, c \in \mathcal{V}$ the scores $g_{col}(a, b, c | m_{int})$ have to be updated, taking into account changes in the parents of some variables. The operator cycles through the above steps as long as there are collider candidates with a score larger than the threshold value γ . As a result, this scheme yields a PDAG where the colliders have been determined.

6.1.3 Orienting the Remaining Edges

After the colliders have been found, the algorithm orients the remaining undirected edges. Guided by the heuristics of parsimony, the aim is to find the orientations of the remaining edges such that a minimal number of additional colliders occurs. In the ideal case, where a perfect map of the probability distribution for the variables exists, this is possible without introducing an additional collider [142].

For simplicity, a greedy scheme is employed. It comprises two procedures, namely *ProposeOrientations* (cf. Algorithm 6.1) and *FixOrientations* (cf. Algorithm 6.2). The former proposes the orientations of edges by arrows of the kinds " \rightarrow ", " \leftarrow " and " \leftrightarrow ". These directions are only temporarily valid, and might be changed by the latter procedure. The edges oriented like " \Rightarrow " and " \Leftarrow " by *FixOrientations* are *fixed* in the sense that their orientations cannot be altered in the remaining process.

```

procedure ProposeOrientations
input: PDAG  $m_{int}$  with edges  $\sim, \Rightarrow, \Leftarrow$ .
output: PDAG  $m_{int}$  with edges  $\sim, \Rightarrow, \Leftarrow, \rightarrow, \leftarrow, \leftrightarrow$ .
(1)  $\forall a, b, c \in \mathcal{V}$ : if  $a \leftrightarrow b \leftarrow c$  with  $\leftrightarrow \in \{\Rightarrow, \rightarrow, \leftrightarrow\}$  and
 $\leftarrow \in \{\sim, \leftarrow\}$  and no edge between  $a$  and  $c$  then substitute
either  $b \sim c$  by  $b \rightarrow c$  or  $b \leftarrow c$  by  $b \leftrightarrow c$ .
(2)  $\forall a, b \in \mathcal{V}$ : while  $\exists a \leftarrow b$  and  $\exists a = x_1, \dots, x_q = b$  ( $q > 2$ )
such that  $x_{i-1} \leftrightarrow x_i$  ( $i = 2, \dots, q$ ) with  $\leftrightarrow \in \{\Rightarrow, \rightarrow, \leftrightarrow\}$  then
substitute  $a \leftarrow b$  by  $a \leftrightarrow b$ .
(3) while edges can be oriented go to (1).

```

Algorithm 6.1: The procedure *ProposeOrientations* tries to orient the edges in such a way that no additional colliders occur. If this is impossible then the involved edges are oriented in "both" directions, indicated by " \leftrightarrow ".

```

procedure FixOrientations
input: PDAG  $m_{int}$ , data  $D$ 
output: PDAG  $m_{int}$ 
(1) call ProposeOrientations.
(2) substitute the edges  $a \rightarrow b$  by  $a \Rightarrow b$ .
(3) among all structures of the form  $a \leftrightarrow b \leftrightarrow c$  without
an edge between  $a$  and  $c$ , orient the one with the highest
score  $g_{col}(a, b, c | m_{int})$  like  $a \Rightarrow b \Leftarrow c$ .
(4) if edges like  $\leftrightarrow$  are present, substitute them all by
 $\sim$  and go to (1).

```

```

each time an edge has been oriented like  $\Rightarrow$  or  $\Leftarrow$ :
(*)  $\forall a, b \in \mathcal{V}$ : while  $\exists a \leftarrow b$  with  $\leftarrow \in \{\sim, \leftarrow, \rightarrow, \leftrightarrow\}$  and
 $\exists a = x_1, \dots, x_q = b$  ( $q > 2$ ) such that  $x_{i-1} \Rightarrow x_i$  ( $i = 2, \dots, q$ )
then orient  $a \Rightarrow b$ .

```

Algorithm 6.2: This procedure resolves inconsistencies regarding the orientations of edges (indicated by " \leftrightarrow ") by using the scoring function g_{col} .

The procedure *ProposeOrientations* (cf. Algorithm 6.1) aims at orienting the edges such that an additional collider does not occur. In particular, step (1) is concerned with avoiding colliders. If it is impossible to find an orientation of the edges without introducing an additional collider, the involved edges are marked as " \leftrightarrow ". For illustration, Figure 6.3 (1) sketches a PDAG, in which the induced colliders lead to inconsistencies when orienting the edges not yet directed. This is because the collider $a \Rightarrow c \Leftarrow b$ requires the edge between c and d to be oriented like $c \rightarrow d$ which then entails the orientation $d \rightarrow e$, and finally $e \rightarrow f$ (see Figure 6.3 (2)). This results, however, in an additional collider at variable f . In contrast, when the orientation process had begun at the other collider, $g \Rightarrow f \Leftarrow h$, the occurrence of an additional collider at f , e or d could only have been avoided by orienting the edges like $e \leftarrow f$, $d \leftarrow e$ and $c \leftarrow d$ (cf. Figure 6.3 (3)). This entails, however, an additional collider at c . Hence, the edges cannot be oriented without introducing an additional collider in either one of the cases. Hence,

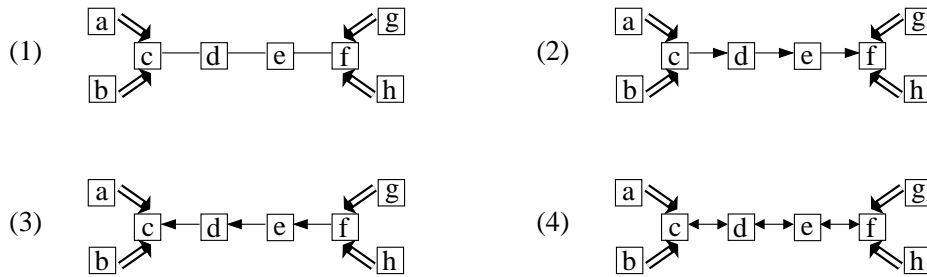


Figure 6.3: This simplistic example shows how the procedure *ProposeOrientations* indicates inconsistencies regarding the orientations of edges.

the procedure *ProposeOrientations* (cf. Algorithm 6.1) orients the involved edges in "both" directions, indicated by edges of the sort " \leftrightarrow ". This is depicted in Figure 6.3 (4).

Since directed cycles can occur at step (1) of Algorithm 6.1, the second step orients edges involved in cycles in "both" directions (" \leftrightarrow "). This is illustrated in Figure 6.4. The single collider, $b \Rightarrow c \Leftarrow d$, causes step (1) to yield a directed cycle, namely $d \Rightarrow c \rightarrow g \rightarrow f \rightarrow e \rightarrow d$. The involved edges are thus marked as " \leftrightarrow " in step (2).

The inconsistencies indicated as " \leftrightarrow " by *ProposeOrientations* (cf. Algorithm 6.1) can be related to model uncertainty regarding the orientations of edges. Our operator resolves these inconsistencies in a greedy way for simplicity so that a single DAG is eventually induced. This is done by the procedure *FixOrientations* (cf. Algorithm 6.2). In step (2) of the procedure *FixOrientations*, the orientations of those edges are *fixed* (by a double-arrow " \Rightarrow " or " \Leftarrow ") which are not involved in an inconsistency. However, when the procedure *ProposeOrientations* yields an edge of the type " \leftrightarrow ", an additional collider has to be introduced. Such a collider is determined in step (3) of the procedure *FixOrientations* (cf. Algorithm 6.2). This is done in a greedy way, namely by orienting that collider candidate $a \leftrightarrow b \leftrightarrow c$ as a collider which is assigned the highest score $g_{\text{col}}(a, b, c | m_{\text{int}})$. Unlike when determining colliders in Section 6.1.2, the scores concerning collider candidates are allowed to be negative here. In case of a negative score, orienting the highest-scoring edges like a collider corresponds to decreasing the absolute score by as little as possible.

Like the procedure *ProposeOrientations*, also *FixOrientations* has to ensure that a directed cycle cannot occur when fixing the orientations. Hence, step (*) has to be carried out each time an edge has been oriented like " \Leftarrow " or " \Rightarrow " by the procedure *FixOrientations*. The step (*) focuses on possible cycles due to edges whose orientations are already *fixed*, i.e. edges of the kind " \Rightarrow ", whereas step (2) in *ProposeOrientations* (cf. Algorithm 6.1) accounts also for proposed orientations (" \rightarrow "). In step (4) of *FixOrientations*, the edges indicating inconsistencies are altered to undirected ones in the PDAG before the steps (1) through (4) are repeated. The interplay between the two procedures *ProposeOrientations* and *FixOrientations* is illustrated in Figure 6.4.

After *FixOrientations* has resolved all the inconsistencies, undirected edges might still be

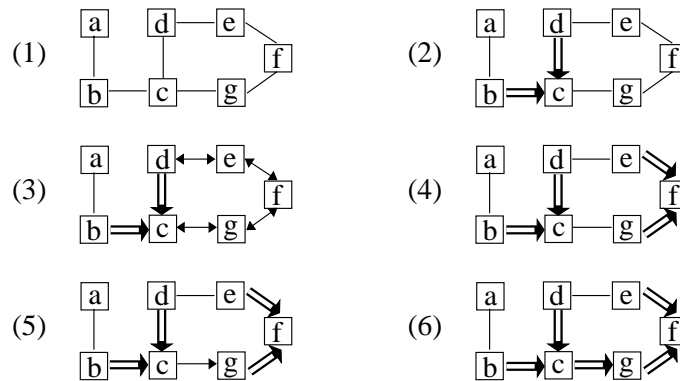


Figure 6.4: This example illustrates the interplay between the procedures *ProposeOrientations* and *FixOrientations* when an inconsistency due to a directed cycle occurs. It is assumed that $b \Rightarrow c \Leftarrow d$ is the only collider induced according to Section 6.1.2, cf. PDAG (2). When the remaining edges are oriented, *ProposeOrientations* indicates inconsistencies (" \leftrightarrow ") due to the possibility of a directed cycle, cf. PDAG (3). Hence, an additional collider has to be introduced. *FixOrientations* chooses the one with largest score. Assume that this is the collider at variable f , cf. PDAG (4). In the subsequent steps, no inconsistencies occur, and the orientation of the edge $c \sim g$ can first be proposed in PDAG (5), and finally be fixed in PDAG (6). The edges $a \sim b$ and $d \sim e$ can be oriented in either direction in equivalent DAGs.

present in the PDAG, e.g. the edges $a \sim b$ and $d \sim e$ in Figure 6.4 (6). These undirected edges are oriented in a final step of the edge-orientation procedure by randomly picking one of those edges and by assigning to it a random orientation. Since further inconsistencies might arise, the procedure *FixOrientations* is called in order to orient the edges affected by this assignment. This is repeated until all edges have been oriented so that a DAG is eventually obtained.

6.1.4 Experiments with the Operator

In the following experiments, some light is shed on the stability of the presented heuristic operator orienting the edges. The operator is used together with simple strategies for including and removing edges, resulting in a search strategy for finding DAGs. Let it be called the *skeleton search strategy* in the remainder of this section. In order to pronounce the power of the above operator, the skeleton search strategy is quite simple on purpose. It works as follows: starting out with the empty DAG, the graph is optimized by in turn carrying out rounds comprising the following three steps:

- (i) apply the presented operator,
- (ii) carry out one step of forward inclusion in the space of DAGs,
- (iii) perform backward elimination in the space of DAGs.

After the orientations of the edges have been optimized by the presented operator in step (i), the second step includes the edge which improves the score by most. Of course, it has to be ensured that no directed cycles occur in the resulting graph. In this scheme, at most *one* edge is included in each round so that the presented operator can optimize the orientations before the next edge is included. The third step allows the algorithm to remove edges. They might have been included in previous rounds where the orientations of some edges might have been different. It can occur that this simple search strategy oscillates among two or more DAGs at the end of the search process, i.e. it may not converge to a unique DAG. This is because the presented operator is non-local in the sense that it can simultaneously change the orientations of more than one edge. For instance, it can occur that an edge is included into the graph after optimizing the orientations in the one round, whereas in the successive round, the above operator orients edges differently entailing the removal of that edge again, and so on. The stopping criterion of the algorithm has to account for this. We thus keep track of the most recent DAGs and eventually choose the one with the highest score. For simplicity, the aim of this scheme is to induce a (local) optimum rather than to account for model uncertainty.

The time-evolution of a typical search process is depicted in Figure 6.5, where the data set was sampled from the alarm network [8] (see also Appendix B). In this experiment, the posterior probability with conjugate priors and the equivalent sample size $N = 1$ served again as the scoring function, see e.g. [81] and Section 3.1.4. Also, we committed ourselves to a uniform prior for the DAGs, i.e. $p(m) = \text{const}$, and imposed no constraints on the network structures. The threshold value $\gamma/2 = 3$ was used in order to include only those edges into the graph which lead to some notable increase in the score of the DAGs (cf. Table 5.5). Since at most one edge is added in each round, the overall number of rounds cannot be smaller than the number of edges in the induced DAG. As shown in Figure 6.5, the number of edges in the intermediate graph increases rather quickly before the end of the search process is reached. Since the information about a correct ordering on the variables is not given as input to our

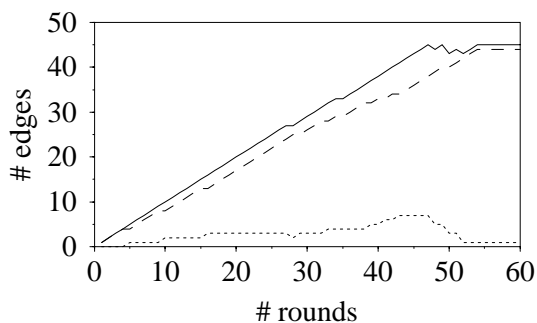


Figure 6.5: Time evolution of the number of edges during the learning process in one of our experiments with the alarm network. A sample with 2,000 cases is used. The solid line indicates the overall number of edges present in the intermediate graph m_{int} after each round; this number is subdivided into the edges which are present in the original alarm network (dashed line) and the ones which are not (dotted line).

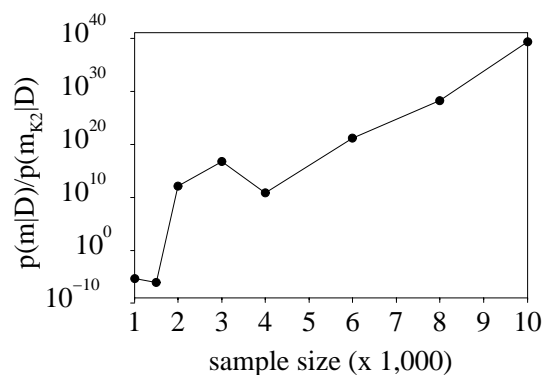


Figure 6.6: In our alarm-network experiments, the posterior probability of the DAG m induced by the skeleton search strategy is compared to the DAG m_{k2} resulting from the K2 search strategy. The geometric mean of 5 samples of each size is depicted.

algorithm, edges are "erroneously" included into the graph during the search process. Towards its end, after the orientations of the edges have been established, most of the edges erroneously included are again removed in this example.

Since a (partial) ordering on the variables is known in this experiment, the K2 search strategy [33, 34] is applicable, and can be used in comparison. Since the K2 algorithm requires an initially specified ordering on the variables, one can expect this search strategy to be quite stable, even when given rather small samples. This was confirmed in [81], where the DAGs with the smallest structural differences to the original alarm-network structure were induced by the K2 algorithm rather than by local search, although the latter was applied in a greedy scheme as well as in combination with simulated annealing. In the experiment depicted in Figure 6.5, the skeleton search strategy found a DAG with a higher score, i.e. posterior probability, than the K2 search strategy did.²

In order to examine the stability of the presented operator when given different sample sizes, we applied the skeleton search strategy to data sets of various sizes, sampled from the probability distribution described by the alarm network [8]. In Figure 6.6, the skeleton search strategy is compared to the K2 search strategy. Given only small samples (with fewer than 2,000 cases), our simple strategy gets stuck at a local optimum which is far from the global one. The prior knowledge about a correct ordering on the variables seems to be beneficial to the K2 procedure when given very small samples. When given samples with at least 2,000 cases, however, our search strategy finds DAGs with a higher score than the K2 search strategy does, although the skeleton search strategy does not use any prior knowledge about the ordering on the variables.

Besides the artificial data sampled from the alarm network, we also used real-world data in our experiments. The three data sets (BOS), (ENV) and (SEW) are described in Appendix B.

²When comparing the different search strategies in our experiments, we used the same scoring function in each of which. Here, the K2 algorithm is thus used with a scoring function which is slightly different from the K2 metric originally employed in [33, 34].

data set	# variables	5-fold cross-validation:	
		skeleton search	local search
(BOS)	14	1.9 ± 0.1	2.1 ± 0.1
(ENV)	11	0.88 ± 0.02	0.92 ± 0.04
(SEW)	5	0.050 ± 0.003	0.051 ± 0.003

Table 6.1: Experiments with three real-world data sets, where the results are assessed by 5-fold cross-validation. The mean and the standard deviation of the Kullback-Leibler divergence is denoted. Small values indicate good learning results.

In these experiments, we chose the Bayesian Information Criterion (BIC) to be the scoring function in structural learning. Given an induced DAG m , the parameters θ_m of the Bayesian network were calculated in a Bayesian way according to Equation 3.1.35. Conjugate priors and a small equivalent sample size $N^l = 5$ were used, like in [81]. We compared the skeleton search strategy to the greedy algorithm based on local search, cf. [81]. The K2 algorithm cannot be applied in this case, as a correct ordering on the variables in the real-world data is unknown. The Bayesian networks determined by the different search strategies are assessed by 5-fold cross validation (see also Appendix A). The results are shown in Table 6.1. Apparently, both search strategies performed very similarly. The skeleton search strategy yields slightly better DAGs than local search does for each of the data sets. Regarding the data sets (ENV) and (BOS), the differences in the two search strategies might be considered notable, because the means appear to be separated to a degree comparable with the value of the standard deviation (cf. Table 6.1). This does not hold for the data set (SEW), where the differences concerning cross validation appear to be negligible. However, the skeleton search strategy found the global optimum with respect to the scoring function BIC in the experiment with this data set,³ whereas greedy local search got stuck at a local optimum.

6.1.5 Discussion

The above operator is similar to the schemes used in the SGS and PC algorithms [142]. However, the presented operator employs a scoring function which assigns a score to each collider candidate. This renders a greedy approach possible. In contrast, the SGS and PC algorithms determine a collider on the basis of the induced conditional independences in a qualitative manner. Moreover, possible inconsistencies regarding the orientations of edges are disregarded by the SGS and PC algorithms, assuming the existence of a perfect map. If inconsistencies occur, the order in which these algorithms proceed through the variables has a crucial impact on the final DAG. Furthermore, the occurrence of directed cycles is not strictly prevented. Both issues are noted in [38]. Thus, the SGS and PC algorithms are only correct if the probability distribution is perfectly known and if there exists a perfect map [142]. When given *finite* data sets,

³Since the data set (SEW) contains only 5 variables, exhaustive search was feasible to determine the global optimum.

the edge-orientation schemes employed by the SGS and PC algorithms are, however, rather unstable [142]. The above operator does not exhibit these weaknesses, and it can be expected to be quite stable, as confirmed in our computer experiments in the previous section.

When evaluating the score, the presented operator accounts for the *parents* of the three variables making up a collider candidate. Consequently, a variable with an *extremely large* number of parents cannot appear in the final DAG, as this is prevented by the term penalizing model complexity, inherent in the scoring function. In contrast, the SGS and PC algorithms do not account for the parents of the collision node when determining colliders. This can lead to a very large number of parents of that variable, giving rise to a Bayesian network with an extremely high model complexity. A consequence is over-fitting, and in the most extreme case the number of parameters can be so large in the Bayesian network that it does not fit into the main memory of a computer.

A shortcoming of the above operator might be its greedy nature, both when determining the colliders in the first step and when resolving the inconsistencies encountered in the subsequent step. Moreover, directed cycles are avoided by a simple heuristics which does not take into account the scoring function at all. This keeps the operator quite simple, but might also cause this scheme to get stuck at a local optimum. Nevertheless, this greedy operator can serve as an efficient way for finding close-to-optimum orientations, as confirmed in our experiments.

The inconsistencies, indicated as " \leftrightarrow " by *ProposeOrientations* (cf. Algorithm 6.1), are related to uncertainty regarding the *orientations* of edges. Hence, graphs like the ones in the Figures 6.3 (4) and 6.4 (3) can provide some additional insight in model uncertainty. This kind of graph is obtained by determining the colliders and applying the procedure *ProposeOrientations*. Moreover, one might be interested in displaying model uncertainty regarding both the presence and the orientations of edges in a single graph, as it was done in the PRONEL project [120]. In this project, given the preliminary summary graph (with the ambiguous regions), the edges were oriented on the basis of the induced conditional independences by an algorithm similar to the edge-orientation schemes in the PC or FCI algorithms [142]. In many of our experiments, however, it appeared that almost every edge was oriented like " \leftrightarrow ". On the one hand, this can be interpreted as a large uncertainty regarding the orientations of edges. On the other hand, such a graph does not necessarily provide notably more information about the orientations than a skeleton or a summary graph do. Hence, a scheme which can resolve the inconsistencies regarding the orientations is essential for inducing the directions in a local optimum, like for instance *FixOrientations*. Uncertainty regarding the orientations might be explored by running the above operator repeatedly, each time starting out from the given skeleton: in each run, the directions of the edges are fixed with a probability depending on the corresponding scores, similar to simulated annealing. According to our experience, the interpretation of the orientations may better be based on the k highest-scoring DAGs than on a graph showing many edges oriented like " \leftrightarrow ".

Orienting the colliders can be viewed as a *local* approximation, which is similar to the approach concerned with the presence of edges in the previous chapters, leading to the necessary path condition. Here, the colliders are determined on the basis of the three variables involved

and of their parents, while the remaining network structure is disregarded. The procedure *ProposeOrientations* accounts for the non-locality of the learning problem, as it tries to orient *all* edges without introducing additional colliders. Naturally, this can lead to inconsistencies (cf. for instance the Figures 6.3 and 6.4), as it is based on the colliders previously oriented by a local approximation. The occurrence of inconsistencies can hence be viewed as a consequence of combining the orientations which have locally been determined in order to achieve a solution regarding the entire graph.

We note that the edges oriented in "both" directions (" \leftrightarrow ") by the procedure *ProposeOrientations* may not be confused with a similar kind of edges, also oriented in both direction. The latter kind of edges plays an important role in causal discovery settings, indicating the presence of hidden variables whose existence can be determined in domains where all variables are continuous with a Gaussian distribution, as described in [142].

6.2 Finding Optimal DAGs

The edges determined by the constraint-based approach, whether employing one of the variants of the necessary path condition or not, are present with a high degree of certainty in optimal DAGs. Conversely, all the edges present in an optimal DAG are not necessarily contained in the parsimonious skeletons. This was discussed in the Sections 4.3.4 and 5.4.3. For this reason, the constraint-based approach is not appropriate for *directly* finding optimal DAGs. Nevertheless, if one aims at finding (locally) optimal DAGs, the edges determined by the constraint-based approach can serve as a basis for other learning algorithms applied subsequently. In principle, such schemes can operate in the space of skeletons or in the space of DAGs. However, only a necessary rather than a sufficient condition for finding optimal DAGs can be applied in the space of skeletons. This requires that the final step has to take place in the space of DAGs (or equivalence classes). For simplicity, we focus on schemes applicable in the space of DAGs, where the DAGs determined by the constraint-based approach serve as the starting graphs. For instance, local search might be carried out. This can be done in a greedy way or combined with simulated annealing, where a relatively small starting temperature might be desirable in order to stay in the vicinity of the initial DAGs during the search process. A very simple search strategy is used in the experiments carried out in the next section. It comprises four successive steps:

- (i) parsimonious skeletons are determined by means of the necessary path condition,
- (ii) each parsimonious skeleton is transformed into a DAG by the edge-orientation operator described above,
- (iii) forward inclusion is carried out in each DAG,
- (iv) backward elimination is performed in each DAG.

Let the first two steps, described in this and the previous chapter, be called the *first extension*. It yields DAGs containing the same edges as the parsimonious skeletons. Since the DAGs induced by the first extension are expected to contain a (slightly) smaller number of edges than the optimal ones, the forward inclusion step is necessary for finding (close to) optimal DAGs. If an edge can be included in either direction with the same improvement in the score, its orientation is chosen randomly. Often, the orientation is, however, determined by the required acyclicity of the resulting graph. Since the orientations assigned in the steps (ii) and (iii) are not altered at a later stage by this scheme, it is crucial that only a few edges are included or eliminated. The last step comprising backward elimination accounts for the edges which have "erroneously" been included due to the heuristics aimed at reducing the number of computed scores (cf. Section 5.4.3). This number can, however, be expected to be rather small, and this step is hence of minor importance.

Let us call the greedy scheme comprising all four steps the *extended algorithm* in the remainder of this chapter. It determines DAGs having some edges added (or removed) compared to the first extension. This simple scheme can be expected to work well if step (i) yields skeletons which are close to optimum.

Since the steps (ii), (iii) and (iv) are carried out for each of the graphs found in step (i), a large number of parsimonious skeletons is prohibitive in this simple scheme. Caching of the computed scores can be used for speeding up computations considerably, as the different skeletons are very similar to each other.

One might be concerned that different skeletons induced in the first step eventually lead to the same DAG after adding and removing edges in the last two steps of the extended algorithm. Indeed, this occurred in our experiments, but only in a small number of graphs (fewer than 10%). Hence, almost every DAG induced by the extended algorithm corresponds to a different local optimum of the scoring function. Among those, the two DAGs with the highest score are considered in the following experiments.

From a Bayesian point of view, one might be interested not only in the different local optima of the posterior probability when used as the scoring function, but also in the "width" of the scoring function at each optimum. Local search might be a suitable search strategy for exploring whether the scoring function exhibits a sharp peak at an optimum, i.e. the neighboring DAGs have a considerably smaller posterior probability than the optimum, or whether the posterior probability of an optimum is only slightly larger than the one of some DAGs in its vicinity.

6.3 Experiments

In the experiments in Chapter 5, the skeletons induced by the constraint-based approach employing the necessary path condition could be compared to the original alarm-network structure in a *qualitative* way only. Now, the operator for orienting the edges, as described at the beginning of this chapter, enables us to determine DAGs. The found DAGs can be compared to each other in a *quantitative* way, for instance by means of the (absolute) scoring function or by cross

validation (cf. Appendix A). For this reason, the learning accuracy of the different approaches can be assessed also when applied to real-world data, besides artificial data sampled from a known Bayesian network.

6.3.1 The Alarm Network

Continuing the experiments of the previous chapter, we compare the DAGs determined by the extended algorithm (cf. Section 6.2) to the one found by the K2 algorithm [33, 34]. The latter algorithm is chosen because of two reasons. First, a correct ordering on the variables is known in this experiment, as required by the K2 algorithm. Second, the K2 algorithm performed extremely well in the alarm-network experiments regarding structural differences [34, 81]. Other approaches like local search, possibly combined with simulated annealing, led to DAGs with a larger structural difference than the one determined by the K2 algorithm [81]. This is not unexpected, since the knowledge about a correct ordering on the variables was only given to the K2 algorithm.

Like before, we assume uniform priors for the DAGs, and use the posterior probability with conjugate priors and the equivalent sample size $N^\dagger = 1$ as the scoring function. In order to compare the different search strategies, this scoring function is also applied to the K2 search strategy [34] (instead of the original K2 metric). Moreover, the threshold value takes the same value as before, namely $\gamma = 3$. In step (i) of the extended algorithm, the least strict si-path is employed by the necessary path condition (cf. Definition 4.16).

In Figure 6.7, the DAG induced by the K2 search strategy is compared to the two highest-scoring DAGs found by the extended algorithm. Apparently, the best DAG induced by the extended algorithm is Markov-equivalent to the original alarm-network structure m_{orig} when given large samples ($\geq 30,000$ cases). Down to samples containing as few as 800 cases, the extended algorithm induces DAGs with a higher posterior probability than assigned to the

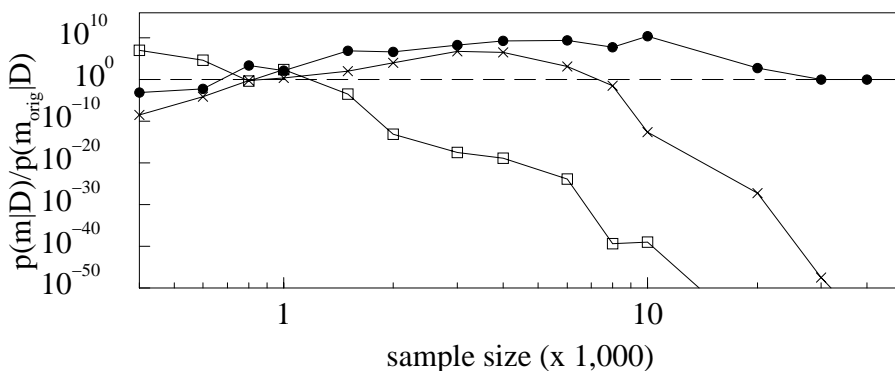


Figure 6.7: The result of the K2 algorithm (□) is compared to the best and second-best DAGs found by the extended algorithm (●, ×). m_{orig} is the original alarm-network structure, and the geometric mean of 5 samples of each size is displayed.

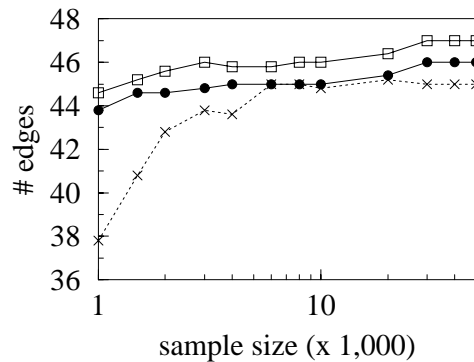


Figure 6.8: The K2 algorithm (\square) tends to include one edge more than the extended algorithm (best DAG, \bullet). In comparison to that, also the number of edges found by the constraint-based approach employing the least strict si-path is shown (\times) (mean of 5 samples of each size).

original DAG m_{orig} .

Regarding the second-best DAG found by the extended algorithm, the posterior probability increases and eventually exceeds the score of the original alarm-network structure, when the sample size decreases from initially large values. This indicates that model uncertainty increases as the sample size diminishes. When the posterior probabilities of the best and the second-best DAGs approach each other, model averaging can lead to an improvement in the predictive accuracy (cf. Section 3.2). However, even at small sample sizes, the difference in the posteriors is still of some orders of magnitude (cf. Figure 6.7). For this reason, model averaging might not lead to a notable improvement in the predictive accuracy in this experiment, because the average is dominated by the best model (cf. also Section 3.2). A similar result was also found in [81]. Since model uncertainty increases further when the sample size drops to very small values, one might expect the posterior probability of the k -best induced DAGs to rise even further. Instead, when given samples with fewer than about 600 cases, the extended algorithm yields DAGs with posterior probabilities smaller than the one of the original DAG. This indicates that, given very small data sets, the extended algorithm might not find networks close to the global optimum. The reason is that, when the sample size is very small, the number of edges determined in step (i) of the extended algorithm is considerably smaller than the optimal number. This was discussed in Section 5.6.2 (cf. Figure 5.19). In this case, simply applying forward inclusion in the subsequent step does not necessarily lead to a solution close to optimum (cf. also Figure 6.7).

The original alarm network contains 46 edges (cf. Figure 5.16). As shown in Figure 6.8, this number is found by the extended algorithm only when given quite large sample sizes ($\geq 30,000$ cases). This number decreases gradually as the sample size diminishes. Over a large range of sample sizes, 45 edges are found to be present in the best DAG, obtaining a larger score than the original network structure. In fact, the best DAG determined by the extended algorithm is Markov-equivalent to the original one, except for the edge $x_{22} \rightarrow x_{15}$ which is favored to be absent by the scoring function (cf. the original DAG in Figure 5.16). This is because

the penalty term regarding model complexity, inherent in the posterior probability, dominates over the improved fit of the Bayesian network model containing the edge $x_{22} \rightarrow x_{15}$. This experiment shows that the original DAG is not necessarily the optimum, since a less complex model is often favored at smaller sample sizes.

In comparison to that, the K2 algorithm yields DAGs with a notably smaller score than the best or second-best DAGs determined by the extended algorithm at all sample sizes except for rather small ones. The reason is that the K2 strategy tends to include an extra edge compared to the original alarm-network structure. This is caused by the greedy nature of the K2 algorithm, as noted in [34]. This additional edge can also be inferred from Figure 6.8, which shows that, over a large range of sample sizes, the K2 algorithm yields one edge more than the extended algorithm. As the sample size decreases, the penalty regarding the additional model complexity (due to the extra edge) reduces so that the posterior probability increases. Eventually, it exceeds the one of the original DAG, showing that the knowledge of the correct ordering on the variables helps the K2 search-strategy greatly at small sample sizes.

The DAG m_{PC} recovered by the PC algorithm (also using the posterior probability as the scoring function, cf. Equation 3.1.34) is assigned the score $\frac{p(m_{PC})}{p(m_{orig})} < 10^{-2000}$, and the best graph m_{first} induced by the first extension gets a score of $\frac{p(m_{first})}{p(m_{orig})} \approx 2.3 \cdot 10^{-66}$ (geometric mean of 5 samples containing 2,000 cases each). This shows that the necessary path condition leads to a considerable improvement in the constraint-based approach. Nevertheless, the induced DAGs are still far from the global optimum, as also the necessary path condition tends to yield graphs with too few edges. A subsequent step of optimization – achieved by including additional edge – is hence necessary.

6.3.2 Real-World Data

Besides the above experiments with artificial data, we also used the three real-world data sets described in Appendix B. In the following experiments, we focus on the posterior probability with conjugate priors serving as the scoring function. A uniform prior for the network structures and the threshold value $\gamma = 0$ are used, because the latter value led to the best results in the following experiments (cf. also Section 5.6.2). The scoring function itself can have quite some influence on the induced graphs, as examined in Section 5.6.2. The equivalent sample size N' is a free parameter in the Dirichlet prior inherent in the employed posterior probability. For this reason, we calibrated the posterior probability by means of a sensitivity analysis. The predictive accuracy of the induced Bayesian networks was assessed by cross validation (cf. Appendix A).

Different variants of the presented learning algorithm are compared to each other and to established schemes:

- the original PC algorithm,
- a modified PC algorithm,
- the first extension (cf. Section 6.2), based on the strictest or on the least strict si-path,
- the extended algorithm (cf. Section 6.2) employing the most strict si-path in the necessary path condition in step (i),
- local search (greedy hill-climbing), starting from the empty graph (cf. also Section 3.3.1),
- local search combined with simulated annealing, beginning at the empty graph (cf. also Section 3.3.1),
- exhaustive search in the space of DAGs if tractable.

The PC algorithm is a typical constraint-based scheme, assuming the existence of a perfect map, and hence not employing a necessary path condition. In the modified PC algorithm, the χ^2 -test is replaced by the posterior probability (cf. Section 3.1.4). Besides that, the presented edge-orientation operator (see Section 6.1) is used instead of the original procedure. While the (modified) PC algorithm determines the certainly-present edges, the benefits of the necessary path condition become apparent in the first extension, yielding ambiguous edges besides the certainly-present ones. We also examined the difference between the least strict si-path and the most rigorous one, i.e. the si-4-path and the si-2-path (cf. the Sections 4.3.3 and 4.3.2). In the extended algorithm, the post-processing step comprising forward inclusion and backward elimination eventually leads to (local) optima. These DAGs are compared to two popular variants of local search, examined in [81]. In domains with only a few variables, we also apply exhaustive search. Although the K2 algorithm typically yields very good results, it is not applicable to the real-world data sets, as a correct ordering on the variables is unknown.

Data on Environmental Influences

The data set on *Environmental Influences on the Condition of Trees* [51] is slightly modified for the use in our experiments, as described in Appendix B. Table 6.2 shows the results of the sensitivity analysis: the Bayesian networks entailed by an equivalent sample size of $N \approx 40$ appear to perform best concerning cross validation. In comparison to that, when the Akaike Information Criterion (*AIC*) is used for structural learning, and when the parameters are subsequently calculated in a Bayesian manner with $N' = 40$ (cf. Equation 3.1.35), cross validation yields 0.68 ± 0.02 for the Kullback-Leibler divergence. This value is slightly worse than the ones entailed by the posterior probability.

The value of $N' = 40$, as suggested by the sensitivity analysis, was used for determining the graphs shown in Figure 6.10, except for the skeleton (A). The latter was recovered by the PC algorithm employing the χ^2 -test with a significance level of $\alpha = 5\%$. Graph (B) was found

equivalent sample size N'	5-fold cross validation:	
	extended algo.	local search (greedy)
10	0.66 ± 0.05	0.67 ± 0.05
20	0.64 ± 0.02	0.64 ± 0.02
40	0.61 ± 0.02	0.62 ± 0.02
80	0.62 ± 0.02	0.64 ± 0.02
160	0.65 ± 0.03	0.66 ± 0.02

Table 6.2: The sensitivity analysis is based on 5-fold cross validation. The mean and the standard deviation of the Kullback-Leibler divergence is denoted.

by the modified PC algorithm, where the χ^2 -test was replaced by the posterior probability. At first glance, it looks counter-intuitive that the χ^2 -test entailed a graph with fewer edges than the posterior probability did, i.e. 11 versus 14 edges. According to the alarm-network experiments in Section 5.6.2, one would have expected the χ^2 -test to yield denser graphs than the posterior probability does. This is also suggested by the fact that the χ^2 -test involves a larger term penalizing model complexity than the posterior probability does. A thorough examination reveals, however, that this unexpected result is caused by the heuristics which focuses on the neighbors of a variable in the intermediate graph, reducing the number of computed scores (cf. Section 5.4.3). This is depicted in Figure 6.9. The large penalty term inherent in the posterior probability entails a large number of edges being removed at low orders. Consequently, this heuristics forbids the evaluation of many scores of high orders. In particular, graph (B) prevents this heuristics from computing scores beyond third order, since each variable has at most four neighbors. In contrast, the small penalty term in the χ^2 -test does not lead to the removal of a large number of edges at low orders, leaving the graph quite dense. Thus, the heuristics allows the computation of many scores of high orders. In tests of high orders, however, the term penalizing model complexity often dominates over the term accounting for model fit. This leads to the elimination of a large number of edges due to tests of high orders in this experiment (cf. Figure 6.9). As a result, when using the χ^2 -test, the overall number of edges removed from the graph is quite large in this experiment.

When the least strict si-path is used in the necessary path condition, no ambiguous edges are found in this experiment, and hence the same graph is obtained as found by the modified PC algorithm, namely graph (B) in Figure 6.10. In contrast, the strictest si-path entails many ambiguous edges (cf. graph (C) in Figure 6.10). Since this graph is denser due to the ambiguous edges, scores of higher orders are computed, similarly to the previous case concerning the χ^2 -test. Consequently, several edges which are certainly present in graph (B) are altered to ambiguous edges in graph (C). Because a single and extremely large preliminary ambiguous region (comprising 30 edges) is yielded by the strictest si-path, exhaustive search for the parsimonious structures was intractable so that approximate search had to be applied (cf. Section 5.3.2). For this reason, some parsimonious skeletons may not be found. In fact, the approximate search yields all the ambiguous edges in graph (C) to be definitely present, while all the other ambiguous edges are determined to be definitely absent. Consequently, the summary

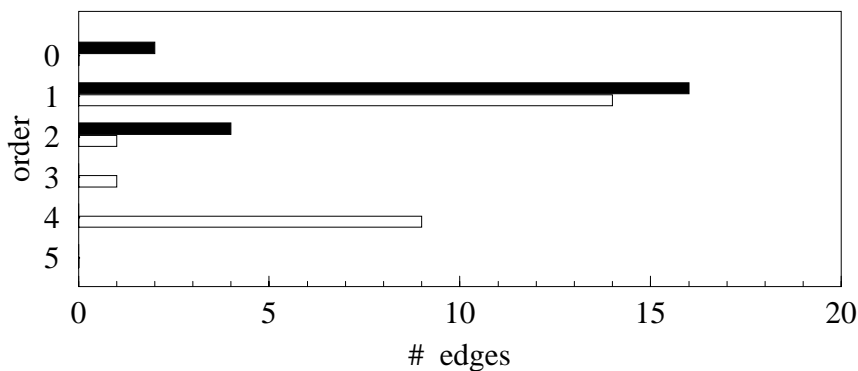


Figure 6.9: The number of edges removed by the PC algorithm due to negative scores of different orders: (black) the posterior probability with $N = 40$ is used, and (white) the χ^2 -test with a significance level of $\alpha = 5\%$ is applied.

graph (C) displays a single skeleton containing 17 edges. After the edges have been oriented by the operator proposed above, the posterior probability of the DAG resulting from graph (C) is larger than the one of the DAG obtained from graph (B) (cf. Table 6.3). This is because the strictest si-path yielded 3 additional edges compared to the least strict si-path. The benefits of employing the strictest necessary path condition are also reflected by the results regarding cross validation in Table 6.3.

All the edges determined by the constraint-based approaches (cf. the graphs (A) through (C) in Figure 6.10) are also present in the graphs (F), (G) and (H), which are (local) optima found by different search strategies. This confirms the claim in Section 4.3.4, saying that the edges induced by the constraint-based approach are present in the optimal DAG with a high degree of certainty.

The DAG (D) in Figure 6.10 is found by the first extension, i.e. by applying the proposed edge-orientation operator to graph (C). Starting out from (D), graph (E) is obtained as an intermediate DAG during the process of forward inclusion, the third step of the extended algorithm (cf. Section 6.2). The inclusion of the edge $x_6 \leftarrow x_7$ is favored by the relative score $g(x_6, x_7, \{x_4, x_5, x_9\}) \approx 501.4$, and subsequently the edge between x_6 and x_8 is added due to the scores $g(x_6, x_8, \{x_4, x_5, x_7, x_9\}) = g(x_8, x_6, \{x_4, x_5, x_7, x_9\}) \approx 142.5$. The orientation of the latter edge is chosen at random, since either direction leads to Markov-equivalent DAGs.

Local search combined with simulated annealing yields graph (H), which contains the edges $x_2 \leftarrow x_8, x_2 \leftarrow x_9, x_3 \leftarrow x_8$ and $x_3 \leftarrow x_9$ in addition to the ones in the intermediate graph (E). If the latter four edges are ignored in graph (H), the graphs (E) and (H) are Markov-equivalent, since the orientations of the edges $x_5 \sim x_6, x_5 \sim x_8$, and $x_6 \sim x_8$ can be reversed (cf. Section 2.2.3).

When the greedy forward-inclusion procedure is continued beyond graph (E), the edges $x_2 \leftarrow x_5$ and $x_3 \leftarrow x_5$ are added due to the scores $g(x_2, x_5, \{x_1, x_3\}) \approx 3.8$ and $g(x_3, x_5, \{x_1\}) \approx 13.7$, respectively. This is shown in graph (F), found by the extended

algorithm	entire data set:			5-fold cross validation
	# edges	# param.	$\frac{p(m D)}{p(m_o D)}$	
extended algorithm: DAG m_o	21	4,982	1.0	0.61 ± 0.02
1 st extension, strictest si-path	17	1,852	$5.9 \cdot 10^{-288}$	0.76 ± 0.08
modified PC algo.	14	1,630	$1.3 \cdot 10^{-383}$	1.00 ± 0.06
local search (greedy)	20	4,898	$1.5 \cdot 10^{-19}$	0.62 ± 0.02
local search (sim. annealing)	23	5,342	$3.5 \cdot 10^9$	0.62 ± 0.01

Table 6.3: Comparison of various learning algorithms: the number of edges, the model complexity (number of independent parameters) and the posterior probability ($N = 40$) of the Bayesian networks induced from the entire data set. Furthermore, the results of 5-fold cross validation are shown, namely the mean and the standard deviation of the Kullback-Leibler divergence.

algorithm, since the backward elimination step did not remove an edge. The difference in the scores of the graphs (F) and (E) is hence given by $3.8 + 13.7 = 17.5$. When edges are added to graph (E), the inclusion of each edge additionally present in DAG (H) is prohibited by the negative scores $g(x_2, x_8, \{x_1, x_3\}) \approx -6.7$, $g(x_2, x_9, \{x_1, x_3\}) \approx -10.1$, $g(x_3, x_8, \{x_1\}) \approx -0.2$ and $g(x_3, x_9, \{x_1\}) \approx -5.2$. However, when *both* the edges $x_2 \leftarrow x_8$ and $x_2 \leftarrow x_9$ are included, the absolute score is increased by $g(x_2, x_8, \{x_1, x_3\}) + g(x_2, x_9, \{x_1, x_3, x_8\}) = g(x_2, x_9, \{x_1, x_3\}) + g(x_2, x_8, \{x_1, x_3, x_9\}) \approx 8.8$. Adding *both* $x_3 \leftarrow x_8$ and $x_3 \leftarrow x_9$ leads to an improvement of $g(x_3, x_8, \{x_1\}) + g(x_3, x_9, \{x_1, x_8\}) = g(x_3, x_9, \{x_1\}) + g(x_3, x_8, \{x_1, x_9\}) \approx 30.7$. Hence, simulated annealing leads to a graph whose score is larger than the one obtained by the extended algorithm. The difference is $8.8 + 30.7 - 17.5 = 22$, which translates into the ratio of the posterior probabilities $p(m_{(H)} | D) / p(m_{(F)} | D) \approx e^{22} \approx 3.5 \cdot 10^9$, as denoted in Table 6.3. The difference between (F) and (H) is apparently a consequence of the greedy nature of the forward inclusion step used in the simple extended algorithm. Since only one edge is added at a time, the forward inclusion procedure would be required to accept a decrease in the absolute score before the additional improvement could be achieved. This is a typical situation showing the limitations of greedy schemes. Hence, there is a lot of space for improvements in the post-processing step of the extended algorithm, replacing the forward inclusion and backward elimination procedures by more sophisticated ones. Nevertheless, the extended algorithm finds the highest-scoring DAGs among all the *greedy* algorithms examined in these experiments, including greedy local search (cf. Table 6.3). The graph found by the latter is depicted in Figure 6.10 (G). It contains one edge fewer than graph (F), namely $x_2 \leftarrow x_5$, since a directed cycle via the variables x_1 and x_6 has to be avoided.

Concerning the results of cross validation (cf. Table 6.3), all the algorithms yielding (local) optima, namely the extended algorithm and the two local search strategies, perform about equally well. In this experiment, the number of independent parameters of the various Bayesian network models increases with the number of edges.

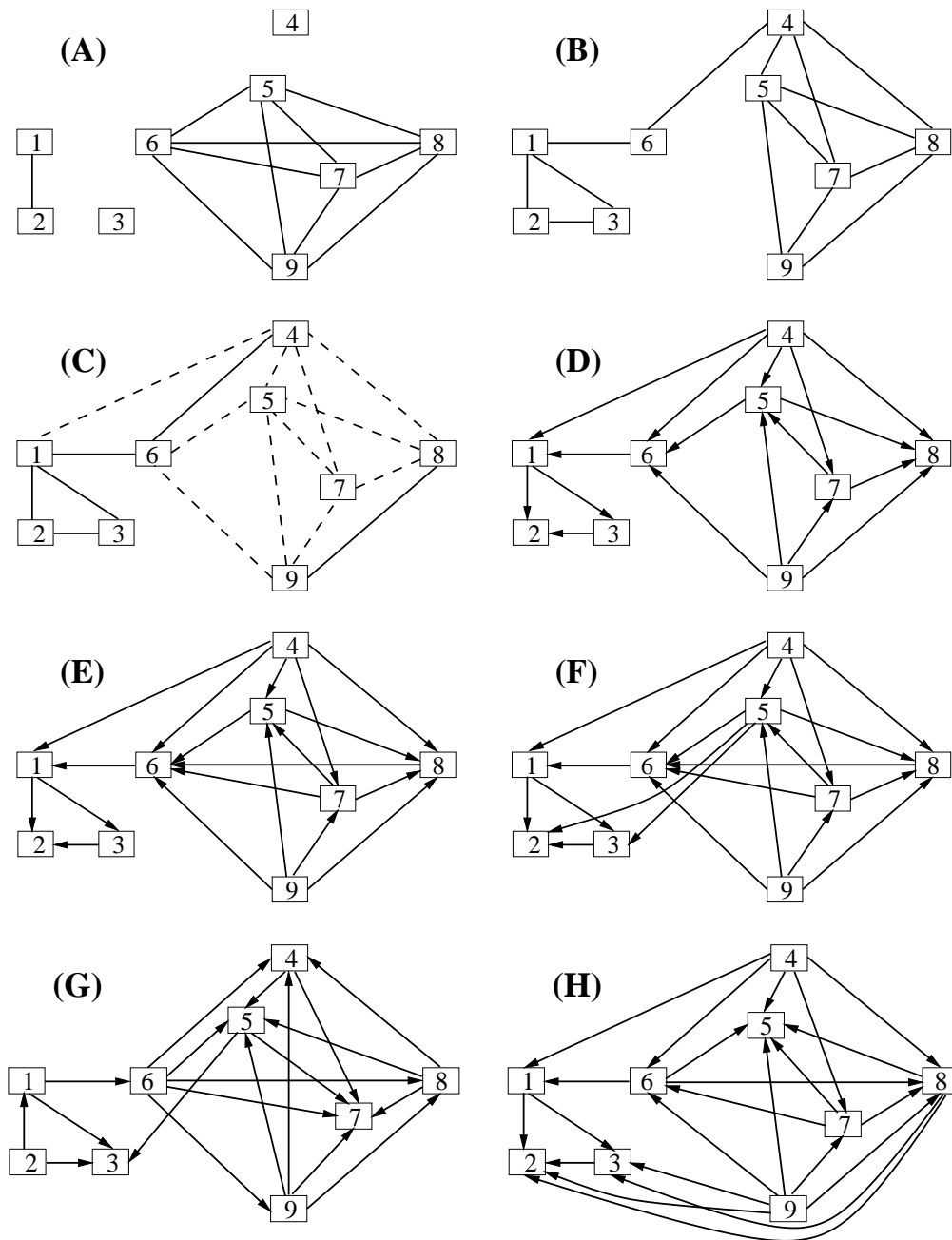


Figure 6.10: The graphs determined by various search strategies from the data on *Environmental influences on the Condition of trees* [51]: (A) PC algorithm with the significance level $\alpha = 5\%$ used in the χ^2 -tests, (B) modified PC algorithm employing the posterior probability with $N' = 40$ (like in the other learning algorithms), (C) necessary path condition employing the strictest si-path, (D) first extension, (E) intermediate graph during the forward inclusion step of the extended algorithm, (F) final graph induced by the extended algorithm, (G) local search (greedy), (H) local search combined with simulated annealing (the starting temperature $T_0 = 3.0$ yielded better results than $T_0 = 10$ or 100 in this experiment, cooling rate 0.95 , final temperature $T_f = 0.2$, cf. also Section 3.3.1).

The induced graphs represent the close associations among the variables x_1 , x_2 and x_3 , as they refer to needle loss (cf. Table B.2). While the orientations of the edges $x_1 \rightarrow x_2$ and $x_1 \rightarrow x_3$ are uniquely determined, the edge $x_2 \sim x_3$ can be oriented in either direction in equivalent DAGs, except for DAG (G). In most of the induced graphs, the variables referring to needle loss do not have any other descendents. This might be expected according to a causal interpretation of the induced graphs. However, such an interpretation may not be appropriate in general, as the presence and the orientations of some other edges cannot be understood in a causal setting, see e.g. the edge between x_7 (soil) and x_8 (wind direction). It is hence more reasonable to view the induced Bayesian network simply as a statistical model. The variables x_6 (use of forest) and x_4 (density of trees in forest) are parents of x_1 (needle loss) in most of the graphs. In some graphs, also the variables x_5 (composition of forest), x_8 (wind direction) and x_9 (elevation) are adjacent to the variables describing needle loss. A further analysis of the graphs is left to the reader.

Boston Housing Data

The same kind of experiments as before were also carried out for the *Boston Housing Data* [12], which was discretized for simplicity (cf. Appendix B). The sensitivity analysis depicted in Table 6.4 yields the best results with respect to cross validation for the equivalent sample sizes $N' \approx 10 \dots 20$. We used the value $N' = 10$ in the following experiments. In comparison to that, when the *AIC* is employed as the scoring function in structural learning and when the parameters are estimated in a Bayesian manner (cf. Equation 3.1.35, with $N' = 10$) then 5-fold cross validation yields a Kullback-Leibler divergence of 2.0 ± 0.2 . This value is about the same as the one obtained by means of the posterior probability (cf. Table 6.4). The performance of the various learning algorithms on this data set is shown in Table 6.5. Regarding the posterior probability as well as cross validation, the necessary path condition – whether based on the least strict si-path or the strictest one – leads to notably better graphs than the modified PC algorithm does. This shows the benefits of using the necessary path condition. The summary graph entailed by the least strict si-path is depicted in Figure 6.11. It contains ambiguous edges, indicating some model uncertainty. This might not be unexpected because of the rather small sample containing 506 cases only. The ambiguous edges can be grouped into five ambiguous

equivalent sample size N'	5-fold cross validation:	
	extended algo.	local search (greedy)
5	1.9 ± 0.2	1.9 ± 0.2
10	1.8 ± 0.1	1.9 ± 0.1
20	1.8 ± 0.1	1.9 ± 0.2
40	1.9 ± 0.1	1.9 ± 0.2
80	1.9 ± 0.1	2.0 ± 0.2

Table 6.4: The sensitivity analysis is again based on cross validation (mean and standard deviation of the Kullback-Leibler divergence).

algorithm	entire data set:			5-fold cross val.
	# edges	# param.	$\frac{p(m D)}{p(m_o D)}$	
extended algorithm: best DAG m_o	43	197	1.0	1.8 ± 0.1
extended algorithm: 2 nd -best DAG	44	207	0.16	1.9 ± 0.3
1 st ext., strictest si-path: best DAG	27	77	$4.1 \cdot 10^{-55}$	2.2 ± 0.1
1 st ext., least strict si-path: best DAG	24	71	$1.7 \cdot 10^{-95}$	2.3 ± 0.1
modified PC algorithm	17	40	$6.5 \cdot 10^{-151}$	2.7 ± 0.2
local search (greedy)	42	233	$2.0 \cdot 10^{-14}$	1.9 ± 0.1
local search (simulated annealing)	43	197	346	1.9 ± 0.2

Table 6.5: This table sketches the properties of the Bayesian networks induced from the discretized Boston housing data by the various learning algorithms, all employing the posterior probability with $N^l = 10$ as the scoring function.

regions, whose parsimonious structures are shown in Table 6.6. The least strict si-path yields $3 \cdot 2^3 = 24$ different parsimonious skeletons, and each of which contains 7 ambiguous edges, besides the 17 edges being certainly present. The latter are also found by the modified PC algorithm. The strictest si-path yields additional ambiguous edges (cf. Table 6.5), entailing a further improvement in the posterior probability as well as in cross validation. The induced summary graph is, however, so dense that it is hard to display in a clearly arranged way. For this reason, this graph is not shown here, like the other DAGs containing more than 40 edges.

The extended algorithm as well as local search yield rather dense graphs, while the skeletons found by the constraint-based algorithms contain considerably fewer edges. This situation had also occurred in the alarm-network experiments in Section 5.6.2 when very small data sets were given. This is hence a typical property of constraint-based algorithm when applied to rather small data sets. In the tiny Boston housing data set, a large number of edges has thus to be added in the post-processing step of the extended algorithm in order to arrive at a local optimum. The backward elimination step removes 2 edges previously included by the first extension. This indicates that, at rather small sample sizes, the number of "erroneously" present edges increases. This occurred also in the alarm-network experiments in Section 5.6.2. When the sample size is very small, the edges induced to be present by the constraint-based approach are thus present in the optimal DAG with a decreased degree of certainty. This is not necessarily a shortcoming of the constraint-based approach in general, but rather a consequence of the heuristics aimed at reducing the number of computed scores (cf. Section 5.4.3).

The posterior probabilities of the best and the second-best DAGs found by the extended algorithm are quite close to each other (cf. Table 6.5). This might be understood as an additional indication for quite large model uncertainty present in the small data set. The best DAG found by the extended algorithm is very similar to the one found by local search combined with simulated annealing (starting temperature $T_0 = 10$, cooling rate 0.95, final temperature $T_f = 0.2$, cf. also Section 3.3.1): both Bayesian networks have the same number of edges as well as the same complexity (cf. Table 6.5). Regarding the posterior probability, simulated annealing leads to a

two approaches, the variable x_1 (median value of homes) was used as the dependent variable, and the others as independent ones. In contrast, learning Bayesian networks aims at fitting a model to the *joint* probability distribution for all the variables, i.e. variable x_1 is not distinct from the others. This is typical for unsupervised learning, whereas the former two approaches correspond to supervised learning. Hence, the results of our experiments are hard to compare with the ones in [17, 76]. The regression tree grown in [9] exhibits only four nodes which appear in splits, namely x_2 , x_7 , x_9 and x_{14} . While the former three variables are also adjacent to x_1 in each of the parsimonious skeletons represented by the summary graph in Figure 6.11, variable x_{14} is adjacent to x_1 neither in Figure 6.11 nor in the denser DAGs induced by the extended algorithm or local search (greedy and simulated annealing). A detailed comparison is impossible also because we discretized the variables in the Boston housing data for the use in our experiments. This was done for simplicity, since the main purpose of the experiments in this thesis is to compare the performance of the different structural learning algorithms rather than to examine various ways for modeling non-linear dependences among continuous variables, as done in [60, 85, 86, 113]. The publications [85, 86] also report results on the Boston housing data. The reported DAGs are quite sparse due to a large value of 25 penalizing the presence of each edge (cf. also Table 5.5), because the aim was to ease the interpretation by means of a sparse graph rather than to optimize the predictive accuracy of the induced Bayesian networks, for instance, assessed by cross validation.

Data gathered by Sewell and Shah

The data set gathered by Sewell and Shah [135] (cf. also Appendix B) contains only 5 variables, causing the search space of DAGs to be reasonably small (29, 281 DAGs). Exhaustive search in the space of DAGs is thus tractable, allowing the computation of the posterior probability for all the DAGs. Consequently, the global optimum can be determined. Exhaustive search is used in the sensitivity analysis depicted in Table 6.7, where a value of $N' \approx 200$ leads to the best results with respect to cross validation. In comparison to that, if the Akaike Information Criterion is used as the scoring function and when the parameters are estimated in a Bayesian way with

equivalent sample size N'	5-fold cross validation: exhaustive search
5	0.045 ± 0.006
50	0.044 ± 0.007
100	0.040 ± 0.006
200	0.040 ± 0.006
300	0.040 ± 0.006
500	0.042 ± 0.005
1,000	0.047 ± 0.005

Table 6.7: This sensitivity analysis suggests a value of $N' \approx 200$. The mean and the standard deviation of the Kullback-Leibler divergence are shown.

algorithm	entire data set:			5-fold cross validation
	# edges	# param.	$\frac{p(m D)}{p(m_o D)}$	
extended algorithm: m_o	7	68	1.0	0.040 ± 0.006
1 st extension, strictest si-path	7	68	1.0	0.040 ± 0.006
modified PC algorithm	7	68	1.0	0.040 ± 0.006
exhaustive search: best	7	68	1.0	0.040 ± 0.006
exhaustive search: 2 nd -best	8	73	0.044	0.040 ± 0.006
local search (greedy)	8	71	0.034	0.040 ± 0.006
local search (sim. annealing)	7	68	1.0	0.040 ± 0.006

Table 6.8: The posterior probability with $N' = 200$ serves as the scoring function in the various learning algorithms. The necessary path condition does not yield any ambiguous edges.

$N' = 200$ (cf. Equation 3.1.35), cross validation yields a Kullback-Leibler-divergence of 0.040 ± 0.007 . This compares to the best values obtained when the posterior probability serves as the scoring function.

When the posterior probability with the equivalent sample size $N' = 200$ is used for structural learning, the DAGs with the two largest scores have posterior probabilities of about 29.9% and 1.3%, respectively (cf. Figure 6.12). Since $x_3 \sim x_5$ is a covered edge in the optimal DAG (see also Section 2.2.3), there is exactly one more DAG equivalent to the optimal one. Concerning the second-best DAG depicted in Figure 6.12, there are two additional Markov-equivalent DAGs, differing from the shown one in the orientation of either one of the edges $x_1 \sim x_5$ and $x_2 \sim x_5$.

Table 6.8 shows that the global optimum is found by exhaustive search, simulated annealing as well as by each of the constraint-based algorithms. No ambiguous edge is yielded by the necessary path condition. The induced skeleton – containing solely certainly-present edges – comprises all the edges present in the optimal DAG. Given this skeleton, the optimal orientations are found by the edge-orientation operator described in Section 6.1. In contrast, the greedy scheme employing local search got stuck at a local optimum. It might not be unexpected that the optimal DAG is induced by most of the learning algorithms under consideration, as the

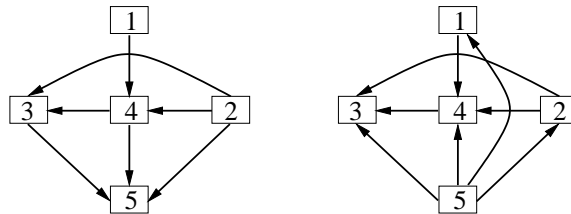


Figure 6.12: The best DAG (left) and the second-best DAG (right) with respect to the posterior probability with $N' = 200$ (determined by exhaustive search).

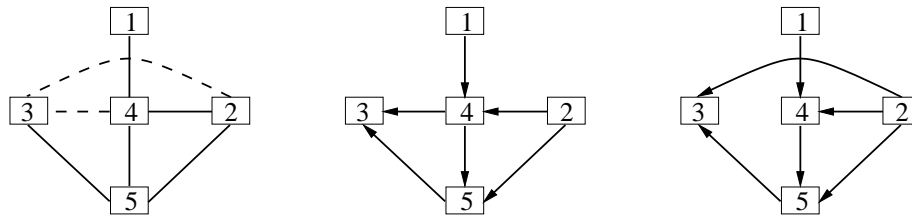


Figure 6.13: When $N^l = 5$ is chosen in the posterior probability, the least strict si-path used in the necessary path condition yields the summary graph (left), where either one of the ambiguous edges is present in the parsimonious skeletons. The two DAGs on the right are obtained by applying the proposed edge-orientation operator to the induced skeletons. As verified by exhaustive search, the left DAG is the global optimum, and there is no other equivalent one. The left DAG has a posterior probability of about 24% and the right one of about $10^{-16}\%$.

sample size is quite large (10,318 cases), while the number of variables is small (5 variables). This is similar to the results in the alarm-network experiments (cf. the Sections 5.6.2 and 6.3.1).

The data set gathered by Sewell and Shah was also analyzed by means of Bayesian networks in [79], where an equivalent sample size of $N^l = 5$ was chosen in the posterior probability. Additionally, constraints regarding the orientations of certain edges were applied, namely it was forbidden that edges pointed towards x_1 or x_2 as well as that edges were oriented away from x_5 . These constraints were motivated by a causal interpretation of the edges. The best DAG complying with these constraints was found to be identical with the left DAG in Figure 6.12. Note that this DAG coincides with the global optimum in the space of all DAGs, i.e. without constraints, when the equivalent sample size $N^l = 200$ is used.

However, when the value $N^l = 5$ is chosen, and when no constraints are imposed on the network structure, the latter DAG has a posterior probability of about $2.4 \cdot 10^{-27}$, and the global optimum in the space of *all* DAGs, i.e. without constraints, is assigned a posterior probability of about 24%, shown in Figure 6.13. The latter graph is not only found by exhaustive search, but also by our extended algorithm: the summary graph depicted in Figure 6.13 is yielded by the necessary path condition employing the least strict si-path. In the parsimonious skeletons, either one of the two ambiguous edges is present. The two DAGs in Figure 6.13 are obtained by applying the presented edge-orientation operator to the found skeletons (cf. Section 6.1).

The above constraints are obviously necessary – given the choice $N^l = 5$ – in order to induce a DAG which can be interpreted in a causal manner. However, the same DAG can also be found without using constraints, namely when a value of $N^l \approx 200$ is chosen, as suggested by the sensitivity analysis in Table 6.7. Of course, without applying constraints the edge $x_3 \sim x_5$ can be oriented in either direction in equivalent DAGs (cf. Figure 6.12), and a causal interpretation of its orientation is hence impossible. Let us note that a causal interpretation of the best DAG suggested by the sensitivity analysis may not be very reasonable in general, as it is merely a coincidence concerning the data set gathered by Sewell and Shah.

6.3.3 Summary

The constraint-based approach employing the necessary path condition and the edge-orientation operator was compared to popular learning algorithms in our experiments. In the alarm-network experiments, over a large range of sample sizes, the extended algorithm (cf. Section 6.2) induced Bayesian networks with a higher posterior probability than the K2 algorithm did, although a correct ordering on the variables was given to the latter as additional input. Apart from that, when given large data sets, the extended algorithm induced a DAG equivalent to the original alarm-network structure. Furthermore, over a large range of "medium"-sized samples, a larger posterior probability was assigned to the best DAG found by the extended algorithm than to the original network structure. The limitations of the proposed algorithm became apparent at small data sets, where the constraint-based scheme, although employing a necessary path condition, yielded considerably fewer edges being present than there are in the optimum.

Similar results were found in the experiments with real-world data. Again, the benefits of extending the constraint-based approach by the necessary path condition were obvious. Moreover, the strictest si-path led to slightly higher-scoring graphs than the least strict si-path, because the former led to skeletons with slightly more edges. Only in the data set (SEW), the global optimum could be induced by all the variants of si-paths employed in the necessary path condition.

The sample size had an impact on the number of edges yielded by the necessary path condition, both in the alarm-network experiments and in the experiments with real-world data, where the three data sets (SEW), (ENV) and (BOS) (cf. Appendix B) contained 10, 318, 6, 168 and 506 cases, respectively. Also the number of ambiguous edges, indicating model uncertainty up to a degree, was found to depend on the sample size in the various experiments. In all the experiments, the extended algorithm yielded DAGs with a larger posterior probability than the graphs found by greedy local search. Local search combined with simulated annealing led to DAGs with higher posteriors when given the data sets (BOS) and (ENV), which showed the limitations of the post-processing step of the extended algorithm due to its greedy nature. Regarding cross validation, the difference between the extended algorithm and local search (greedy or combined with simulated annealing) was rather negligible.

Conclusions

In the recent years, two main approaches to structural learning in Bayesian networks have evolved. The one is aimed at optimizing a scoring function by means of a heuristic search strategy, while the constraint-based approach relies on the conditional independences and dependences derived from the data. Since structural learning in Bayesian networks is an NP-hard problem [14, 27, 84], there is demand for efficient heuristics. While the constraint-based approach has proven to be very efficient in many experiments [23, 24, 142], it is only understood under certain assumptions, which can be expected to hold when an infinite amount of data is given. For that reason, this thesis was concerned with understanding and improving the constraint-based approach for those cases where only *finite* data sets are given, as typical in practical applications.

Given a finite amount of data, we showed the benefits of viewing the constraint-based approach as a particular search strategy aimed at optimizing a scoring function initially specified. While the optimality of a directed acyclic graph (DAG) is well-defined with respect to a scoring function, a perfect map of the probability distribution implied by a finite data set typically does not exist. The latter is, however, assumed by established constraint-based approaches.

We used the property of decomposability of scoring functions, which holds when given complete data and in the absence of hidden variables, to introduce *relative* scoring functions and to derive important interrelations among different relative scores. A main advantage of relative scoring functions is their applicability both to the constraint-based approach as well as to the algorithms aimed at optimizing a scoring function. For this reason, relative scoring functions allow one to understand the relationship between these two approaches.

One important insight is that the graphs recovered by the constraint-based approach tend to contain fewer edges than optimal DAGs. More precisely, an edge found to be present by this approach is also present in a local optimum. Only the heuristics aimed at reducing the number of computed scores can entail a few edges to be present, although those edges are actually absent in optimal DAGs. However, theoretical considerations as well as our computer experiments show that the latter effect is rather negligible in many applications. Hence, the edges determined by the constraint-based approach are present in (locally) optimal Bayesian network structures with a high degree of certainty. Conversely, an optimal DAG might contain additional edges besides the ones found by the constraint-based approach. Thus, if one aims at finding local optima, a post-processing step is necessary. A greedy scheme for this task was presented.

Moreover, we have proposed various extensions of the constraint-based approach itself, which led to considerable improvements in the induced graphs when given finite sample sizes. The main part of this thesis was devoted to the first step of the constraint-based approach, concerned with learning the presence of edges. While state-of-the-art constraint-based approaches look at each edge more or less independently of the others, we propose to take into account a certain vicinity of each edge when deciding about its absence or presence. This can be viewed as an approximation to the exact learning problem, where the entity of a Bayesian network structure is divided into several vicinities. This can be achieved by employing the *necessary path condition* we derived from properties of optimal DAGs. This condition allows an edge to be absent only if certain other edges and some sort of paths are present in its vicinity. For instance, the simplest case involves two edges where the absence of the one requires the presence of the other edge, and vice versa. This means that the necessary path condition yields interdependencies regarding the presence of various edges, i.e. it determines alternative structures, possibly of various sizes. This provides considerably more insight than merely knowing about the presence of each edge independently of all the others. For computational efficiency, the derived condition applies to undirected rather than directed graphs. This implies that it can only be a *necessary* condition, i.e. the edges required to be present are necessary for the resulting DAG to be optimal, but an optimal DAG might contain additional edges. The benefits of employing the necessary path condition were illustrated in various examples and experiments in this thesis.

Compared to state-of-the-art constraint-based approaches, the utilization of a necessary path condition can reduce the number of the edges "erroneously" absent from the induced graphs. This is because the necessary path condition can prevent from removing an edge, even when a corresponding conditional independence has been induced. In other words, all the induced independences do not necessarily entail the removal of the corresponding edges. This is a main difference to state-of-the-art constraint-based approaches, not employing any necessary condition of this kind.

Furthermore, model uncertainty regarding the presence of edges or certain structures in the graph can be discovered by means of the necessary path condition. Of course, this is only possible up to a certain degree. An exact treatment of model uncertainty would be NP-hard, anyway. Model uncertainty discovered by an algorithm employing the necessary path condition typically does not involve the entire graph. In fact, the induced graphs often have a large number of edges in common, and they differ only concerning a few edges, called ambiguous edges. Particularly in Bayesian networks with a large number of variables, the ambiguous edges can usually be partitioned into different ambiguous regions such that the presence of the edges in different ambiguous regions is independent of each other. This independence renders an efficient computation of the different graphs possible, as the latter can be obtained by combining the local results determined in each ambiguous region separately.

Apart from that, the discovered uncertainty can be displayed to the user in a single graph [144], called summary graph. Unlike a list of the different structures, this kind of visualization can considerably ease the interpretation of multiple solutions. Together with the necessary path condition, this has entered the PRONEL project [120]. The utilization of a single graph was later also adopted in [85], where uncertainty regarding the presence of certain structures was

calculated by standard Bayesian model-averaging.

We have discussed four variants of the necessary path condition, differing in their strictness, i.e. the number of edges and the sorts of paths required to be present in the vicinity of an absent edge. In our experiments, all four variants led to considerable improvements compared to established constraint-based schemes, which do not employ a necessary condition. Compared to this enhancement, the four variants yielded very similar results. In principle, it is, of course, possible to improve upon the presented necessary path conditions, e.g. by considering vicinities of increased sizes. The above results suggest, however, that the quality of the induced graphs can be improved only slightly by stricter conditions, while the computational costs may grow considerably.

In the proposed algorithm, the necessary path condition is represented in terms of a set of rules, rendering very efficient simplifications possible in order to find the various network structures. The computational effort additionally required by this extension is rather negligible compared to the, short, computation time consumed by other constraint-based approaches, as computing the conditional independences is most tedious. Moreover, we pointed out that the presented interdependencies among relative scores can be exploited to speed up computations additionally. We found the constraint-based approach also very appropriate for parallel computing, in particular for a master-and-slave scheme.

The second part of the algorithm is concerned with finding the orientations of edges, after the skeletons have been determined in the first part. Like uncertainty regarding the presence of edges or structures could occur in the first step of the constraint-based approach, uncertainty concerning the orientations of edges might arise in the second part. In our experiments, we found that the latter kind of uncertainty affects much more edges than the former does. This is also clear from theoretical considerations, since the orientation of an edge can often depend strongly on other edges far way in the graph, e.g. due to the acyclicity of the Bayesian network structure. For this reason, typical edge-orientation schemes, relying on induced conditional independences and dependences, are quite unstable, even when given rather large data sets. We thus proposed an operator which, given an undirected graph, is aimed at inducing the optimal orientations with respect to a scoring function. For simplicity, this is done in a greedy way, which is rendered possible due to the utilization of a scoring function. Uncertainty regarding the orientations of edges is hence ignored. This algorithm has proven to be quite robust in our computer experiments. Furthermore, while established constraint-based schemes can yield Bayesian network structures where a variable has a very large number of parents, this is prevented by the presented operator, employing a scoring function.

In our computer experiments, the benefits of the proposed extensions of the constraint-based approach became apparent. When given real-world data, the presented extensions of the constraint-based approach combined with a greedy post-processing step induced higher-scoring Bayesian network structures than other greedy search strategies. Only local search combined with simulated annealing, which is a quite time-consuming stochastic search strategy, induced slightly higher-scoring DAGs. We also carried out various experiments with the alarm network, a popular benchmark for structural learning in Bayesian networks. The best

DAG induced by the presented extensions was equivalent to the original network structure when given large samples. Moreover, given medium-sized data sets, a DAG with a score higher than the one of the original network structure was found. To our knowledge, this has not been reported about any other learning algorithm up to now. In particular, the proposed extensions were capable of finding higher-scoring Bayesian network structures than the K2 algorithm [33, 34] did, except for rather small samples. Since the latter algorithm requires a correct ordering on the variables as additional input, it typically finds graphs very close to the original one [33, 34]. Considering structural differences, it performed even better than local search combined with simulated annealing in the experiments reported in [81].

A particular property of the constraint-based approach is that it can be divided into several successive steps, rendering improvements in many details possible, both to advance the quality of the induced graphs beyond the progress achieved in this thesis as well as to further increase the computational efficiency. Moreover, when the constraint-based approach is understood as a strategy aimed at optimizing a scoring function, it may be combined with various other learning strategies. Of course, the presented improvements concerning the constraint-based approach to structural learning in Bayesian networks may also be transferred to other graphical models.

With the advent of the World Wide Web as well as with the progress made in the area of biotechnology a lot of data has become available, waiting to be analyzed by automated algorithms, as a manual analysis is often intractable. We are looking forward to seeing data mining algorithms become more important in the future and provide better insight into the associations important among the variables in a domain. Moreover, these algorithms can also serve as a basis for automatically building intelligent systems for reasoning under uncertainty. We believe that structural learning in Bayesian networks is a promising tool in the fields where uncertainty plays a prominent role. It has thus to be of major practical concern to have efficient learning algorithms at hand, applicable in large domains.

A

Cross Validation

Cross validation can be used for assessing the predictive accuracy of a learned model. It is crucial to distinguish between training data and test data in this procedure. After a model has been learned from the training data, its predictive accuracy is assessed on the basis of the test data. Since the latter samples are not used when learning the model, over-fitting is penalized so that complex models do not necessarily perform better than simple ones.

The predictive accuracy of the model can be assessed by comparing the joint probability distribution q described by the learned model with the one implied by the test data, denoted as p . This can be quantified, for instance, by means of the Kullback-Leibler divergence, or cross entropy [98],

$$KL(p, q) = \sum_{i \in \mathcal{I}(\mathcal{V})} p(i) \log \frac{p(i)}{q(i)}, \quad (\text{A.0.1})$$

where the sum ranges over all joint states $i \in \mathcal{I}(\mathcal{V})$ of the variables in the set \mathcal{V} . When the probability distribution p is determined by the cell counts implied by the test data, the sum extends only over the configurations actually present in the data. As most of the joint states typically do not occur in the test data, this entails a tremendous reduction of the computational effort. The probability $q(i)$ predicted for a certain configuration $i \in \mathcal{I}(\mathcal{V})$ by the induced Bayesian network can easily be computed according to Equation 2.2.1. The Kullback-Leibler divergence is non-negative and vanishes if and only if the two distributions p and q are identical. Hence, small values of $KL(p, q)$ indicate small prediction errors of the induced model.

A very popular variant is k -fold cross validation (often $k = 5, 10$), where a given data set is split into k parts of roughly equal size. When the j^{th} part of the data ($j = 1, \dots, k$) serves as test data, the other $k - 1$ parts are used as training data. Let q_j denote the probability distribution described by the model learned on the basis of these $k - 1$ parts of the data, while p_j designates the distribution implied by the j^{th} test data. Symmetry among the various parts of the data is achieved when this procedure is carried out for all $j = 1, \dots, k$, i.e. each part serves as test data at one point. The result of this procedure is then given by the mean and the standard deviation of the k values obtained for the Kullback-Leibler divergence.

B

Data Sets

Besides the artificial data sampled from the *Alarm Network* [8], we also used three real-world data-sets in our computer experiments,

- (SEW): the data gathered by Sewell and Shah [135],
- (ENV): the data set on *Environmental Influences on the Condition of Trees* [51],
- (BOS): the *Boston Housing Data* [12].

A brief description of the different data sets is provided in the following.

The Alarm Network

The *Alarm Network* was developed as a model for an emergency medical system [8]. This Bayesian network has become the most popular benchmark for assessing structural learning algorithms. It comprises 37 variables, each of which taking 2, 3 or 4 different discrete values. They are described in Table B.1. The structure of the alarm network is shown in Figure 5.16. Since it was built by experts, most of its 46 directed edges can be interpreted in a causal manner. The parameters of the Bayesian network model, that is the conditional probabilities regarding each variable and its parents, are taken from the alarm-network file on the *Netica* homepage [3]. We sampled data sets of various sizes from the joint probability distribution described by this model. For this task, we used the routines provided by the *Hugin* software [87].

variable	number of states	meaning
1	3	central venous pressure
2	3	pulmonary capillary wedge pressure
3	2	history of left ventricular failure
4	3	total peripheral resistance
5	3	blood pressure
6	3	cardiac output
7	3	heart rate obtained from blood pressure monitor
8	3	heart rate obtained from electrocardiogram
9	3	heart rate obtained from oximeter
10	3	pulmonary artery pressure
11	3	arterial-blood oxygen saturation
12	2	fraction of O ₂ in inspired gas
13	4	ventilation pressure
14	4	CO ₂ content of expired gas
15	4	minute volume, measured
16	3	minute volume, calculated
17	2	hypovolemia
18	2	left-ventricular failure
19	2	anaphylaxis
20	2	insufficient anesthesia or analgesia
21	2	pulmonary embolus
22	3	intubation status
23	2	kinked ventilation tube
24	2	disconnected ventilation tube
25	3	left-ventricular end-diastolic volume
26	3	stroke volume
27	2	catecholamine level
28	2	error in heart rate reading due to low cardiac output
29	3	true heart rate
30	2	error in heart rate reading due to electrocautery device
31	2	shunt
32	3	pulmonary-artery oxygen saturation
33	3	arterial CO ₂ content
34	4	alveolar ventilation
35	4	pulmonary ventilation
36	4	ventilation measured at endotracheal tube
37	4	minute ventilation measured at the ventilator

Table B.1: Key to the variables in the *Alarm Network* [8] (adapted from [142]).

The Data Set on Environmental Influences on the Condition of Trees (ENV)

The data set on *Environmental Influences on the Condition of Trees* (ENV) [51] comprises 11 variables and 6,168 cases. The variables are depicted in Table B.2. Two variables from the original data set were omitted in the computer experiments, because the one is basically an index and the other is not documented. Moreover, we reduced the number of states of the variables x_8 and x_9 such that x_8 (wind direction) took on four states (north, north-east; east, south-east; south, south-west; west, north-west) as did x_9 (≤ 610 ; $610 < \dots \leq 660$; $660 < \dots \leq 710$; > 710).

variable	original number of states	modified number of states	meaning
(omitted)	$\gg 100$	–	cluster number
1	6	6	needle loss of conifers in six categories
2	3	3	needle loss of conifers in three categories
3	2	2	needle loss of conifers in two categories
4	5	5	density of trees in forest
5	4	4	composition of forest cover
6	3	3	use of forest
7	5	5	soil at location of trees
(omitted)	9	–	not further documented
8	9	4	main wind-direction at location of trees
9	$\gg 10$	4	elevation (above sea level)

Table B.2: Key to the variables in the data set concerning *Environmental Influences on the Condition of Trees* [51].

The Boston Housing Data (BOS)

The *Boston Housing Data* (BOS) is concerned with the housing values in the Boston metropolitan area. It was first analyzed in [76] and later in [9]. We used the data set available from [12]. It contains 14 variables, discrete and continuous ones, and 506 cases. A key to the variables is given in Table B.3. For the use in our computer experiments we discretized this data set such that each variable became binary. For simplicity, only the univariate distribution for each variable was considered, and it was aimed at achieving about even cell counts for the two discrete states of a variable. More advanced schemes for discretizing data are described in [45, 52, 56, 114].

variable	meaning
1	median value of homes
2	crime rate
3	percentage land zoned for lots
4	percentage non-retail business
5	on Charles River
6	NO _x concentration
7	average number of rooms
8	percentage built before 1940
9	distance to employment centers
10	accessibility to radial highways
11	tax rate
12	pupil teacher ratio
13	percentage black
14	percentage lower-status population

Table B.3: Key to the variables in the *Boston Housing Data* [12]. We modified the variables such that each one became binary.

The Data of Sewell and Shah (SEW)

Sewell and Shah [135] gathered 10,318 cases from Wisconsin high-school students in order to investigate the influences which affect attending college later. It contains only 5 variables. Their meaning is shown in Table B.4. We used the data reproduced in [79].

variable	number of states	meaning
1	2	sex
2	4	socioeconomic status
3	4	intelligence quotient
4	2	parental encouragement
5	2	college plans

Table B.4: Key to the variables in the data set gathered by Sewell and Shah [135].

Bibliography

- [1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrox and F. Caski, editors, *Second International Symposium on Information Theory*, pages 267–81, Akademia Kiado, Budapest, 1973.
- [2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–23, 1974.
- [3] Alarm Network, from the Network Library of Norsys Software Corporation. <http://www.norsys.com/networklibrary.html>.
- [4] K. Ali and M. Pazzani. Classification using Bayesian averaging of multiple, relational rule-based models. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 207–17. Springer, 1996.
- [5] J. H. Badsberg. Model search in contingency tables by CoCo. *Computational Statistics*, 1:251–6, 1992. See also the homepage of his CoCo software, <http://www.math.auc.dk/~jhb/CoCo/cocoinfo.htm>.
- [6] O. Barndorff-Nielsen. *Information and Exponential Families*. John Wiley and Sons, 1978.
- [7] M. S. Bartlett. Contingency table interactions. *Journal of the Royal Statistical Society, Supplement*, 2:248–52, 1935.
- [8] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pages 247–56. London, 1989.
- [9] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics*. John Wiley and Sons, 1980.
- [10] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. John Wiley and Sons, 1994.
- [11] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis*. MIT Press, 1975.

- [12] Boston Housing Data. See for instance the StatLib – Datasets Archive, <http://lib.stat.cmu.edu/datasets/boston>.
- [13] R. Bouckaert. Belief network construction using the minimum description length principle. Technical report, UU-CS-1994-27, Department of Computer Science, Utrecht University, The Netherlands, 1993.
- [14] R. Bouckaert. *Bayesian Belief Networks: from Inference to Construction*. PhD thesis, Department of Computer Science, Utrecht University, The Netherlands, 1995.
- [15] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann, 1998.
- [16] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–40, 1996.
- [17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wodsworth Statistics / Probability Series, 1984.
- [18] W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, School of Computing Science, University of Technology, Sydney, Australia, 1990.
- [19] W. Buntine. Theory refinement on Bayesian networks. In B. D’Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann, 1991.
- [20] W. Buntine. A guide to the literature on learning graphical models. Technical Report IC-95-05, NASA Ames Research Center, 1995.
- [21] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, 1997.
- [22] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, Menlo Park, California, 1995.
- [23] J. Cheng, D. A. Bell, and W. Liu. An algorithm for Bayesian belief network construction from data. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 83–90, 1997.
- [24] J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 325–31, 1997.
- [25] S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–21, 1995.

-
- [26] D. M. Chickering. A transformational characterization of equivalent network structures. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 87–98. Morgan Kaufmann, 1995.
- [27] D. M. Chickering. Learning Bayesian networks is NP-complete. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 121–30, 1996.
- [28] D. M. Chickering. Learning equivalence classes of Bayesian network structures. In E. Horvitz and F. V. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 150–7. Morgan Kaufmann, 1996.
- [29] D. M. Chickering. Fast learning from sparse data. In K. Laskey and H. Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 109–15. Morgan Kaufmann, 1999.
- [30] D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. In E. Horvitz and F. V. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 158–68. Morgan Kaufmann, 1996.
- [31] D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997.
- [32] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–7, 1968.
- [33] G. Cooper and E. Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In B. D’Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 86–94. Morgan Kaufmann, 1991.
- [34] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–47, 1992.
- [35] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.
- [36] J. N. Darroch, S. L. Lauritzen, and T. P. Speed. Markov fields and log-linear models for contingency tables. *The Annals of Statistics*, 8:522–39, 1980.
- [37] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–80, 1972.
- [38] D. Dash and M. Druzdzel. A hybrid algorithm for the construction of causal models from sparse data. In K. Laskey and H. Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 142–9. Morgan Kaufmann, 1999.

- [39] A. P. Dawid. Conditional independence is statistical theory, with discussion. *Journal of the Royal Statistical Society, Series B*, 41:1–31, 1979.
- [40] A. P. Dawid. Statistical theory. The prequential approach. *Journal of the Royal Statistical Society, Series A*, 147:277–305, 1984.
- [41] W. E. Deming and F. F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11:427–44, 1940.
- [42] A. P. Dempster. Upper and lower probabilities induced by multivalued mapping. *The Annals of Mathematical Statistics*, 28:325–39, 1967.
- [43] A. P. Dempster, N. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm, with discussion. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
- [44] P. Domingos. Bayesian averaging of classifiers and the overfitting problem. In *Proceedings of the International Conference on Machine Learning*, pages 223–230. Morgan Kaufmann, 2000.
- [45] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.
- [46] D. Draper. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society, Series B*, 57:45–97, 1995.
- [47] D. Edwards. *Introduction to Graphical Modeling*. Springer, 1995.
- [48] D. Edwards and T. Havránek. A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72:339–51, 1985.
- [49] D. Edwards and T. Havránek. A fast model selection procedure for large families of models. *Journal of the American Statistical Association*, 82:205–13, 1987.
- [50] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.
- [51] Environmental influences on the condition of trees. Datasets at the Department of Statistics, University of Munich, Germany,
<http://www.stat.uni-muenchen.de/data-sets/floss/floss.html>.
- [52] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1022–7. Morgan Kaufmann, 1993.
- [53] Y. Freund and R. E. Shapire. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning*, pages 148–56. Morgan Kaufmann, 1996.

-
- [54] N. Friedman. The Bayesian structural EM algorithm. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 129–38. Morgan Kaufmann, 1998.
- [55] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–63, 1997.
- [56] N. Friedman and M. Goldszmidt. Discretization of continuous attributes while learning Bayesian networks. In *Proceedings of the International Conference on Machine Learning*, pages 157–65. Morgan Kaufmann, 1996.
- [57] N. Friedman, M. Goldszmidt, and A. Wyner. Data analysis with Bayesian networks: A bootstrap approach. In K. Laskey and H. Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 196–205. Morgan Kaufmann, 1999.
- [58] N. Friedman, M. Goldszmidt, and A. Wyner. On the application of the bootstrap for computing confidence measures on features of induced Bayesian networks. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 197–202, 1999.
- [59] N. Friedman and D. Koller. Being Bayesian about network structure. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 201–10. Morgan Kaufmann, 2000.
- [60] N. Friedman and I. Nachman. Gaussian process networks. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 211–9. Morgan Kaufmann, 2000.
- [61] R. M. Fung and S. L. Crawford. Constructor: A system for the induction of probabilistic models. In *AAAI-Proceedings of the National Conference on Artificial Intelligence*, pages 762–69, 1990.
- [62] K. R. Gabriel. Simultaneous test procedures – some theory of multiple comparisons. *The Annals of Mathematical Statistics*, 40:224–50, 1969.
- [63] D. Geiger and D. Heckerman. Learning Gaussian networks. In R. L. de Mantaras and D. Poole, editors, *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 235–43. Morgan Kaufmann, 1994.
- [64] D. Geiger, D. Heckerman, H. King, and Ch. Meek. Stratified exponential families: Graphical models and model selection. Technical report, MSR-TR-98-31, Microsoft Research, Redmond, Washington, 1998.
- [65] D. Geiger, D. Heckerman, H. King, and Ch. Meek. On the geometry of DAG models with hidden variables. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 76–85, 1999.

- [66] D. Geiger and Ch. Meek. Graphical models and exponential families. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 156–164. Morgan Kaufmann, 1998.
- [67] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–42, 1984.
- [68] W. Gibbs. *Elementary Principles of Statistical Mechanics*. Yale University Press, 1902.
- [69] P. Giudici and P. J. Green. Decomposable graphical Gaussian model determination. *Biometrika*, 86:785–801, 1999.
- [70] C. N. Glymour and F. G. Cooper, editors. *Computation, Causation & Discovery*. AAAI Press, 1999.
- [71] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society, Series B*, 14:107–14, 1952.
- [72] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–32, 1995.
- [73] M. Haft, R. Hofmann, and V. Tresp. Model-independent mean-field theory as a local method for approximate propagation of information. *Network: Computation in Neural Systems*, 10:93–105, 1999.
- [74] J. M. Hammersley and P. E. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.
- [75] E. J. Hannan. The estimation of the order of an ARMA process. *The Annals of Statistics*, 8:1071–81, 1980.
- [76] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 1978.
- [77] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.
- [78] D. M. A. Haughton. On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, 16(1):342–55, 1988.
- [79] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 301–54. Kluwer Academic Publishers, The Netherlands, 1996.
- [80] D. Heckerman and D. Geiger. Learning Bayesian networks: A unification for discrete and Gaussian domains. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 274–84. Morgan Kaufmann, 1995.

-
- [81] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
- [82] D. Heckerman, Ch. Meek, and G. Cooper. A Bayesian approach to causal discovery. Technical report, MSR-TR-97-05, Microsoft Research, Redmond, Washington, 1997.
- [83] E. Herskovits and G. Cooper. Kutató: an entropy-driven system for the construction of probabilistic expert systems from data. In P. Bonissone, M. Henrion, I. Kanal, and J. Lemmer, editors, *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 54–62. Morgan Kaufmann, 1990.
- [84] K.-U. Höffgen. Learning and robust learning of product distributions. Technical Report 464, Department of Computer Science, University of Dortmund, Germany, 1993.
- [85] R. Hofmann. *Lernen der Struktur nichtlinearer Abhängigkeiten mit graphischen Modellen*. PhD thesis, Department of Computer Science, Technical University of Munich, Germany, 2000. Verlag dissertation.de, Berlin, Germany, ISBN 3-89825-131-4.
- [86] R. Hofmann and V. Tresp. Discovering structure in continuous variables using Bayesian networks. In *Proceedings of the Conference on Advances in Neural Information Processing systems 8*, pages 500–6. MIT Press, 1996.
- [87] HUGIN Expert A/S. Aalborg, Denmark, <http://www.hugin.com>.
- [88] H. Jeffreys. *Theory of Probability*. Oxford University Press, 1961.
- [89] F. V. Jensen. *An Introduction to Bayesian Networks*. University College London Press, 1996.
- [90] D. S. Johnson, Ch. H. Papadimitriou, and M. Yannakakis. How easy is local search? In *26th Annual Symposium on Foundations of Computer Science*, pages 39–42, 1985.
- [91] M. I. Jordan, editor. *Learning in Graphical Models*. Kluwer Academic Publishers, The Netherlands, 1998.
- [92] R. L. Kashyap. Optimal choice of AR and MA parts in autoregressive moving average models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:99–104, 1982.
- [93] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90:773–96, 1995.
- [94] R. E. Kass and L. Wasserman. A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. *Journal of the American Statistical Association*, 90:928–34, 1995.
- [95] R. W. Katz. On some criteria for estimating the order of a Markov chain. *Technometrics*, 23:243–9, 1981.

- [96] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In D. Geiger and P. Shenoy, editors, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 302–13. Morgan Kaufmann, 1997.
- [97] R. Korf. Linear-space best-first search. *Artificial Intelligence*, 62:41–78, 1993.
- [98] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.
- [99] W. Lam and F. Bacchus. Using causal information and local measures to learn Bayesian networks. In D. Heckerman and A. Mamdani, editors, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 243–50. Morgan Kaufmann, 1993.
- [100] S. L. Lauritzen. *Lectures on Contingency Tables*. Department of Mathematics, Aalborg University, Denmark, third edition, 1989.
- [101] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
- [102] S. L. Lauritzen. Causal inference from graphical models. Technical report, R-99-2021, Department of Mathematical Sciences, Aalborg University, Denmark, 1999.
- [103] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.
- [104] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems, with discussion. *Journal of the Royal Statistical Society, Series B*, 50(2):240–65, 1988.
- [105] S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57, 1989.
- [106] H. Linhart and W. Zucchini. *Model Selection*. John Wiley and Sons, 1986.
- [107] D. Madigan, S. A. Andersson, M. D. Perlman, and C. T. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics – Theory and Methods*, 25(11):2493–519, 1996.
- [108] D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam’s window. *Journal of the American Statistical Association*, 89:1535–46, 1994.
- [109] D. Madigan, A. E. Raftery, J. York, J. M. Bradshaw, and R. G. Almond. Strategies for graphical model selection. In P. Cheeseman and R. Oldford, editors, *Selecting Models from Data: Artificial Intelligence and Statistics IV*, pages 91–100. Springer, 1993.
- [110] D. Madigan and J. York. Bayesian graphical models for discrete data. Technical Report 259, Department of Statistics, University of Washington, Seattle, Washington, 1993.

-
- [111] Ch. Meek. Strong completeness and faithfulness in Bayesian networks. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 411–8. Morgan Kaufmann, 1995.
- [112] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–92, 1953.
- [113] S. Monti and G. F. Cooper. Learning Bayesian belief networks with neural network estimators. In *Proceedings of the Conference on Advances in Neural Information Processing systems 9*, pages 579–84. MIT Press, 1997.
- [114] S. Monti and G. F. Cooper. A multivariate discretization method for learning Bayesian networks from mixed data. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 404–13. Morgan Kaufmann, 1998.
- [115] R. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, CRG-TR-91-2, Department of Computer Science, University of Toronto, Canada, 1993.
- [116] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [117] J. Pearl. Causal digrams for empirical research. *Biometrika*, 82:669–710, 1995.
- [118] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, 2000.
- [119] A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Department of Computer Science, Stanford University, 1999.
- [120] Esprit Project 28932-PRONEL: Probabilistic Networks and Learning. Funded by the European Union. Partners: HUGIN Expert A/S, Schlumberger Telecom, Siemens AG, TIGA Technologies. Sept. 1998 – Febr. 2000.
- [121] PVM web site. http://www.epm.ornl.gov/pvm/pvm_home.html.
- [122] A. Raftery. Hypothesis testing and model selection. In *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.
- [123] A. Raftery. Bayes factors and BIC. Technical report, No. 347, Department of Statistics, University of Washington, Seattle, Washington, 1998.
- [124] M. Ramoni and P. Sebastiani. Learning Bayesian networks from incomplete databases. In D. Geiger and P. Shenoy, editors, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 401–8. Morgan Kaufmann, 1997.
- [125] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–71, 1978.

- [126] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11:416–31, 1983.
- [127] J. Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14:1080–100, 1986.
- [128] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49:223–39, 1987.
- [129] R. W. Robinson. Counting unlabeled acyclic digraphs. In C. H. C. Little, editor, *Lecture Notes in Mathematics 622: Combinatorial Mathematics V*. Springer, 1977.
- [130] D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–92, 1976.
- [131] R. Sangüesa and U. Cortés. Learning causal networks from data: a survey and a new algorithm for recovering possibilistic causal networks. *AI Communications*, 10:31–61, 1997.
- [132] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–64, 1978.
- [133] S. L. Sclove. Small-sample and large-sample statistical model selection criteria. In P. Cheeseman and R. W. Oldford, editors, *Selecting Models from Data*, pages 31–39. Springer, 1994.
- [134] R. Settimi and J. Q. Smith. On the geometry of Bayesian graphical models with hidden variables. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 472–9. Morgan Kaufmann, 1998.
- [135] W. Sewell and V. Shah. Social class, parental encouragement, and educational aspirations. *American Journal of Sociology*, 73:559–572, 1968.
- [136] G. R. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [137] R. Shibata. Selection of the order of an autoregressive model by Akaike’s information criterion. *Biometrika*, 63:117–26, 1976.
- [138] M. Singh and M. Valtorta. An algorithm for the construction of Bayesian network structures from data. In D. Heckerman and A. Mamdani, editors, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 259–65. Morgan Kaufmann, 1993.
- [139] D. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed acyclic graphical structures. *Networks*, 20:579–605, 1990.
- [140] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.

-
- [141] P. Spirtes, C. Glymour, and R. Scheines. Causality from probability. In J. Tiles, G. Mc-Kee, and G. Dean, editors, *Evolving Knowledge in the Natural and Behavioral Sciences*, pages 181–99. Pitman, London, 1990.
- [142] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer Lecture Notes in Statistics 81, 1993.
- [143] H. Steck. On the use of skeletons when learning in Bayesian networks. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 558–65. Morgan Kaufmann, 2000.
- [144] H. Steck and V. Tresp. Bayesian belief networks for data mining. In *Proceedings of the 2nd Workshop "Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme" (DMDW99)*, pages 145–54. LWA Sammelband, ISBN 3-929757-26-5, University of Magdeburg, Germany, 1999.
- [145] M. Stone. Cross-validators choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B*, 36:111–47, 1974.
- [146] J. Suzuki. A construction of Bayesian networks from databases based on the MDL scheme. In D. Heckerman and A. Mamdani, editors, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 266–73. Morgan Kaufmann, 1993.
- [147] TETRAD project homepage.
<http://hss.cmu.edu/html/departments/philosophy/TETRAD/tetrad.html>.
- [148] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In P. Bonissone, M. Henrion, I. Kanal, and J. Lemmer, editors, *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 255–70. Morgan Kaufmann, 1990.
- [149] N. Wermuth and S. L. Lauritzen. Graphical and recursive models for contingency tables. *Biomatrika*, 70:537–52, 1983.
- [150] N. Wermuth and S. L. Lauritzen. On substantive research hypotheses, conditional independence graphs and graphical chain models, with discussion. *Journal of the Royal Statistical Society, Series B*, 52:21–72, 1990.
- [151] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley and Sons, 1990.
- [152] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–59, 1992.
- [153] S. Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–85, 1921.
- [154] L. A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11:199–228, 1983.